Universty of Tartu

Faculty of Science and Technology

Institute of Technology

Mateus Surrage Reis

Smart Traffic Control Using Optimised Convolutional Neural Network

Bachelors thesis (12 ECTS)
Science and Technology Curriculum - Robotics and Bioengineering

Supervisor:
Prof. Gholamreza Anbarjafari

Tartu 2019

Tark liikluse juhtimine rakendades optimeeritud konvuleerivat närvivõrku

Lühikokkuvõte:

Uusimad objekti tuvastus meetodid kasutavad oma töös konvuleerivaid närvivõrke, mis arvutuslikust küljest on ressursiahned. Täpsete tuvastusmudelite jooksutamine manussüsteemides vajab palju optimeerimist, eriti kui seade peab toimima reaalajas. Käesolev töö kirjeldab targa ülekäiguraja teemärgi loomist: nutiseade, mis on mõeldud liikluse juhtimiseks. Seadistades iga SPC posti eraldi närvivõrku, oli võimalik märkimisväärselt tõsta kiire ja ebatäpse närvivõrgu selgust. See saavutati kasutades ära kolmest kaamerast tulevate sisendpiltide minimaalset varieeruvust. Algoritmi täpsust parandati 80 klassilise üldnärvivõrgu 33.1% mAP pealt 60.7% mAP peale, rakendades ainult liiklusega seotud pilte koos seitsme erineva teemakohase klassiga.

CERCS: T111 Pilditehnika, T170 Elektroonika, T120 Ssteemitehnoloogia, arvutitehnoloogia

Keywords: CNN, tehisnärvivõrk, nutistu, autonoomne, tehisintellekt

Smart Traffic Control Using Optimised Convolutional Neural Network

Abstract:

The state-of-the-art in image object detection is in convolutional neural networks, which is a computationally expensive base to build on. To run accurate detection in an embedded device, additional optimization is required if there is a need to run it real-time on each frame of a video. This thesis details work done in the development of a smart pedestrian crosswalk: an Internet of Things enabled embedded platform for traffic control. By fine-tuning an individual neural network for each SPC post, it was possible to significantly boost accuracy in a fast, low-accuracy CNN. This was accomplished by taking advantage of the low variation in possible input images, being drawn from only 3 cameras per post. The improvement was from 33.1% mAP in general context images and 80 classes to 60.7% mAP on solely traffic images and seven traffic-relevant classes.

CERCS: T111 Imaging, image processing; T170 Electronics; T120 Systems engineering, computer technology

Keywords: CNN, neural networks, IoT, autonomous, AI

Table of Contents

List of Figures

Abbreviations and Acronyms

CNN - Convolutional Neural Networks

SPC - Smart Pedestrian Crosswalk

AP - Average Precision

mAP - Mean Average Precision

IoU - Intersection over Union

GST - GStreamer

FP - False Positive

FN - False Negative

TP - True Positive

YOLO - You Only Look Once (CNN architecture name)

FC - Fully Connected (CNN layer)

IoT - Internet of Things

1 Introduction

In this thesis, one particular task is considered and worked on: the use of an object detection convolutional neural network for traffic control, in the context of development of a Smart Pedestrian Crosswalk (SPC) IoT platform, as shown in figure 1.1. The SPC is a smart traffic sign, which can be used anywhere, where increased pedestrian safety and data input are required. SPC uses AI, V2X, sensor fusion and more than 30 different features to adapt to different traffic situations and weather conditions. The state-of-the-art device is useful in many different example situations, some of which are presented below. The platform has limited processing power and convolutional neural networks (CNN) have heavy computational requirements, and as such one of the primary concerns was to optimize and automate the process in such a way to make object detection viable for use in these embedded computers, as well as make inroads in the more complex interconnected process of making use of that object detection information to make possible object tracking and prediction.

The creation of such an embedded device is motivated by the growing push and popularity of

Figure 1.1: The SPC: Smart Pedestrian Crosswalk.

smart devices and connected in traffic, driven by self-driving cars and the recent surge in artificial intelligence and neural network advances [1–3]. It's only in the past five years that the use of a computer capable of this kind of camera-based object detection in the field has become financially and practically viable.

The amount and variety of modelling and processing that can be done by such a device is too

wide to fully cover all possibilities, and Bercman's SPC is yet a nascent project with a year of development invested in it. Therefore, in this thesis the focus will be on the optimization method used to allow an acceptable balance of accuracy and speed in this commercial product, approaches that were considered, relevant background information on the related fields, as well as possible improvements on the process that were unable to be realized for this particular work.

The SPC itself is an embedded computer inside a pedestrian crossing walk sign equipped with cameras and other sensors. The final goal is for each SPC unit to be a mostly autonomous deciding agent, capable of sensing the traffic environment around itself and, in the case of a situation warranting it, reacting in some sense — by communicating with other smart devices, flashing LEDs and playing warning sounds.

