HAW
HAMBURG

BACHELORTHESIS
Thien Phuc Tran

Application of Multi-Fusion Network for Human-Object Interaction Detection

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Thien Phuc Tran

Application of Multi-Fusion Network for Human-Object Interaction Detection

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Thomas Clemen

Eingereicht am: 10. December 2019

Thien Phuc Tran

Thema der Arbeit

Anwendung von Multi-Fusion Network zur Erkennung von Mensch-Objekt-Interaktion

Stichworte

Deep learning, Multi-Fusion Network, Human-Object Interaction Detection, Handlungserkennung, Objektklassifizierung

Kurzzusammenfassung

Diese Abschlussarbeit stellt Multi-Fusion Network Architektur für Erkennung von Mensch-Objekt-Interaktion mit mehreren Kameras vor und implementiert eine Anwendung für einen spezifischen Anwendungsfall eines Getränkekühlschranks. Die Abschlussarbeit präsentiert einfache jedoch effektive Vorgehensweisen zur Reduzierung der erforderten Trainingsdatenmenge und des Risikos von Overfitting, insbesondere im Umgang mit kleinem Datensatz, der üblich von individueller Person oder kleiner Organisation aufgenommen wurde. Das Modell erreichte eine Testgenauigkeit von 91.235% und ein vergleichbares Ergebnis im praktischen Test an der Veranstaltung Solutions Hamburg 2019. Multi-Fusion Network ist leicht zu skalieren durch gemeinsame lernbare Parameter und auch so leichtgewichtig, dass es auf kleine Geräte mit durchschnittlicher Rechenleistung laufen kann. Multi-Fusion Network könnte für Indoor-Aktivitäten Erkennung für Smarthome Anwendungen oder Gaming-Erlebnis angewendet werden.

Thien Phuc Tran

Title of Thesis

Application of Multi-Fusion Network for Human-Object Interaction Detection

Keywords

Deep learning, Multi-Fusion Network, Human-Object Interaction Detection, Action Recognition, Object Classification

Abstract

This thesis proposes Multi-Fusion Network architecture for human-object interaction detection with multiple cameras and implements an application for a specific use case of a drink refrigerator. The thesis also introduces simple but effective approaches for minimizing the required amount of training data and the risk of overfitting, especially when dealing with a small dataset that is commonly recorded by a person or small organization. The model achieved 91.235% test accuracy and comparable result in the real-world test at the event Solutions Hamburg 2019. Multi-Fusion Network is easy to scale thanks to shared learnable parameters. It is also lightweight to run on small devices with average computation capability and, therefore, can be used for smart home applications, gaming experiences, or augmented reality.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Human-object interaction detection

Computers were invented to help people complete their tasks more comfortably, faster, and better. In the early ‘90s, the first smartphone appeared in the market. It worked as a pocket computer to help people do their everyday tasks everywhere without having to sit in front of a computer. One must only take his phone out, and perform some swipes and some touches on the screen. Nowadays, in the era of Internet-of-Things, people want to put computers in everything, where limitations start to become apparent. Computers cannot anticipate or recognize human activities without the user having to give them a clue or somehow telling them about his intent. This problem can be solved in different ways, either improving user experiences for more convenience, which forms the research field Human-Computer Interaction or letting computers understand what is happening in the surroundings and automatically know what to do.

Human-object interaction detection is a computer vision research field that 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 manners. There has been much hype around deep learning in the last decade, especially in computer vision fields since they are drastically 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 [13] and InteractNet [5] are also convolutional neural networks with special architecture. Besides, HAKE and InteractNet involve human action embedding and region proposal networks. Such complex approaches require high computational capability; therefore, it would be overkill 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.

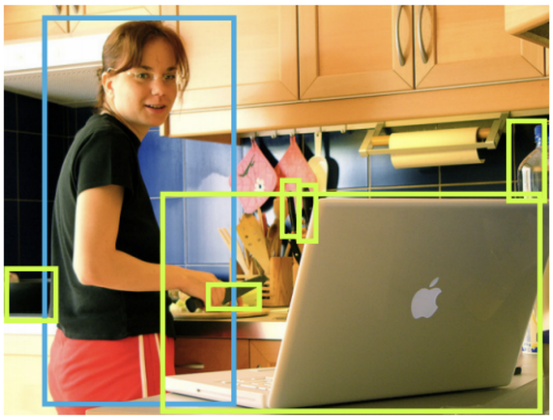


Figure 1.1: Region proposal network [5]

1.2 Cooperation of multiple cameras

A monitoring system that uses a single camera could struggle to work reliably caused by dead zones, blind spots of the camera setup, or occlusions that occur when an object hides another object.

For that reason, many systems use multiple cameras 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 constructing an algorithm to merge observed information from all cameras into a final one. 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 can lead to further problems.

A human can understand actions regardless of view angles because he/she has been seeing and learning them since he/she was born. If one can not tell which action is being performed, one tries to move around and look at the action from different view angles before one can undoubtedly answer. In consideration of this intuition, deep learning is well-suited for solving this kind of problem without requiring any calibration or having to follow any installation rules.

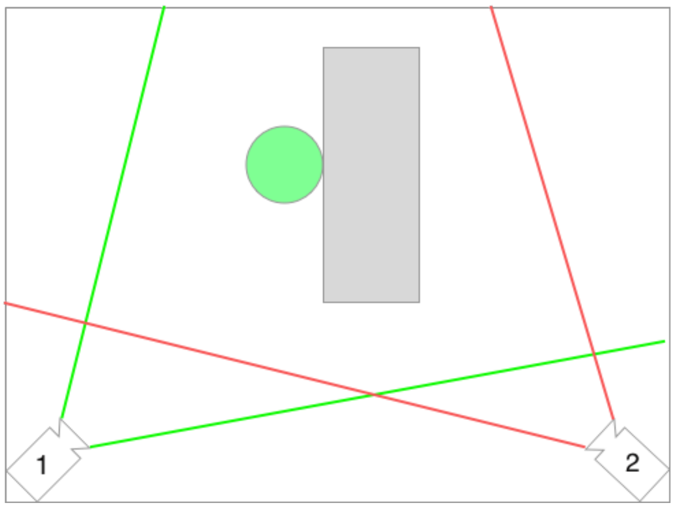


Figure 1.2: The occlusion problem: Only the camera 1 can see the object

1.3 Objectives

The thesis proposes a neural network architecture and some novel techniques to solve the problem. A practical use case is also implemented for testing and evaluating purpose: a refrigerator with multiple cameras and a computer inside that can recognize fundamental interactions (taking, putting, or nothing) between users and objects (drinks, fruits). The objectives of the thesis are:

- To create a small dataset with multiple cameras for training and evaluation.
- To construct a system that produces reasonable results or at least better than random and to evaluate its potential.
- To research new techniques to enhance the overall performance further. The major aim here is not only high accuracy but also the real-time capability of the system so that the system can work continuously and give instant feedback back to users.

1.4 Structure

The thesis is split into six chapters. The **first chapter** gives a brief introduction to the problem of human-object interaction detection and why the author constructs a new architecture instead of using state-of-the-art methods. This chapter also explains the reason for using multiple cameras and difficulties when dealing with multiple cameras.

The **second chapter** aims to introduce common concepts and best practices in computer vision and propose the architecture. The core building blocks, their inspiration, and motivation are explained. This chapter also discusses some considerations during architecture construction.

The **third chapter** covers the process of recording a dataset, difficulties, and considerations during the process. This chapter also discusses some data augmentation techniques that can be applied in this work and define a new term of "consistency" and different levels of consistency when augmenting data.

The **fourth chapter** presents pre-processing techniques, attention mechanisms, training techniques, and considerations to speed up the training process. Experiment results and practical test results are reported at the end of the chapter.

The **fifth chapter** analyzes difficulties and discusses different approaches to make the system capable of working with continuous video streams and working fast enough to perform all calculations in real-time.

The **last chapter** summarizes all the work done, discusses the potential of the proposed architecture and techniques and future work that can be done on top of this thesis.

2 Model architecture

The core concept of Multi-Fusion Network is to split tasks, solve, and then combine them again to get the final result in a divide-and-conquer manner. However, splitting tasks into small ones that can easily be solved, and combining information from outputs in deep learning requires some considerations.

2.1 Input

Input in this project is a sequence of frames captured by all the cameras mounted in the refrigerator. Each input is assumed to contain only one action and one object. The model needs to recognize both for each input. There are three options to consider:

1. Each possible pair of action-object is encoded as one class.
2. Actions and objects are separately encoded as individual classes; a single model predicts for each input an action-object pair.
3. Actions and objects are separately encoded as individual classes; two separate models predict action and object.

The first option theoretically can be used, but then the problem is not well modeled. The model should learn the actual movement patterns of actions and visual features of objects to be robust against variations. Should a new action be added, this action must also be recorded with all existing object classes in the dataset and vice versa, leading to extraordinarily high but unnecessary costs.

The second option divides the problem into two small ones, meaning there are two classification heads, one for action classification and the other one for object classification. This option seemed to be the right choice at first since each part of the model has its

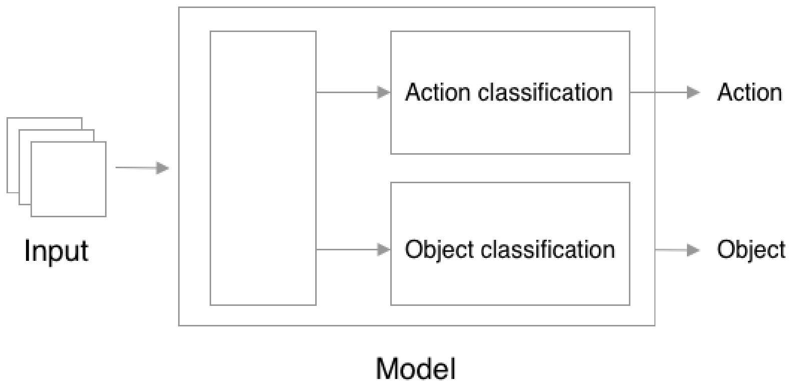


Figure 2.1: Second approach

particular job. It was, therefore, implemented and experimented. Actions were relatively well recognized, but objects were often misclassified.