2 Theoretical Background

2.1 Object Detection

Computer vision is a field with a long history; one famous early effort happened in 1966, when a professor assigned a group of students to solve object recognition over a single summer. We now know the problem is not that simple. It's only in the last decade that progress to the point of practical use of generalized detection in consumer applications has been made in the field. For more specialized applications, such as cell counting [4, 5] and industrial processes [6, 7], computer-vision based usage is already established practice.

As the name states, the objective of an object detector is to detect and classify objects. It takes in an image as input and outputs a set of bounding boxes as well as a classification label for each object (generally, a probability, rather than a definite answer). Other notable computer vision categories include:

- Object segmentation, where rather than bounding boxes, the goal is to find each pixel in the objects.
- Image classification, where the image as a whole is classified. Not suitable to be used with complex images containing multiple objects or situations.

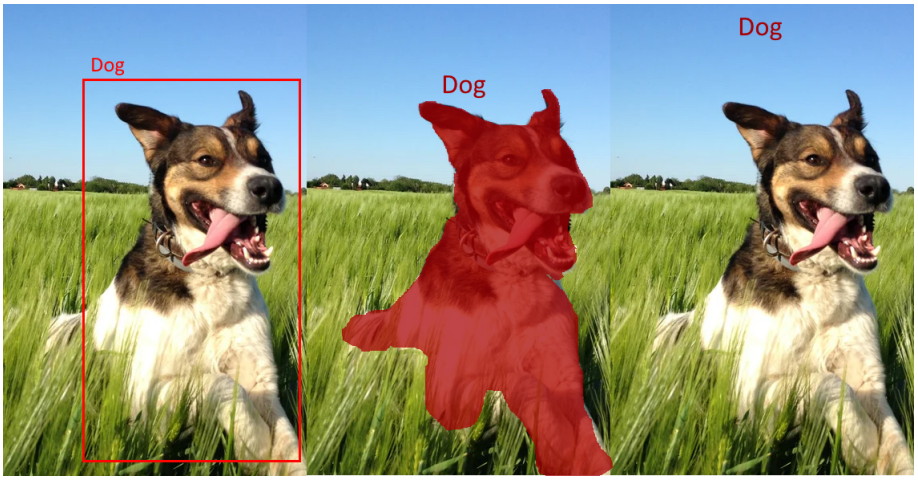

Figure 2.1: From left to right: Object detection, object segmentation, and image classification.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a type of neural network which takes advantage of spatial patterns in its input data, as opposed to, e.g., a multi-layer perceptron, where every neuron in each layer is connected to every neuron in the next. In CNNs a single value (pixel) in an output feature map is modified by a small 'square' of surrounding pixels from the previous layer, in an image convolution operation, from which the network takes its name from. The convolutional 'filters' are the neural network weights, modified by the training process. Once a filter has 'learned' a certain shape, it's able to find it anywhere in the image.

For the past decade, CNNs have consistently outperformed prior methods such as SVMs for object detection and other computer vision tasks. In 2012, AlexNet [8] won the ImageNet Large Scale Visual Recognition Challenge, boasting a more than 10% accuracy lead on the next place in the rankings at the time. This leap was made, interestingly, not through an advance in theory or novel algorithm, but by the use of Graphics Processing Unit (GPU) to make previously infeasibly computationally expensive networks practical.

The basic operation of CNNs, like other neural networks, is straightforward: it consist of multiple layers of operations made on an input image. In training, upon going through the network, the obtained output is compared to a desired output, and adjustable weights on each neuron are changed based on that comparison. Here are presented some examples of common layers in a CNN.

2.2.1 Convolutional Layer

The basic type of layer that defines the network. Takes in an image and runs a number of filters: conceptually, it can be understood that each filter roughly learns a particular shape. Filters in layers early on in the network learn very basic elements such as edges, while later filters combine those basic shapes into increasingly complex combinations — objects. The output "images" created by running a filter over an image are commonly called feature maps.

2.2.2 Pooling Layer

The goal of pooling layers is to reduce dimensions on the feature maps, a method of downsampling. A 2x2 max pooling block, for instance, outputs only the maximum value in each mutually exclusive 2x2 pixel block, cutting the width and height of the input image by half. Downsampling feature maps in this way increases robustness against size and position of features in the image.

2.2.3 Residual Block

The residual block is a recent development, from Kaiming He et al.'s ResNet [10], but not a complex one to understand. In a basic level, it is a piece of the network where the data may skips two or three layers. Although deep neural networks are effective, results do not simply grow linearly (or even uniformly) better the deeper a network is. This is known as the degradation problem, and residual block are an effort to ameliorate one known cause of the degradation problem — the fact that deep networks are sometimes unable to approximate simpler functions, such as an identity function.