It turns out, the fact that only one object can be put (or taken) in each input can be exploited to enhance the model while keeping the same data usage. The third approach divides the model into two parts. The action classification model receives video as input, while the object classification model receives video frames instead. In doing so, the object classification model now has much more training data.

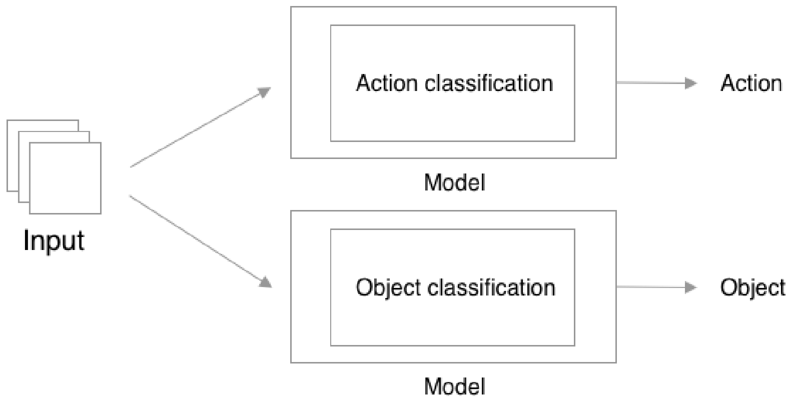


Figure 2.2: Third approach

This approach still encounters another issue with video frames. Not all video frames of "taking apple" can be used for class "apple" because there are also moments, at which the hand is not holding anything when it even moves or is visible. Some pre-processing techniques can solve this problem, which will be discussed in chapter 4.

2.2 Feature extractor and Transfer learning

2.2.1 Feature extraction

Deep learning models make a prediction based on features. In computer vision tasks, a feature is a unique piece of information that describes observed characteristics in the image, such as line, curve, color, or more intricate detail. Despite visual information it contains, features are encoded as numbers, forming a multi-dimensional feature space.

Feature extractor is responsible for finding and extracting informative visual features in input images into their corresponding numeric forms, allowing classifiers (or regressors) to use this information more effectively rather than just looking at pixels matrices. Feature extractor is, therefore, a crucial building block in deep learning for computer vision.

Convolutional neural network

A feature extractor usually is a convolutional neural network. It consists of multiple 2D convolutional layers for recognizing patterns, and pooling layers (max pooling or average pooling) for reducing input sizes or smoothing input. A convolutional layer contains a set of rectangular parallelepiped filters. These filters are small in terms of width and height, commonly ranging from 1 to 7, and have the same depth as that of input (depth is 3 if the input is an RGB image).

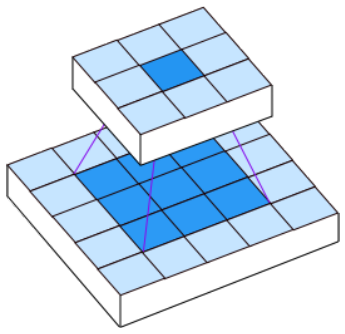


Figure 2.3: How convolutional layer works

Instead of connecting an input to all neurons at once during the forward pass, convolutional filters slide across the input and compute dot products as usual. In other words, neurons will be connected to a small region of input at each step, resulting in a so-called receptive field that tries to find patterns in local regions. Each filter is responsible for finding a specific visual pattern.

Pooling layers

A pooling layer is often inserted between successive convolutional layers. Pooling layers can help to reduce the width and height of input passing to the next convolutional layer and, therefore, reduce the number of learnable parameters, computational costs, and risks of overfitting. Two common types of pooling are max pooling and average pooling.

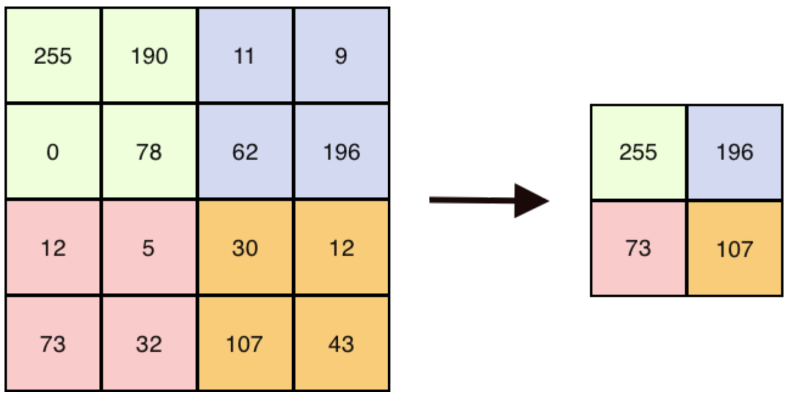


Figure 2.4: How max pooling works

Max pooling uses a filter that slides across the input and returns the max value (the brightest pixel) in the window at each step, while average pooling's filter returns the average value of that window instead. Max pooling is useful when one wants to reduce image size while keeping significant visual patterns. Average pooling is often used to smooth out input images.

Residual neural network architecture

Constructing an efficient convolutional neural network is not simply stacking one convolutional layer on top of another. Researchers have published various architectures in the last few years such as AlexNet [11], Inception [22], VGG [17] and ResNet [7] with a wide range of accuracy and computational complexity.

The ResNet architecture is used in this work as a feature extractor because of its simplicity and efficiency. The core of ResNet architecture is the residual block, which contains a skip connection (or identity mapping). Accuracy of a neural network will get saturated and begins to degrade at some point when one tries to keep increasing its depth and. Skip connections help to construct a deeper model without suffering from performance degradation. Besides, stacking too many layers will cause the problem of vanishing gra-

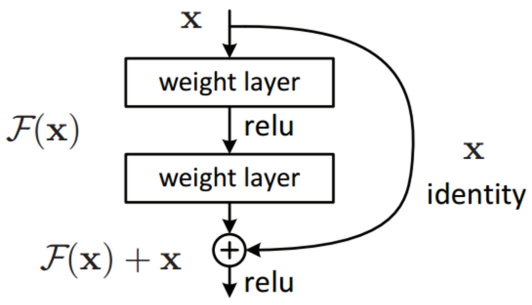


Figure 2.5: A building block in ResNet architecture with skip connection [7]

dient. Larger gradients can be backpropagated through skip connections to the initial layers; this not only helps to avoid vanishing gradient but also maintains the learning speed of initial layers as high as that of final layers.

2.2.2 Transfer learning

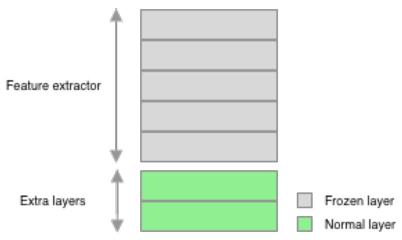
The intuition behind transfer learning is that humans can transfer knowledge across tasks; in other words, humans can utilize the acquired knowledge to solve related tasks. Based on that idea, transfer learning is a method that reuses a pre-trained model developed for a task as a basis for developing a model for another task. People rarely train a model from scratch, especially in computer vision fields, because it is hard to have sufficient data, and it costs too much time to do so. Also, researches have proved that using a pre-trained base model can help to converge faster than training from scratch. When a model should be trained with a small dataset for object detection, it is a best practice to utilize a pre-trained model. A base model pre-trained with ImageNet Dataset [4] not only can help to converge faster but also to achieve better accuracy because ImageNet is one of the largest datasets and contains many visual features that exist in real life.

There are two common application scenarios of transfer learning:

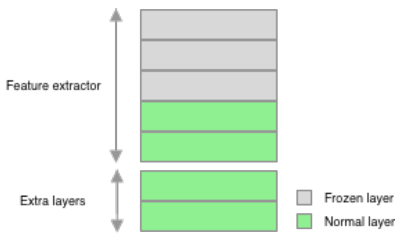
1. **Fixed feature extractor:** Freeze the model's weights and remove the last layer (classifier). The extracted features can be used to classify other objects or for

further processing.

2. **Fine-tuning:** Freeze nothing or only parts of the model and remove the classifier. Weights of those layers will also be updated after backpropagation.



(a) Fixed feature extractor



(b) Fine-tuning

Figure 2.6: Common approaches to transfer learning

An experiment is performed to find out which strategy is best for this specific use case. The base model was completely frozen at the beginning. However, it turns out that video frames differ from normal images a lot since they also contain blurry visual features of moving objects; as a result, the model could only achieve very low accuracy. Therefore, two last convolutional blocks of the base model are unfrozen for the model to learn those blurry features. As a result, accuracy significantly increased.

2.3 Concepts

2.3.1 Temporal fusion

Many research papers have proposed various approaches for video understanding such as 3D Convolutional Network, Inflated 3D Convolutional Network [2], CNN-RNN and especially Temporal Relation Network (TRN) [24].

This project applies the idea of TRN for temporal fusion block. Based on Temporal Segment Network (TSN) [23], TRN can learn to reason relations between changes of entities along the time axis. Rather than using optical flow to learn movement patterns, TRN only sees 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 project 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 cost.

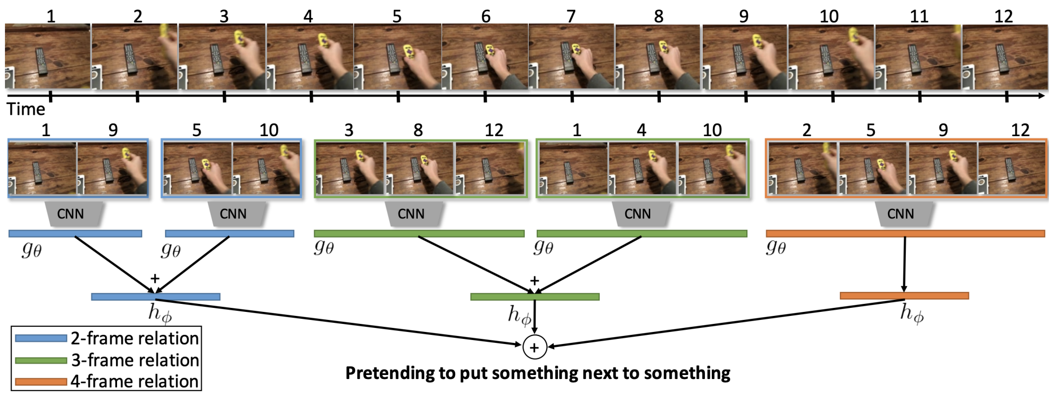


Figure 2.7: How Temporal Relation Network works [24]

Although TRN can be implemented with multi-scale time relations, single scale time relation is used in this work. 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 representative positions of movement are completely captured because every hand position is crucial to understand actions, in this case, a skipped snippet can also lead to 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. The temporal fusion block can be a fully connected layer as in the original paper or a more complex block. In this project, the acquired information is fed into a Gated Recurrent Unit (GRU) [3], 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."

Gated recurrent unit and sequence understanding

Gated recurrent unit is a mechanism in recurrent neural networks – a common type of neural network for processing sequences thanks to its natural architecture. Recurrent neural networks are neural networks with a loop in themselves. They can store the output of previous calculation (or hidden state), which will contribute to successive calculations. In other words, recurrent networks receive two inputs: actual input and hidden state. The hidden state is updated after each calculation, forming a kind of short-term memory. Recurrent neural networks can be applied for understanding sequences because the final

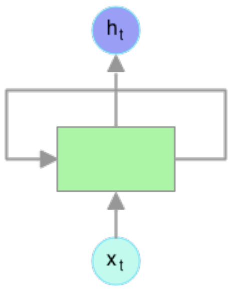


Figure 2.8: Loop in recurrent neural networks

hidden state, contains information of every time step so that it can ideally encode the meaning of the whole input sequence.

2.3.2 View fusion

Due to the limitations of camera perspectives, a camera may not be able to see the 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. The ideal mapping of temporal fusion block (before view fusion)

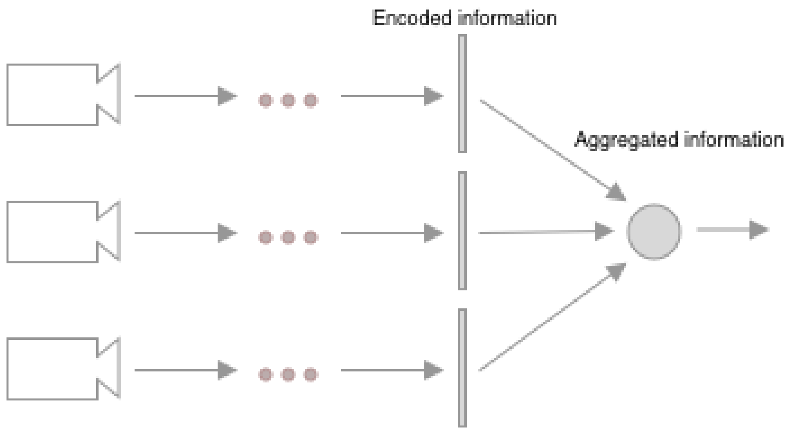


Figure 2.9: The illustration of how view fusion works

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

Process view

The pre-processing blocks contain pre-processing operations such as inactivity removal, background subtraction, and density-based cropping, which clean the input and prepare them for further processing. Those operations will be described further in Chapter 4 and Chapter 5.

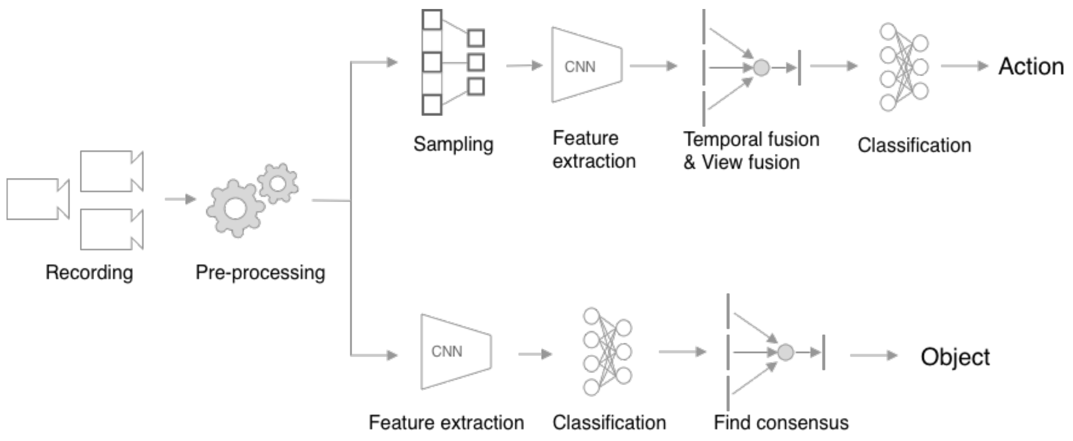


Figure 2.10: Architecture overview

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.

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 consensus. Different policies can be applied to find consensus. The most frequently classified object (with a high confidence score) will be chosen in this project.

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. One can conclude that Multi-Fusion Network architecture is scalable because an increasing number of cameras does not involve an increasing amount of parameters.

2.5 Discussion

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 dataset more effectively and make the models more intuitive and trivial 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.

Feature extractor in this project is a pre-trained convolutional network model – ResNet. ResNet architecture has skip connections, which allows constructing a deeper model without suffering from vanishing gradients and performance degradation. Transfer learning helps to reduce the amount of work and data needed to achieve good performance. The feature extractor is pre-trained on the ImageNet dataset, which contains a lot of common visual features in real life; this helps the model to learn faster and better. Due to blurry features of movements that only exist in videos, the last two convolutional blocks of the pre-trained feature extractor are not frozen for them to adopt new features.

The temporal fusion block looks at every frame from a single camera and learns to understand movement patterns by using a gated recurrent unit. The ideal mapping function of temporal fusion should give a vector containing informative movement information without respect to the camera angle so that those vectors from multiple cameras can be easily aggregated into one final movement describing vector in the view fusion stage.

Despite having multiple input sources, all learnable parameters in Multi-Fusion network architecture are shared. The fact that all weights are shared brings good scaling potential because adding more cameras will not increase the number of parameters.

3 Dataset

3.1 Objects, actions and camera placements

This dataset contains 7 object classes and 3 actions, shot by MQ013CG-E2 cameras (1.3 Megapixel) with wide-angle lenses at 20 FPS.

Figure 3.1: The camera and lens used to record the dataset

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. For instance, Lemonaid+ Blutorange and Lemonaid+ Limette share the same shape, but different colors (pink-red and green); Granini Orange and Apple have different shapes but have quite the same color.

An object will be taken, put, or nothing happens at all in each video. Four cameras are used to capture actions from different angles, preventing objects from being fully covered

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

Table 3.1: Objects and actions in the dataset

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 working with low-resolution images correctly.

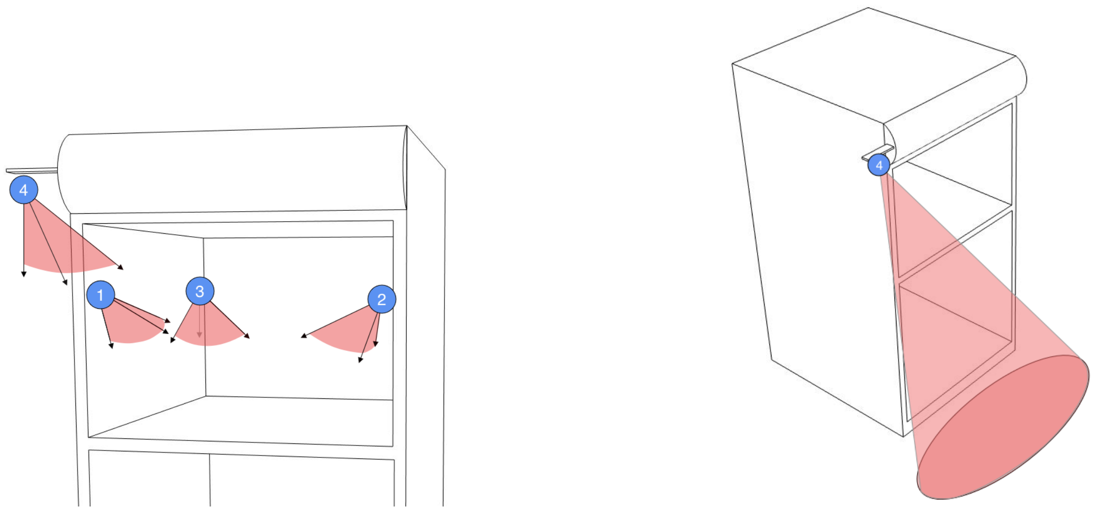


Figure 3.2: Camera positions

In the beginning, all four cameras are used to record 4 × 1818 videos. However, the model trained with those videos was strongly overfitted regardless of how hyperparameters are 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 dataset. 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.