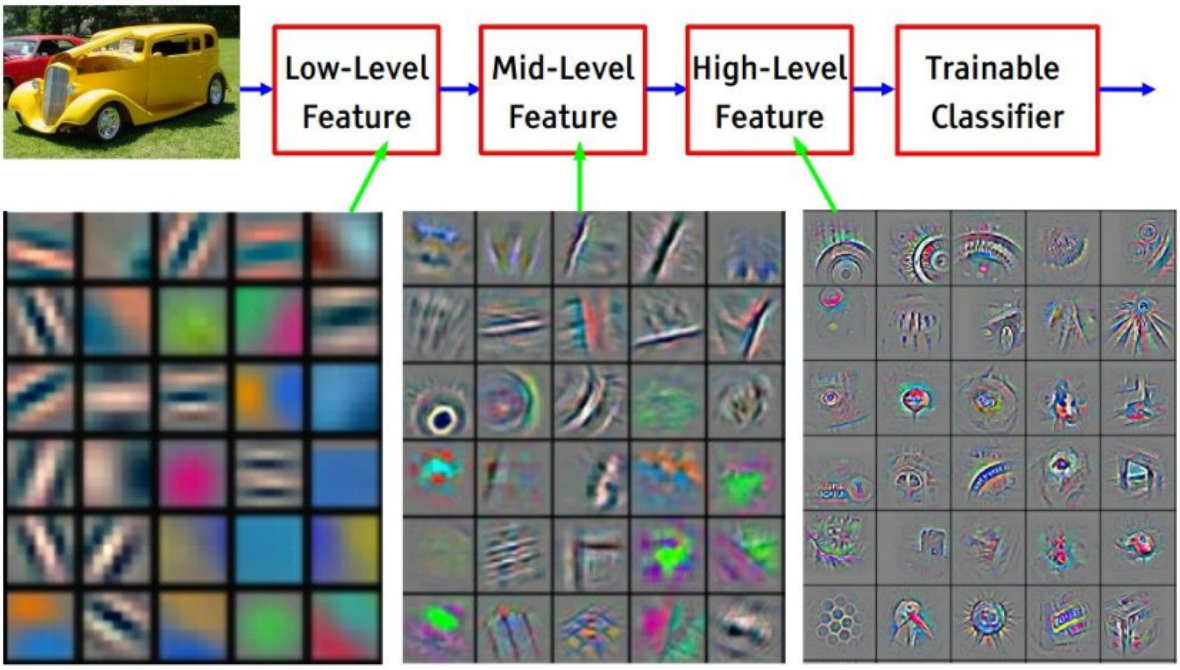

Figure 2.2: Visualizations of filters in a CNN: The filters on the middle block are formed by weighted additions of filters to the left, and those on the right block by weighted additions of filters on the middle. [9]

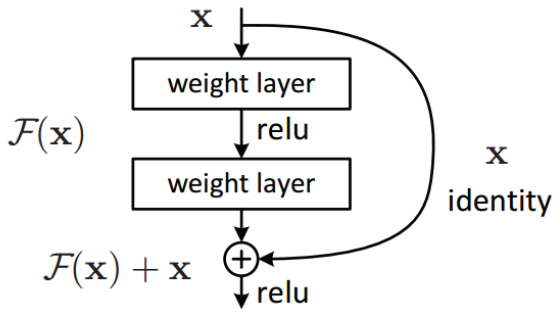

Figure 2.3: Basic layout of a residual block.

In other words: in some cases, a shallower network performs better than a deeper one, and residual blocks give deeper networks a way to 'mimic' a shallower network by skipping layers in some capacity.

2.2.4 Fully Connected Layer

Fully connected(FC) layers were present in the earliest CNNs: Yann LeCun et al.'s LeNet [11], in 1998, consisted of 2 interleaved convolution/subsampling layers, followed by 2 FC layers. In such a layer, there is no spatial information to be used: every neuron from the previous layer is connected to every layer to the next. For this reason, they're generally used at the end of CNNs, once the size of the feature maps has been reduced to a computationally manageable level.

FC layers are used at the end of networks to aggregate the information obtained in the previous convolutional layers into a discrete output such as a classification label.

2.3 Accuracy Measure

A commonly used accuracy metric for object detection is mAP: mean Average Precision. To reach a mAP, first a full recall-precision curve must be created. Let us consider, in the context of predictions:

$$TP = True\ Positive, \quad FP = False\ Positive, \quad FN = False\ Negative$$
$$Precision = \frac{TP}{FP + TP}, \quad Recall = \frac{TP}{TP + FN} \tag{2.1}$$

In a precision-recall curve, each data point is created by setting a certain certainty threshold and retrieving the precision and recall values for that threshold. There is a trade-off between precision and recall; for example, with a very high threshold of 0.95, the number of false positives would logically be much smaller than a lower threshold. On the other hand, any object with a confidence score below 0.95 would be classified as negative, leading to many false negatives and a high recall rate.

The average precision(AP) score is then the area under this curve — as precision and recall both have possible values going from 0 to 1, the square described by them also has a possible area between 0 and 1, usually expressed in percentage. Mean average precision is the mean AP taken over all classifications.

Since object detection involves bounding boxes rather than simple classification, there is a need for a parameter that evaluates whether a certain detection is 'true' or 'false'. This parameter is the intersection over union (IoU). It compares the label bounding box with detected bounding box, and the fraction of the intersection area over the union of the two boxes; an exact perfect fit of the two boxes would have the area of the intersection equal to the area of the union, giving an IoU of 1.

2.4 YOLO Network Architecture

The base network architecture used for object detection in this project was Joseph Redmon's YOLOv3. This architecture achieves a good ratio of speed and accuracy, suitable for traffic applications, by using a single pass to detect bounding boxes and objects (see figure 3.1). The traditional approach to object detection and localization requires multiple networks or multiple passes for those separate tasks. YOLOv3 associates bounding box positions and sizes during training, which makes it particularly suitable for an application involving a static camera and cars running along fixed lanes. There is a variant of the network, called tiny YOLO, which trades a great deal of accuracy for speed by heavily cutting down on the number of layers.

This network architecture divides the image into a $S \times S$ grid of squares, where each square can predict B bounding boxes, and there are C possible object classifications. Thus, the network output tensor has size:

$$S \times S \times B \times (1 + 4 + C) \tag{2.2}$$

Where the "1" is the box confidence score — that is, how confident the network is that there is an object inside the box, and the "4" are the bounding box offsets. In addition, YOLOv3 has multiple improvements compared to its first iteration, such as:

- Detection across multiple scales using the concept of pyramid feature networks. Candidate bounding boxes are predicted from those three scales, followed by feature extraction.
- Multi-label classification. Unlike in earlier versions, each detection contains C output probabilities, rather than the mutually exclusive classification.
- The fully connected layers from earlier versions are replaced by anchor boxes, making class probability detection take place at the bounding box level rather than the grid cell level.
- K-means clustering on the training set bounding boxes, to find the best matching anchor boxes.
- Use of state-of-the-art residual blocks and skip connections.