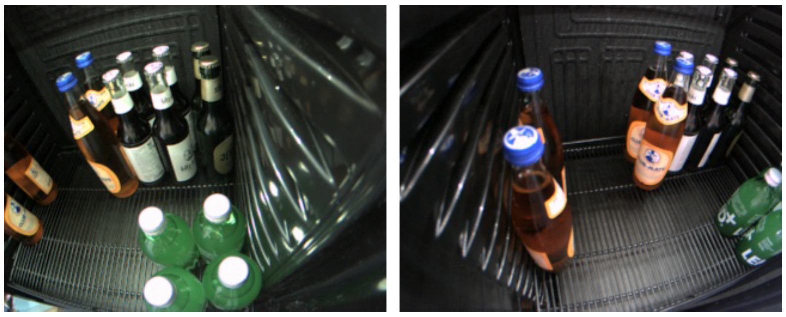


Figure 3.3: Images captured by camera 1 and 2

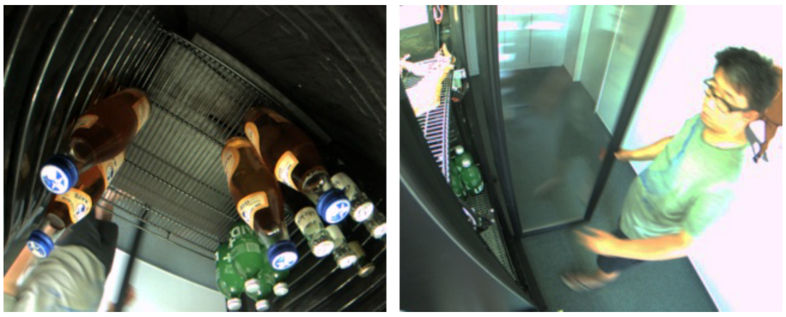


Figure 3.4: Images captured by camera 3 and 4

Due to the overfitting problem, the third and fourth cameras are removed. The model did show significant improvement by making reasonable classifications, accuracy increased slowly epoch-by-epoch and stopped at about 50% and about 30% for action classifier and object classifier, respectively. The evaluation was performed on a model with base architecture – i.e., without additional blocks, which will be introduced in Chapter 4 and helped a lot to boost performance significantly.

New dataset, new recording process

A new dataset must be recorded with only two cameras. Now dataset is split into two parts: action dataset and object dataset. The action dataset will be recorded as usual: each video contains one action on an object (or nothing). The object dataset takes much less time to record: A person holds an object in front of the cameras and tries to interact with that object by performing actions such as turning, rotating, holding, and so forth. The main benefit of this splitting is that it will be easier to add a new object to the dataset, and the teaching process makes a lot more sense. Should the refrigerator learn a new object, all one must do are just holding that object in front of the cameras and moving it around for some minutes. Furthermore, teaching the model to learn a new action is also more comfortable. One does not need to interact with every object available in the dataset but on an arbitrary object.

3.2 Data augmentation

Data augmentation is an essential technique in machine learning, helps to train the model to be robust against variations, and to be able to generalize better. The key idea is to transform input in different ways in order to enrich the dataset. The model can then learn various variations of a single input and recognize its pattern later with ease. For example, a cat detection model with data augmentation will recognize real cats much better than one without data augmentation, because cat images in real life can have numerous positions as well as camera angles.

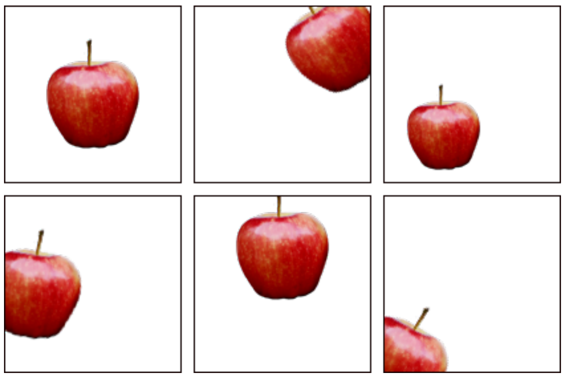


Figure 3.5: Examples of image augmentation

Data augmentation is especially popular when working with images. The same augmentation techniques can also be applied to video datasets since videos are just stacks of images. Depending on use cases, some constraints must be considered instead of blindly applying all transformations.

3.2.1 Consistency

The aim of augmenting is to generate more samples based on existing data. Those generated data must, however, be as realistic as possible. It would make no sense and bring no benefit if generated data are too unrealistic, or such data that the model will never work on. Consistency means, in this work, constancy or slow changes of some properties in a data sample. For example, applying random flip on an image dataset would be fine, but will cause adverse effects on a video dataset, when video frames are suddenly flipped back and forth. Two levels of consistency are defined as follows:

Consistency within a single camera

This type of consistency ensures that the augmented video still looks realistic and reasonable. In other words, if a transformation is applied on a frame, the same or a comparable transformation must also be applied to the remaining frames, so that the augmented video can maintain continuity.

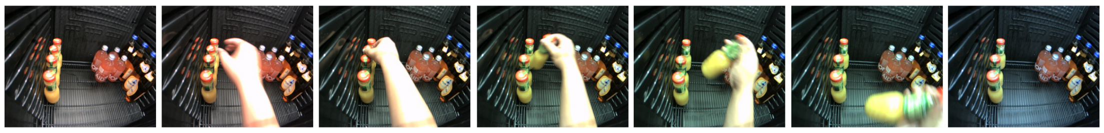


Figure 3.6: Consistency within a single camera

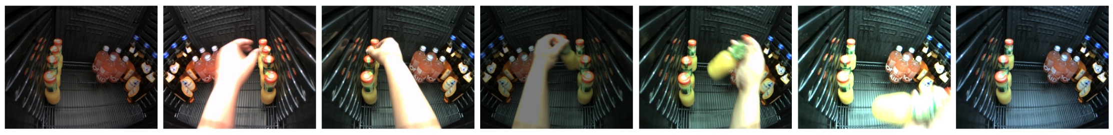


Figure 3.7: Inconsistency within a single camera

Consistency across multiple cameras

Consistency across multiple cameras requires the videos to be already consistent within a single camera. Furthermore, the videos must be consistent concerning both space and time, so that augmented videos still can maintain the effect all cameras were filming at the same time at the same place. Is a video reversed or flipped, the others must also be reversed or flipped.

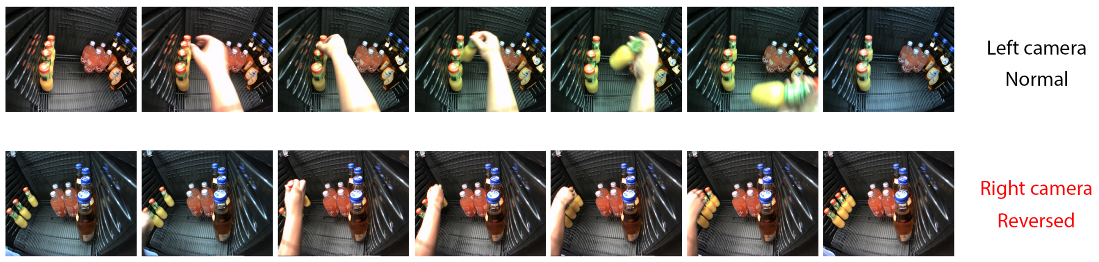


Figure 3.8: An example of not fulfilling multi-camera consistency

Correctly determining the consistency level for transformations will help to gain more benefit from data augmentation. The lower the consistency level is required, the more data can be generated. Choosing the wrong consistency level for just one transformation can even lead to big problems in model performance. The next sections will discuss the possible transformations and their required consistency level.

3.2.2 Brightness

Brightness affects the color intensity and, therefore, also object classification performance. Changing brightness helps the model to be robust against dynamic lighting conditions. The cameras in this work are configured with Auto-Exposure/Auto-Gain (AEAG), which means exposure and gain values are automatically modified if lighting conditions change. AEAG does help to achieve good and stable brightness. However, there is still a crossing phase when lighting conditions change, where some darker (or lighter) images are captured and evaluated before new exposure, and gain values are calculated.

Each individual camera has its own exposure and gain values, which not necessarily need to be equal to or proportional to that of other cameras. In addition, lighting conditions can suddenly change, and two consecutive frames can then have two brightness values with a large distance. It can be concluded that brightness augmenting requires no consistency.

The brightness value varies randomly from 0.9 to 1.1 in this project because this interval reflects all lighting changes that can be performed during the experiment.

3.2.3 Horizontal flipping

Horizontal flipping is one of the most popular techniques in image augmentation. Horizontal flipping helps to vary the position and posture of objects. It is obvious that horizontal flipping requires consistency within a single camera. It is important to notice that the model should learn to understand and to recognize actions and objects from different points of view. Once something changes in a camera, it may also change in some other cameras in the very same way. Depending on the relative position between the cameras, horizontal flipping may also require consistency across some or all cameras.

Understanding physical changes across all cameras plays an important role; synchronizing posture of objects and human hands across all cameras is therefore also relevant. In this project, horizontal flipping subjects to require multi-camera consistency.

3.2.4 Zooming

Zooming in or out helps mainly to generate objects in different sizes so that the model can recognize them better. Keeping the size of objects in sync across all cameras is overkill for this project. Therefore, single-camera consistency's constraints are applied. Each camera can have different but constant zoom ratios because the cameras have fixed focal length and statically mounted.

3.2.5 Reversing and label mapping

Reversing video is the most trivial option to augment when it comes to the time axis. Multi-camera consistency is obviously required since the same sequence of action must be observed by all cameras. One should notice that reordering frames can also lead to changes in video meaning. However, that consequence is the most meaningful advantage of this technique: hand positions and postures can be reused to generate other actions.

The target label of actions must be re-mapped after reversing, as shown in Figure 3.9.

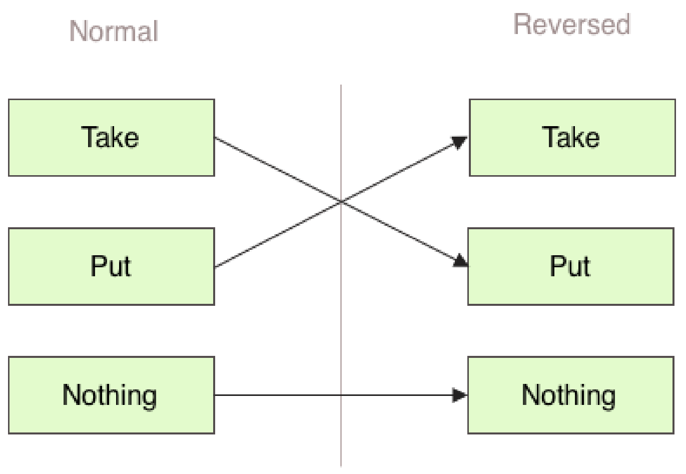


Figure 3.9: Action mapping schema

Short snippets can also be individually repeated and reversed. In this case, labels do not have to be re-mapped. However, this is not experimented in this project due to its complexity, and the cameras are not exactly synchronized in time with each other. This type of augmentation should be experimented and evaluated in future works.

3.3 Discussion

One of the biggest concerns when solving problems using supervised learning is determining which kind of data should be used for training and how to obtain them. A dataset must be recorded in this work because no such data for this problem are published. The dataset for this project is though small, but required much effort to obtain.

Data augmentation is essential, especially when working with a small dataset. Data augmentation helps to generate more variations based on existing data, helps the model to be more robust against transformations, and achieve better ability to generalize.

The term "consistency" defines some constraints to follow when applying a transformation in order to keep the augmented video as realistic as possible and not to tamper their information. Single-camera consistency requires the augmented video to contain a frame sequence that appears to be actually captured by a camera. Multi-camera consistency means the videos from all cameras must be in sync with each other in respect of time, position, and posture of objects.

Brightness can vary continuously or suddenly, and each camera can have a different brightness value depending on exposure and gain values of each camera. Therefore, changing brightness requires no consistency.

Horizontal flipping changes the posture and position of objects. Multi-camera consistency is needed because all cameras should perceive the same change so that the model can learn the relation between changes in all cameras.

Zoom helps to generate objects in different scales and requires single-camera consistency because the cameras are statically mounted, and it is not necessary to keep object sizes in sync across all cameras.

Reversing video can lead to changes in the meaning of the video, which means labels must also be re-mapped. Multi-camera consistency is required since the same sequence of movements must be perceived by all cameras.

Understanding different levels of consistency in data augmentation will help to gain maximum benefit. Blindly applying transformation could lead to low performance of the model.

4 Experiments and results

The datasets 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 datasets 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 dataset. Background, surrounding objects, and clothing were infrequently changed. Since it is vastly labor-intensive to record such a dataset 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.

4.1 Finding 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); temporal attention is neglected to let the action classifier learn to understand the video by reasoning action sequences.

4.1.1 Spatial attention

A soft attention mechanism [15] was inserted before the feature extraction model and responsible for blacking-out irrelevant parts of input to minimize their impacts on predic-

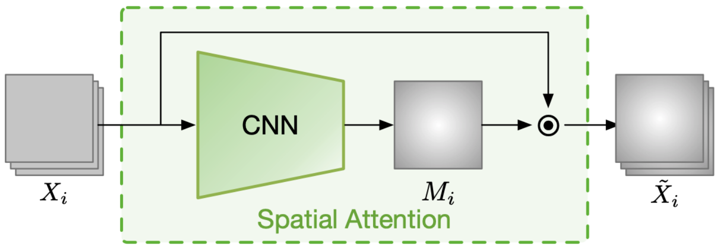


Figure 4.1: Spatial attention block [15]

tion. The spatial attention block consists of a convolutional network with '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 with ease. The input image will be feed 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.

4.1.2 Background subtraction as an attention mechanism

Background subtraction is a technique used to segment 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. The traditional methods, eg., Frame Differencing, give 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 are developed in the last few years. They are, however, too computationally expensive for being used in a real-time application due to their convolutional encoder-decoder architecture.

Fortunately, all cameras are stationarily mounted to the refrigerator in this specified use case. Thus, traditional algorithms could be applied to generate 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 [19]. Though the library was

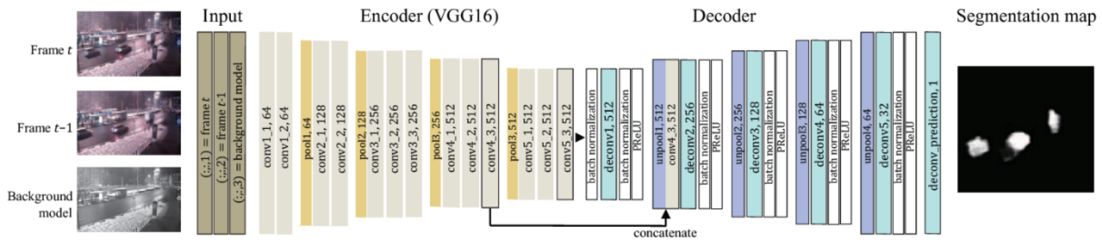


Figure 4.2: Encoder-decoder architecture for foreground detection [14]

written in C++, BGSLibrary can also be used in Python, Java, and MATLAB by means of its wrappers.

BGSLibrary supports different OpenCV [1] 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 in 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:

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

Frame Differencing, SuBSENSE [20], and Local Binary Pattern with Markov Random Field (LBP-MRF) [10] are the representative algorithm for group 1, 2, and 3, respectively..

This is a trade-off between speed and quality. Slow algorithms are not suitable due to the real-time capability requirement of this project. Fast algorithms produce low-quality masks with a high noise level, which makes the relevant features in images barely recognizable. To that end, LBP-MRF is selected on the grounds that its speed is just right to fulfill the real-time requirement, and its mask quality is more than acceptable. More precisely, the density of positive pixels in the area of the moving object is much

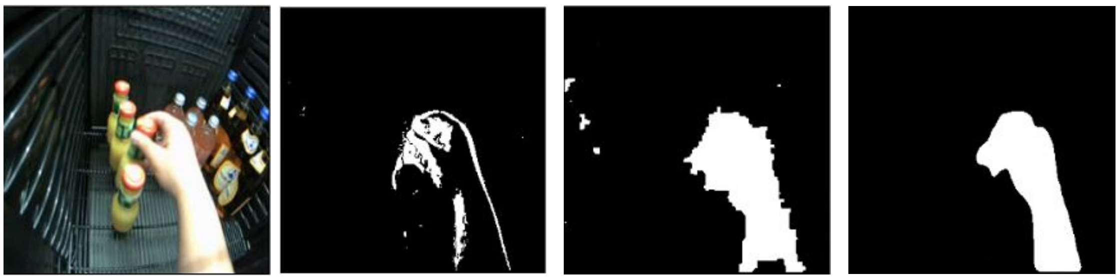

(a) Input image (b) Low quality (c) Average quality (d) High quality

Figure 4.3: Different mask qualities with a wide range of noise level and coverness

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 right in the next section.

4.2 Zooming to the region of interest

Input video frames will contain black regions after multiplication with their masks. The average percentage of area that contains moving object in the recorded dataset 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; 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 of the project.

4.2.1 Spatial Transformer

Spatial Transformer [9] 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 Spatial Transformer Network (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 $\theta =$

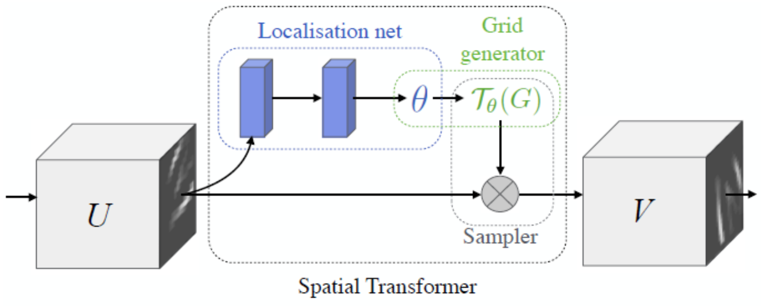


Figure 4.4: Spatial Transformer Network [9]

$\begin{bmatrix}\theta_{11} & \theta_{12} & \theta_{13}\\ \theta_{21} & \theta_{22} & \theta_{23}\end{bmatrix}$. 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.

The reason affine transformation is applied to a sampling grid instead of directly to the input image is differentiability. One problem is also known as forward-mapping and backward-mapping in image processing tasks. Affine transformation is a mapping process that doesn't necessarily need to be surjective or injective. Therefore, if transformation is applied directly to the input image, some pixels in the transformed image wouldn't even have been assigned a value. In addition, target coordinates could be fractional while they must be integers, a bilinear sampler is used to solve this by taking account of all pixels lying around those fractional coordinates. Since bilinear interpolation is fully differentiable, the original paper suggested bilinear sampler for this stage, but it can also be replaced by other samplers.

Only translation and scale are needed in this case, 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 did work as a plugin to many classification tasks on datasets such as MNIST [12], Street View House Numbers [16], German Traffic Signs [21], etc. However, the zooming effect of STN in this specific dataset is not remarkable and reliable. Although it did zoom into moving objects successfully, sometimes, input images were squeezed or translated until they disappeared.

4.2.2 Density-based cropping

To utilize the fact that there is nothing other than the moving object is visible in every frame, density-based cropping is introduced based on the assumption: Density of positive pixels in an area that contains the moving object is significantly higher than that in an area solely containing background.

Image thresholding

Image thresholding is the simplest method to generate binary images from grayscale images. Each pixel in the input image will be marked black if its intensity is less than a defined threshold.

Mean filtering

Mean filtering is a technique in image processing, commonly used for eliminating noise or smoothing images. Mean filtering uses a sliding window that replaces the center value with the mean of all other pixels (its neighbors) in the window.