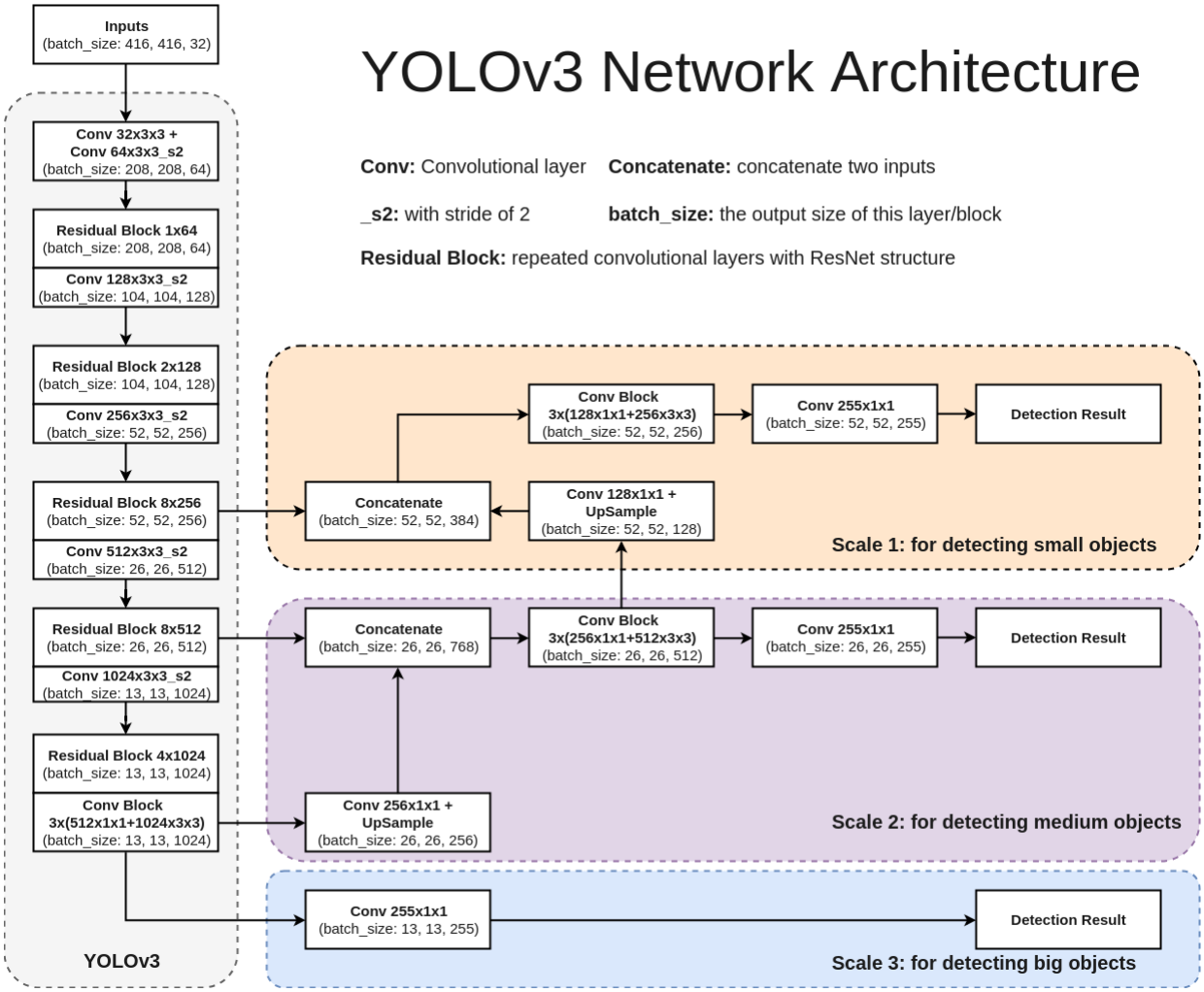

Figure 2.4: Large YOLO variant network architecture. [12]

3 Application

3.1 Motivation

Automotive traffic is a complex system, vital to our cities, and it has deadly components, with road accidents being one of the leading causes of accidental death globally. Consequently, regulating this system in as safe and reliable a manner possible is an ongoing concern and is likely to remain one for as long as automobiles continue to be a predominant form of transportation.

Using automation for the task, then, is a natural idea, since human oversight is very limited when dealing with the kind of scale and complexity that a road transport grid entails. This can be seen, as a simpler example, in the overwhelming worldwide spread and success of tools like automated speed cameras and traffic lights. One more recent facet of this phenomenon is the idea of steadily increasing automation, with connected devices interacting with, managing, monitoring traffic [13, 14]. In its most extreme form, this idea might take the form, for instance, of fully automated self-driving cars, receiving data from sensors placed abundantly near roads. Any dangerous abnormality would be broadcast locally through the network and instant measures taken, without human intervention in between.

The advantages of such an approach are largely in control and safety: faster response times to accidents, better warnings, higher uptime, coverage. Consider what could be done in a hypothetical future city where the asphalt itself passively senses and identifies the automobiles above them, overlaying every car onto a map of the streets in real time, accessible by emergency service operators: Car theft would cease to exist within city limits, ambulances would be able to expediently get anywhere, traffic law violations would instantly get logged, accidents minor or major would be instantly spotted and could be analyzed for blame, and so on.

Such 'smart' automation is yet nascent and a full-scale smart grid in those terms would be a monumental engineering undertaking.

3.2 SPC Information

The SPC project represents another attempt at this concept, albeit one currently less complex than wide-ranging, self-driving-vehicle connected ones. The physical unit itself consists of a network-capable computer inside a pedestrian crossing sign, equipped with three cameras, a radar unit, lidar sensors, LED strips framing the crosswalk sign and loudspeakers. There is versatility in what can be achieved with such a kit and development is ongoing, but for the purposes of this thesis, the SPC is meant to serve as a local warning, warning people nearby and possibly authorities when irregular traffic activity is detected, lighting, as well as eventually serving as a point in a network of smart crosswalk signs.

For detection of relevant objects, the approach chosen is CNN based, with the camera images serving as input. The main challenge was to improve computational performance to a degree that the object detection CNN ran at an acceptable speed on the NVIDIA Jetson TX2, a low-powered embedded computer when compared to the powerful machines that tend to be used with CNNs.

The SPC is equipped with NVIDIA JetPack: a comprehensive software development kit for building AI applications. It consists of Ubuntu 16.04 packaged with libraries and frameworks useful for computer vision and neural network usage: CUDA, cuDNN, VisionWorks; as well as specialized software to interface with Jetson hardware. Using a normal Ubuntu Linux installation makes it easy to work with, needing no specialized infrastructure or exclusive frameworks. The use of a conventional linux operating system as a base simplifies the entire process.

The software video pipeline is based on the open source multimedia framework GStreamer(GST), capable of being used in complex configurations as a C/C++ programming library or for simpler tasks such as video recording, remuxing, conversions and so on. NVIDIA provides plugins for the framework, many specialized and optimized for Jetson hardware, as well as other packaged computer vision and artificial intelligence libraries such as cuDNN, VisionWorks and OpenCV.

3.3 Problem Description

As mentioned in 2.4, the goal is to automate traffic control — and to do that in detail, it's necessary to sense traffic. In fullness, the SPC will be able to sense, make decisions, interact and communicate, but in this thesis the focus is on sensing and detection, and in particular, image detection using CNNs. A relatively high processing speed is required to maintain minimal delay and good accuracy between events happening and being detected and registered by the SPC. Darknet was the software platform used, with the base, unchanged, unchanged, slower variant of YOLOv3 resulting in five frames per second(FPS), while the fast variant ran at roughly 25 FPS — the former was deemed too slow, and the latter judged not to be accurate enough, since large gaps between detections of a moving object on frame would interfere with any tracking algorithm.