Principle

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 thanks to its high pixel density, while noise areas will fade to gray. Image thresholding is then applied to remove noise pixels completely. The cropping procedure is trivial; 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 in this project is 96x96, the averaging kernel size is 5x5, and the threshold value is 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 eliminated; even one tiny noise in a corner that passed the thresholding stage could affect the result severely.

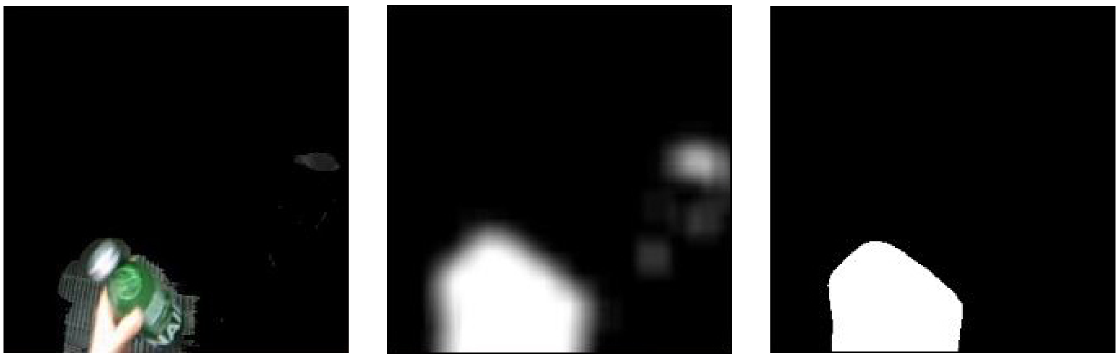

(a) Input image (b) After mean filtering (c) After thresholding

Figure 4.5: Illustration of how density-based cropping works

Unlike spatial transformer, although 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.

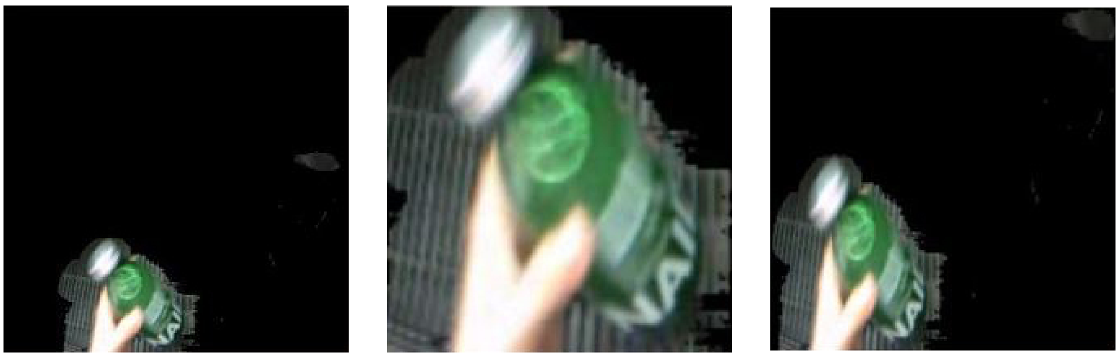

(a) Input image (b) Good result (c) Bad result

Figure 4.6: Success and failure of density-based cropping

4.3 Evaluation

4.3.1 Trade-off between speed and memory

It's common to see many implementations reading videos directly in *.webm. Sometimes, video frames were extracted into separate JPEG images in order to reduce loading time during training. Seeing that the amount of data in this project is relatively small, the videos were converted into *.npy to improve loading speed further The videos were first background-subtracted, resized to 96x96, and then converted to *.npy format to reduce loading time during training.

4.3.2 Cyclical learning rate

One can imagine the value of the loss function as a man finding a way down the hill, optimizer as a guide telling the man which direction to go, and learning rate as his step length. The learning rate determines to what extent new information affects the knowledge base of the mode. Too low learning rate leads to training inefficiency, i.e., the model will unnecessarily require many updates before it can reach the minimum. Or even worse, too high learning rate causes divergence behaviors of the loss function.

Annealing learning rate

If the learning rate is constant, it will be too large to converge at some point and will cause the value of loss function to fluctuate around the (local) minimum. Commonly, the learning rate is set relatively high at first to find the minimum area quickly. The learning rate will be decreased step by step during training to explore that area deeper.

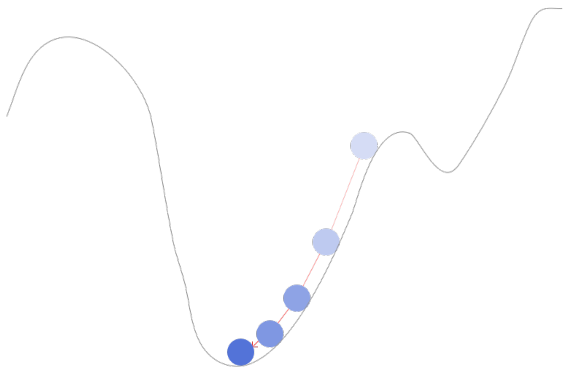


Figure 4.7: Decreasing learning rate helps to explore deeper

One should notice that minima are not necessarily global, and the model will be stuck at a random local minimum as long as it does not take any step out of that area, which is often the case.

Cyclical learning rate

The paper "Cyclical Learning Rates for Training Neural Networks" [18] discussed a new idea of cyclical learning rates as a variant of annealing learning rate that helps to eliminate the need of lots of experiments to find the best learning rate. The learning rate is set high at the initial stage, decreases from time to time, and the process is then reset (or gradually increased) after the learning rate reached a defined minimum value. Resetting the learning rate allows getting rid of the local minimum and find another if it is not robust.

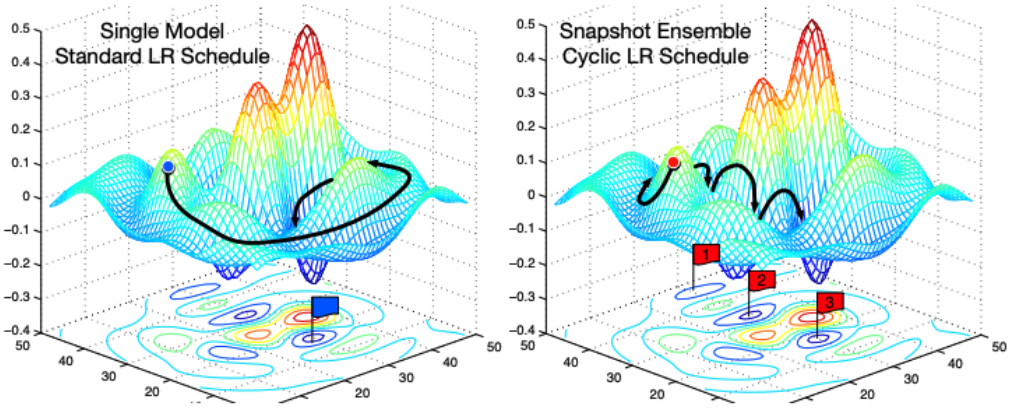


Figure 4.8: The model is converging to and escaping from several minima when applying cyclical learning rate [8]

This project implements a triangular learning rate policy, which is also introduced in the original paper – e.g., the learning rate increases and then decreases linearly.

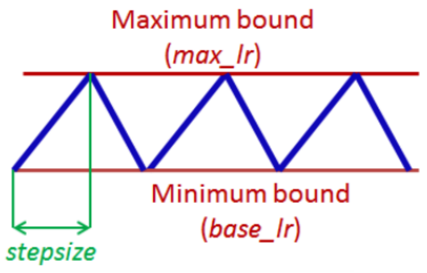


Figure 4.9: Triangular learning rate policy [18]

4.3.3 Learning rate range test

The paper "Cyclical Learning Rates for Training Neural Networks" also introduced the concept of the learning rate range test. The learning rate is set to a lower bound and gradually increased, and losses will be recorded after each increment. In this experiment, the learning rate grows from 10^{-7} to 0.1 exponentially.

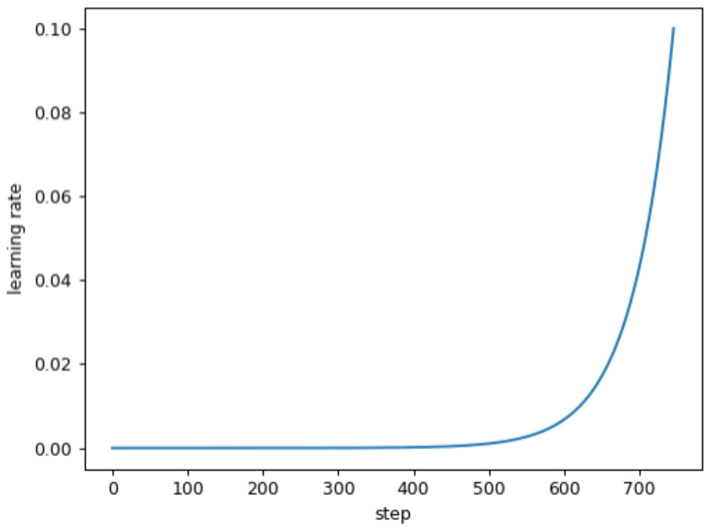


Figure 4.10: Learning rate increases exponentially

When entering the area of the optimal learning rate, a drastic drop in the loss can easily be observed. One can identify three different phases with different behaviors of the loss function in Figure 4.11. Loss value does not change much in the first phase because the learning rate is too low, then comes a steep decrease when entering the optimal area. Exiting the optimal area as the learning rate keeps increasing, it starts to fluctuate and slowly increase.

Good lower bound and upper bound according to the measured loss in this experiment are 1.6×10^{-5} and 5×10^{-4}, respectively.