The issue then becomes one of optimization: either to speed up the slow network while maintaining accuracy, or increase accuracy in the fast network, without losing speed. To this end, two modifications to YOLO were briefly considered:

3.3.1 Recurrent Network

Modifying the CNN to be recurrent in some way is a natural fit for videos: YOLO is a single-shot detector, meaning it identifies and detects objects per image independently. With video, and in specific considering traffic video, objects generally move in predictable patterns. Feeding the bounding-box outputs of the last frame into the next, apart from the usual pixel data, would carry a great weight in information, and could have improved the small YOLO variant significantly in tracking performance without sacrificing speed.

The major disadvantage of this approach would have been high complexity: it would have necessitated direct merging with the tracking algorithm, as well as a major overhaul of the existing CNN infrastructure.

3.3.2 Staggered Detection

Another possible approach would have been to use the more accurate network, but not run it every single frame: every number of frames, the heavier network would be run, and the frames in between would be interpolated in some way. With the assumption that people and cars in a street don't generally change directions or velocity that often, this might have been a reasonable compromise — except, the points where cars and people change directions abruptly are precisely the most important ones to capture, prompting the abandonment of this idea.

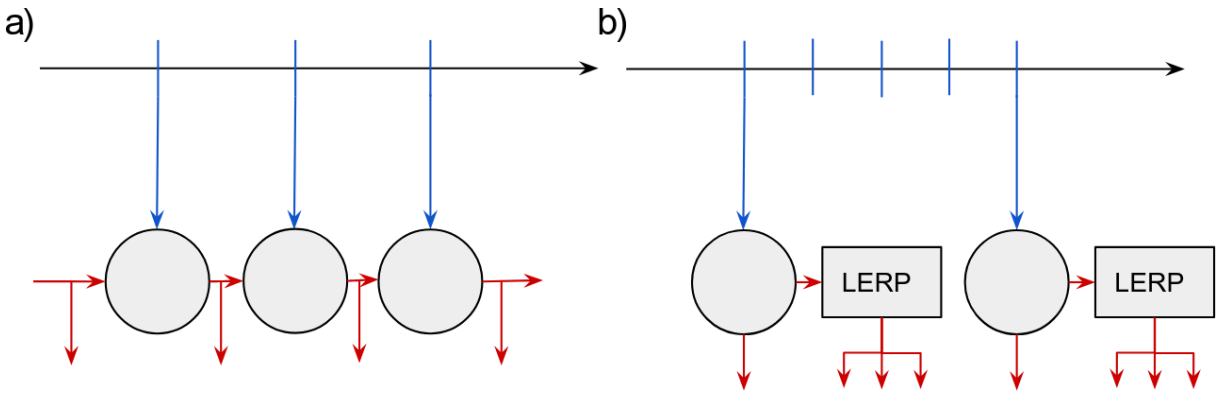

Figure 3.1: a) Recurrent Network b) Staggered Detection.The black horizontal arrow represents the video timeline, blue lines represent frame data, red arrows represent bounding box output, circles represent the network detection process itself, and the rectangles represent a non-neural network prediction algorithm. Note that in each case, the prediction mechanism would take in bounding box data from more than just the prior network run; this was omitted from the diagram for clarity.

4 Methodology

4.1 Summary

The final chosen method was simpler than those previously proposed, and the basis of it is in leveraging the reduction in problem space that comes from having a limited number of camera angles and contexts. With only three cameras, and knowing they must detect the same kinds of objects from the same positions, it is possible to strategically overfit the network during training, gaining consistent accuracy in exchange for losing generalization in the model, as when something unpredictable happens.

By coupling this with an automated labeling system that runs off-site and ahead of time on a more powerful computer, it is possible to have a small, fast YOLOv3-tiny running real-time on Jetson TX2 computers, with a balance between speed and accuracy, making a significant improvement on using either of the stand-alone YOLO variants by themselves.

In a more practical sense, the way to make use of this "strategic overfitting" is to have one set of network weights for each post; each group of cameras attached to one post will have its own footage used for training of an individualized CNN. With inputs so limited limited in scope to 3 camera angles, this strategy becomes viable.

4.2 Automated labeling

The process of automatic labeling begins by capturing a large amount of frames — in testing, 100.000 were used per training run, from one hour of footage. The resolution accepted by the network is 608×608, and it crops and resizes mismatched input sizes down to that square size, making a square aspect ratio ideal for capture.

After capture, the footage is extracted into static images and sent to a far more powerful computer: they are run through the more general-purpose, non-finetuned large YOLO, extracting labels that are saved in training format for tiny-YOLO. This method of automatic labeling is not perfect — even the higher-accuracy YOLO variant is not perfect, after all, clocking in at 57.9% mAP at 0.5 IoU threshold — but given that automobiles and pedestrians on a street are a consistent subject with few complicating factors, it was sufficient.