4.3.4 Weight initialization

Choosing the right optimizer and learning rate is crucial. Weight initialization also plays the same important role while training a model. If parameters are not correctly

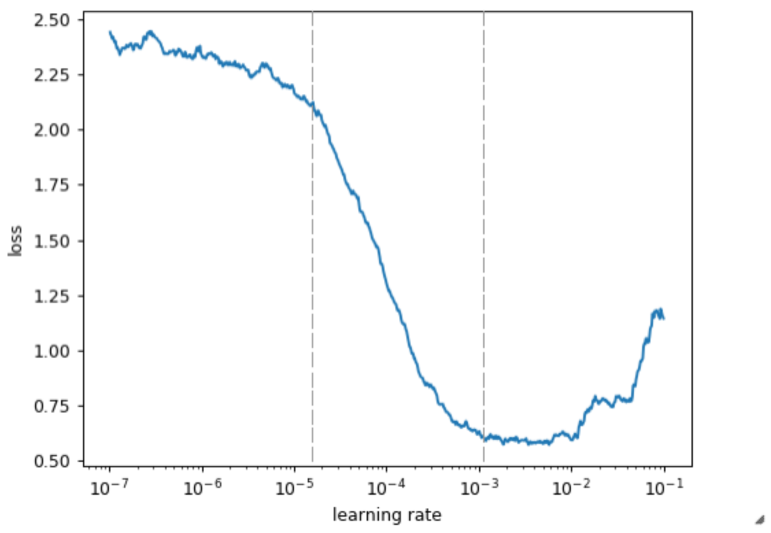


Figure 4.11: Behavior of loss function as the learning rate grows

initialized, vanishing/exploding gradient problems may occur, or even worse, converging to a minimum could even be impossible.

Xavier initialization

Xavier Glorot and Yoshua Bengio introduced a better concept of initializing random weights in the paper "Understanding the difficulty of training deep feedforward neural networks" [6]. Weights are initialized from a uniform distribution in interval $[-1, 1]$ and scaled by gain $\times \sqrt{\frac{6}{\text{fan\_in+fan\_out}}}$, where fan_in, fan_out are respectively the number of input and output units and *gain* is an optional scaling factor and depends on activation function applied on this layer. This project uses ReLU as activation function at all places, the recommended *gain* value for ReLU is $\sqrt{2}$.

4.3.5 Training results and model performance

The models are trained on a single GTX 1080Ti for maximal 50 epochs with SGD optimizer. If the model achieves better accuracy than the previous one, a checkpoint is saved. The early-stopping mechanism is applied in the training progress; the training is stopped if accuracy does not improve in 5 epochs. 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 (without classification layer) 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 come into existence due to hand and object motion. If this last block is frozen, the model can never 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 4.1: The result table of the action model

The number of features in this work is much smaller than that of the ImageNet dataset. 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 project. 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 smaller 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 worked well in the test, but it still can make false decisions, especially when unseen things appear (e.g., different sleeve colors) or when other objects are also to be seen in the foreground (bad background subtraction).

The refrigerator is equipped with an external display for showing recognition results. It is then placed at the event Solutions Hamburg 2019 and Sommerfest 2019 for letting people try. The models did also work even at the party, where many other light sources could have 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 dataset. However, more quantitative and quali-

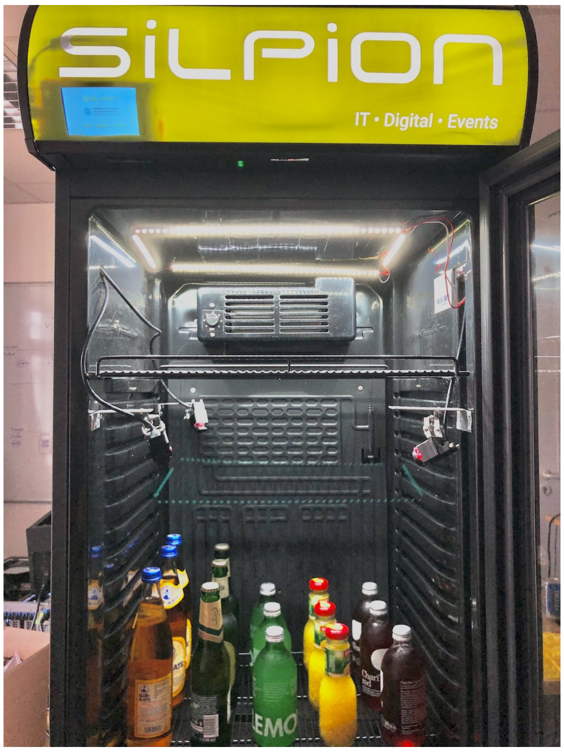


Figure 4.12: A photo of the beverage refrigerator

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

4.4 Discussion

Supervised learning tasks are data-consuming giants. Training a model in a supervised way requires a lot of data, and it is extremely labor-intensive to make a large dataset with a diversity of features. Nevertheless, it may be possible to minimize the amount of required data by analyzing project requirements, dataset, and then applying some techniques to reduce dimensionality. One can notice in this project that decisions strongly depend on the presence/absence of an object currently in motion. Attention mechanism will, therefore, bring remarkable contribution to reducing the amount of data if it can focus on the moving object.

Background subtraction is used in this project as an attention mechanism. The main advantage of using background subtraction is its simplicity with respect to computational cost and effort required for implementation since a public background subtraction framework is provided by Andrews Sobral. Local Binary Pattern with Markov Random Field is selected in this project, and the algorithm successfully helped to reduce the amount of required data and to increase accuracy. However, classification result thus strongly depends on background subtraction algorithm and on its foreground mask quality. Aside from that, dimensionality reduction also means information loss, which may have a negative impact on the result if project requirements, data, and important features were not carefully analyzed. There is a trade-off between speed and mask quality while choosing an algorithm. Choosing a well-suited algorithm to generate an adequate mask with an acceptable speed for a specific use case is, therefore, crucial and also not a trivial task.

Zooming into attended regions help to reduce waste of computation, and more importantly, to help convolutional layers to recognize features better. Spatial Transformer is designed to be plugged into any existed model with ease, consists of a localization network that learns to generate parameters for affine transformation and a sampler. Spatial Transformer worked well in many popular datasets such as MNIST and German Traffic Sign but struggled to work reliably in this project. The reason is very likely to be the lack of data for it to identify relevant clues for generating proper transformation.

Density-based cropping is introduced in this project, provides a simpler way to zoom into a region of interest on the assumption nothing but the object of interest is to be seen in the input image. This method can zoom correctly in most cases, and it fails when there is more noise in the input image than usual. In other words, density-based cropping relies strongly on noise eliminating, and choosing a good threshold for noise eliminating is not trivial.

Noise in foreground masks is formed in the background subtraction stage, which means there is a risk of the domino effect. Bad choice of background subtraction algorithm also leads to the inability of density-based cropping as well as adverse influence on classification results.

The proposed Multi-Fusion Network achieved good performance in this project despite a very small amount of training data, ~ 100 videos for each action, and ~ 1000 images for each class of object. However, more data from different domains is needed in order to evaluate its performance further.

5 Real-time recognition

Working with a training dataset and test dataset is easier because it is the only job the model is designed to complete. The model receives input, performs calculations, and returns corresponding output. However, a few more things need to be done to allow the model to work in real-life applications.

This section aims to make the refrigerator capable of working in real-time with continuous video streams from multiple cameras. To work in real-time and real-life scenarios, not only the pre-processing procedure and the models must be fast enough to process every frame from all cameras, but also there must be a mechanism being capable of sensing when an action begins and ends so that videos can be trimmed and forwarded to processing units.

5.1 Working with continuous video streams

In practice, videos are streamed from all cameras continuously. Hence the system will have no clue about when an action starts and ends. Therefore, it will be not capable of working properly without an additional mechanism since it is designed to receive video clips that contain a single action and an object.

Solutions discussion

Many researchers are attempting to detect those moments by using deep learning approaches. The most common and also state-of-the-art approach is to use a model parallel with the existing model that predicts whether an action is starting or ending in the current frame. However, the project boundaries may explode when applying this method, and extra computational costs could shut the real-time ability of the model down.

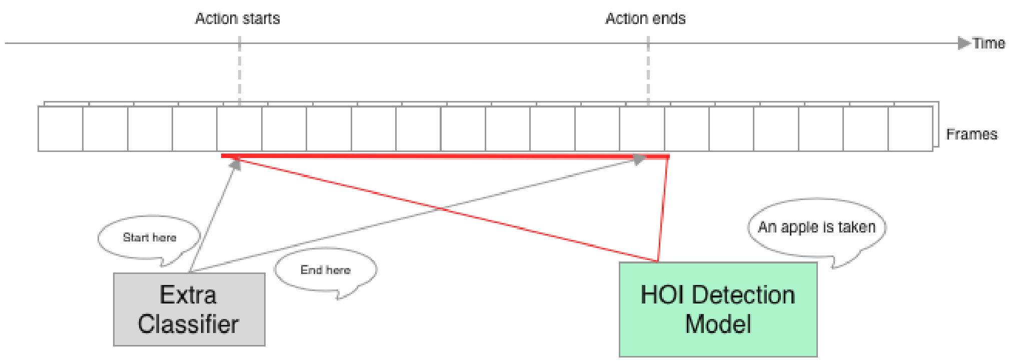


Figure 5.1: Illustration of the approach using extra classifier for detecting start and end of actions

Another approach is to use a fixed-size buffer to store video frames. The buffer will be used as input for the model, and the prediction will be performed after every n frames. This requires no additional model or complex mechanism. The contra of this approach is low reliability because inputs may contain one, two, or even three actions, or only parts of them, depending on buffer size. Actions can be overlooked by using this approach.