Automated labeling is not *required* for this method to work, and in fact, manual labeling of the acquired frames would result in higher accuracy, given that human labor is still superior to neural networks in this sort of vision task. However, SPC is meant to be a commercial product, with tens of post units, each with their own trained network. This would result in many man-hours of work labeling images for questionable benefit. That said, it may be a worthwhile modification,

even requiring manual labeling — the final accuracy difference between manual labeling and automated labeling was not tested.

4.3 Fine-tuning

Prior to fine-tuning, tiny-YOLO was pre-trained on a subset of the Google OpenImages dataset comprising the 80 classifications used by COCO dataset. Meaning, if a certain image did not contain any COCO objects, it was excluded from training. This produced roughly 800.000 images used for training. Data augmentation was used for all training, with changes in exposure, saturation, hue and random cropping (jitter). Such training results in a network capable of detecting objects in general images, but as mentioned, accuracy suffers when it comes to a specialized application.

After the pre-training, the network was fine-tuned on the extracted footage images and YOLOv3-created labels. Constraining the problem space is known to make it easier to get good results not only on neural networks, but machine learning methods in general — the opposite effect to the curse of dimensionality. This effect is typified, for example, with digit classification on the MNIST numbers dataset. The input data is a mere 28×28 pixels of grayscale (8bit) and there are only 9 output classes; the record accuracy for that task is 99.79%, and it is not difficult to achieve above 99%, even with much less elaborate models than a full neural network with multiple hidden layers. MNIST is an extreme example, but the principle applies.

In this case of static camera footage, however, such constraining of the problem space is much 'softer' than MNIST: there is no hard limiting of input resolution, output size, or discarding color channels. Instead, the network itself compresses the output space just by having a limited variety in input. If the goal were a network capable of detecting a wide variety of objects in a common context, this would lead to overfitting.

The same process of fine-tuning can also be used to change the possible objects that can be detected: by keeping only the earlier convolutional layers of the network and re-training the later ones — which, as detailed in 2.3.1., contain the more complex object filters — in essence, rearranging the combinations of more primitive filters to detect new objects. This practice is called *transfer learning*.

There is still a danger of overfitting even in this fine-tuning scenario if a certain condition fails to be accounted for. For instance, if all training data was taken in the summer and the post continued to use the trained network into winter, it would undoubtedly lower in accuracy — due to this season issue in particular, it's currently being considered that this training would have to be redone every few months, or otherwise a large dataset spanning the year would be needed. As there's not been enough time to test this, it remains speculative.

4.4 Tracking

As mentioned, YOLO is a single-shot detector, meaning that for there to be continuity between frames, a tracking algorithm must be used. The tracking algorithm chosen is SORT as described by Samuel Murray [15], using the paradigm of tracking-by-detection. In this paradigm, each frame's worth of tracking happens in three steps: Detection, prediction, then association.

First, objects are detected in the frame. Second, based on prior detections, a prediction is made on where each object will be. Finally, a comparison is made between the detected and predicted objects, which are thus associated based on similarity.

5 Results & Discussion

Using the outlined method, the speed at which it ran remained in line with the expected YOLOv3-tiny performance, running at 25 to 30 frames per second on Jetson TX2 hardware, and the addition of a tracking algorithm only had a negligible impact on speed. The mAP, however, shot up to 60.7% up from YOLOv3-tiny's reference mAP value of 33.1%. This value was measured considering only relevant classes in detection: bus, truck, car, bike, motorbike, person; as opposed to the reference values' full 80 classes in common contexts.

There was a limitation in amount of available data: With another one hour stretch of video, half of the video was used for training, the other half for mAP measurement, and the evaluation was made against the YOLO-created labels. This means it's unrepresentative of a final product where many different recordings made in various points in the day were used for both training and evaluation; one single hour of a day is bound to have many more similarities between frames. It's a similar issue to using training data in measurement: if training and evaluation data are too similar, the power of generalization of the model ends up being insufficiently tested.

Even if the real value is ten or fifteen points lower than the measured 60.7%, however, it's a large improvement and sufficient for usage in the tracking algorithm, just by retraining on a specialized dataset. The major advantage is how much of the training process can be nearly fully automated when using YOLO labeling.

In a way, the method is a way of directly translating high accuracy in general object detection CNNs directly into speed for a more specialized problem set — in our case, one possible, easy-to-implement improvement is to use another network rather than large YOLOv3 for the task of labeling YOLOv3-tiny training data, as the labeling happens off-site with no time pressure. YOLOv3 is made to be fast, and there are other network architectures that sacrifice that speed for higher accuracy. Such chains of networks, using ones output as the label for another, hold promise as powerful tools in the field of artificial intelligence, building automation on top of automation.

Another possible improvement would be in the process of data collection. Since YOLO is a single-shot detector, it's not an advantage to have 30 frames per second of footage: frames that are near each other in footage are similar enough that just going through the data augmentation process renders them roughly equivalent as training data. Instead, capturing two to five still images per second would reduce redundancy and data space burdens in data collection, in exchange for taking a longer amount of time to record an equivalent number of frames.