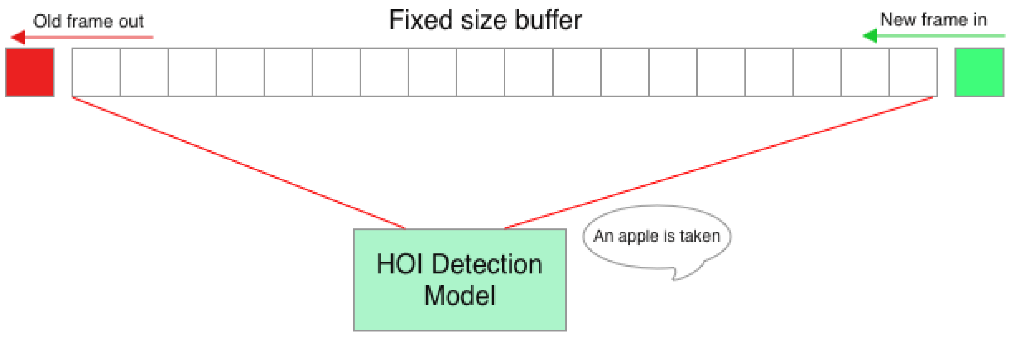


Figure 5.2: Illustration of the approach using a fixed size buffer for continuous detection

In addition, one may think a high confidence score can be used as a signal for good input and classification results. However, it is no longer the case in the modern deep neural networks, especially when working with too small datasets. Most neural networks suffer from overconfidence, meaning the confidence score is often too high and no longer reflects the true correctness likelihood. There are calibration techniques to solve the problem of overconfidence, such as Temperature scaling. However, this approach does not show its effectiveness when working with a tiny dataset.

The simplest but most effective approach

This project makes use of the fact that users must open the door before doing anything. An assumption is made that only at most one action is performed after opening the refrigerator's door for the sake of simplicity. An external reed switch is installed in the refrigerator, and a triggering magnet is installed in the door to detect the door state. The cameras only start recording when the door is open and stop after the door is closed.

This physical mechanism is simple but effective and well suited for the refrigerator. The model can now receive videos that contain only one single action without extra computational cost.

5.2 Real-time capability

As mentioned in chapter 3 and chapter 4, Multi-Fusion Network architecture is designed to work as fast by keeping the number of learnable parameters small. The choice of pre-processing algorithms, especially the background subtraction algorithm, also aims to achieve fast computation as well. The tiny spatial size of input frames (96x96) helps to significantly increase the model's speed up to hundreds of frames per second.

Utilizing time

Rather than starting to compute after the door is closed, the system computes right away after receiving frames from the cameras. Every frame is pre-processed and forwarded to the object classifier immediately to minimize the idle time of the GPU. After the door is closed, every frame in the video is already pre-processed and ready to be used for recognizing action, allowing the action recognition task to be completed in an instance of time. The whole process is fast enough for users to try the refrigerator and instantly receive the result.

Speed bottleneck

Unfortunately, the BGSLibrary does not support GPU at the moment. Hence the whole pre-processing step must be performed on the CPU. The deep learning model works entirely on GPU. Because the model works extremely fast with 96x96 inputs, the only bottleneck is in the pre-processing step. The negative effect of this bottleneck will become more significant when the number of cameras increases since it has to complete computation for many more frames within a second. Depending on accuracy requirements, one could keep reducing input size, choose another more lightweight background subtraction algorithm to work faster, or use another implementation that has GPU support. Input size of 96x96 and Local Binary Pattern with Markov Random Field are chosen in this project because only two cameras are actually used, and LBP-MRF can work fast enough on CPU to process from these two cameras in real-time.

5.3 Discussion

Action recognition models are designed to process video clips and assign a single label to each of them. Working fast is not enough for a model to be able to work in real-time because most of real-time action recognition tasks also involve dealing with continuous video streams, which is still a problem that many researchers attempted to solve at the moment. An additional mechanism is needed to determine when an action starts and ends in the stream.

Using another state-of-the-art deep learning model to solve the problem of continuous video streams may overstep the project boundaries as well as negatively affect the real-time capability of the entire system since much more extra work is required. Using a fixed-size buffer could solve the problem but with low reliability, and not every action fits in the buffer size. This project solves this problem by installing a reed switch to detect door state changes, so that input videos can be trimmed correctly without adding any extra computation.

Multi-fusion Network architecture is designed to work fast and be easy to scale. All learnable parameters in this architecture are shared, hence adding more cameras will only increase workload but not the size of the network.

However, the bottleneck in this project resides in the pre-processing step, namely the background subtraction. The implementation of background subtraction algorithms in this project does not have GPU support, such a heavy computation on CPU will significantly affect the overall performance. Local Binary Pattern with Markov Random Field is chosen in this project because it is still fast enough to work in real-time with two cameras. Another lightweight algorithm, such as Adaptive Background Learning, can be applied for increasing the computational speed with the cost of accuracy degradation.

6 Conclusion

6.1 Summary

The primary target of this work is to construct a deep learning solution for human-object interaction detection with multiple cameras and apply it on a refrigerator that tries to recognize human action and object that is the human is interacting with.

This project introduced an architecture called Multi-Fusion Network for understanding videos from multiple sources by linking observations from each video. Frame sampler is used to deal with varied video length and minimize redundant information. The temporal fusion block tries to understand movements from a single video and encode it into a uniform format. View fusion block combines outputs from the temporal fusion block and returns a final vector representing all observed movements. A standard softmax layer then uses this vector for action classification. Parallel to this process, all frames from the sampler are forwarded to a conventional object classifier to recognize the object. Outputs from the object classifier are combined into a final one by following a defined policy. Similar to the concept of divide-and-conquer and separation of concerns approach, Multi-Fusion Network breaks the problem into small ones, solves them, and tries to combine the results.

A dataset must be recorded to train and evaluate the model. Four cameras mounted in the refrigerator records thousands of videos. Although, the model struggled with the overfitting problem because the background (both inside and outside of the refrigerator) rarely changes. This problem is eliminated by using background subtraction as an attention mechanism and removing two cameras that are looking outwards to ensure that only actions being taken inside of the refrigerator are recorded. Besides, using background subtraction helps to significantly reduce the amount of required training data to achieve good performance, from thousands to merely hundreds of videos.

This project also introduced simple but effective density-based cropping that can zoom into the object of interest as an alternative to Spatial Transformer with the assumption that input frames contain nothing but the object of interest. Although, density-based cropping requires a well-configured threshold parameter to work as desired. Otherwise, density-based cropping will be negatively affected by noises in input frames.

All learnable parameters in Multi-Fusion Network are shared, which makes this architecture easy to scale up without sacrificing much computational speed. However, the pre-processing step causes a speed bottleneck in this project because the background subtraction algorithm runs on CPU since BGSLibrary does not have GPU support at the moment. The model in this project is still able to work in real-time without noticeable delay with an input size of 96x96 from two cameras.

In conclusion, applying deep learning to solving unpopular problems is not trivial because of the limitation of available data. Making a new dataset is extraordinarily labor-intensive and requires much effort. Data recorded just by a person or 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. Multi-Fusion Network reduces the amount of training data and therefore also reduces the required effort and the risk of overfitting. Using background subtraction as a strong attention mechanism can also help to reduce overfitting. However, which background subtraction algorithm to use 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. A lot of unexpected cases in real-life experiments are inevitable. Input standardization and noise removal are much more crucial in practice. This project 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 90.438% accuracy on the test dataset with only hundreds of training samples. The refrigerator was in the event Solutions Hamburg 2019 for testing under real-world conditions and did achieve comparable accuracy to the test result.

6.2 Future work

This project implements a small use case to demonstrate the effectiveness of Multi-Fusion Network as well as some approaches to overcome overfitting problems in practice, such as background subtraction as attention mechanism, density-based cropping, and inactivity trimming. Due to the limited time and resources of this project, only a small dataset could be recorded. A larger dataset is required for proper evaluation, or Multi-Fusion Network must be implemented for a different use case that already has a large public dataset so that one can see how well it performs in comparison with others. Another potential of Multi-Fusion Network could be indoor activity recognition for gaming experiences, smart home applications, or augmented reality devices. This architecture could be used in embedded systems with average computational capability or small toolkits such as Google Coral or NVIDIA Jetson, thanks to its lightweight.

Bibliography

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Georgia Gkioxari, Ross Girshick, Piotr Dollár, and Kaiming He. Detecting and recognizing human-object interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8359–8367, 2018.

[6] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *ArXiv*, abs/1704.00109, 2017.

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[10] Csaba Kertész and Vincit Oy. Texture-based foreground detection. 2011.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[13] Yong-Lu Li, Liang Xu, Xijie Huang, Xinpeng Liu, Ze Ma, Mingyang Chen, Shiyi Wang, Hao-Shu Fang, and Cewu Lu. Hake: Human activity knowledge engine. *arXiv preprint arXiv:1904.06539*, 2019.

[14] Kyungsun Lim, 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)*, 2017.

[15] Lili Meng, Bo Zhao, Bo Chang, Gao Huang, Wei Sun, Frederich Tung, and Leonid Sigal. Interpretable spatio-temporal attention for video action recognition, 2018.

[16] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] Leslie N. Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2015.

[19] Andrews Sobral. BGSLibrary: An opencv c++ background subtraction library. In *IX Workshop de Visão Computacional (WVC'2013)*, Rio de Janeiro, Brazil, Jun 2013.

[20] Pierre-Luc St-Charles, Guillaume-Alexandre Bilodeau, and Robert Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, 2014.

[21] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. pages 1453 – 1460, 09 2011.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[23] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[24] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.

A Technical specifications

Model: Dell Precision T3600 Workstation

Processor: Intel® Xeon(R) CPU E5-1620 0 @ 3.60GHz × 8

Graphics: EVGA GeForce GTX 1080 Ti/PCIe/SSE2

Memory: 32 GiB

Disk: SSD 256 GiB

Power supply: 650W

OS: Ubuntu 16.04 LTS

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name:	______________________
Vorname:	______________________

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Application of Multi-Fusion Network for Human-Object Interaction Detection

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

______________	______________	______________________
Ort	Datum	Unterschrift im Original