A third improvement would be making a middle ground network between the large networks which are general for any image and the small network running on the posts themselves. By using large amounts of labeled relevant, but not specific data in the training of such a middle

ground network, it could achieve a similar effect. In the example of this thesis, this would mean fine-tuning or training another large network with exclusively images of traffic, further improving the labeling of tiny-YOLO training data. Although leveraging the narrow application space involved in using static camera data to sense traffic information allows for significant accuracy optimizations, there is much space for cumulative improvements in the area of practical object detection.

In figure 5.1 a number of example detections made by the large YOLO network can be seen; note that since it's the general object detection network, non-traffic related classes can still appear, although during the fine-tuning process, data imbalance — that is to say, the rarity of such objects — makes it so they do not appear in the final fine-tuned network reliably.

Figure 5.1: Detection and tracking examples, including low light conditions.

6 Conclusion

In the work relating to this thesis, neural network transfer learning and fine-tuning were investigated and used, making possible a task that would have been intractable had a nave approach been used. There was no access to footage from the same physical camera position at multiple time points, only in single one-hour stretches, making a proper rigorous numeric investigation of the fine-tuning effect impractical. Nevertheless, the large observed improvement in accuracy, made simply by limiting the scope in CNN training data, is promising. The fact that there are multiple possible improvements visible is also an encouraging prospect.

Data collection up until this point was heavily limited, leading to an over-abundance of qualitative observations. At this point in the project Bercman is moving to live tests with the SPC posts, allowing for collection of footage at all times of the day from a fixed, unchanging camera angle. With this additional data, there is a hope that solid measurements may be taken and more mathematical certainty of the observed effects may be observed.

This thesis also makes use of the fact that automation can be built upon automation; albeit imperfect, CNN-generated training labels were used alongside scripts that do not require human intervention to create an efficient pipeline wherein an individualized CNN targeting a single set of static cameras is quickly and automatically trained. This new approach may be suitable for many other tasks where computing power is harshly limited. Rather than one large general neural network, a small system where specialized networks are trained with little human intervention can be employed.

References

[1] Gunnar Stefansson and Kenth Lumsden. Performance issues of smart transportation management systems. *International Journal of Productivity and Performance Management*, 58(1):55–70, 2008.

[2] Alexander Howard, Tim Lee, Sara Mahar, Paul Intrevado, and Diane Woodbridge. Distributed data analytics framework for smart transportation. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1374–1380. IEEE, 2018.

[3] Ming-Ching Chang, Yi Wei, Nenghui Song, and Siwei Lyu. Video analytics in smart transportation for the aic'18 challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 61–68, 2018.

[4] Antti Lehmussola, Pekka Ruusuvuori, Jyrki Selinummi, Heikki Huttunen, and Olli P. Yli-Harja. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE Transactions on Medical Imaging*, 26:1010–1016, 2007.

[5] Weidi Xie, J Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, 6(3):283–292, 2018.

[6] Giovanni Garibotto, Pierpaolo Murrieri, Alessandro Capra, Stefano De Muro, Ugo Petillo, Francesco Flammini, Mariana Esposito, Cocetta Pragliola, Giuseppe Di Leo, Roald Lengu, Nadia Mazzino, Alfredo Paolillo, Michele D'Urso, Raffaele Vertucci, Fabio Narducci, Stefano Ricciardi, Andrea Casanova, Gianni Fenu, Marco De Mizio, Mario Savastano, Michele Di Capua, and Alessio Ferone. White paper on industrial applications of computer vision and pattern recognition. In *International Conference on Image Analysis and Processing*, pages 721–730. Springer, 2013.

[7] Elias N Malamas, Euripides GM Petrakis, Michalis Zervakis, Laurent Petit, and Jean-Didier Legat. A survey on industrial vision systems, applications and tools. *Image and vision computing*, 21(2):171–188, 2003.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[9] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] CyberAILab. A closer look at yolov3, 2018.

[13] Zhanyi Wang. The applications of deep learning on traffic identification. *BlackHat USA*, 24, 2015.

[14] Jiyong Chung and Keemin Sohn. Image-based learning to measure traffic density using a deep convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1670–1675, 2018.

[15] Samuel Murray. Real-time multiple object tracking-a study on the importance of speed. *arXiv preprint arXiv:1709.03572*, 2017.

Acknowledgments

I'd like to thank:

- My co-supervisor Eric for constantly bugging me into writing my thesis. It worked.
- My supervisor Prof. Gholamreze Anbarjafari for giving me the opportunity of working on this project to begin with.
- All the ICV team members, but especially the people working alongside me in the SPC project.

Non-exclusive licence to reproduce thesis and make thesis public

I, Mateus Surrage Reis,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   University of Tartu Faculty of Science and Technology Institute of Technology
   Mateus Surrage Reis

   supervised by Prof. Gholamreza Anbarjafari

   (a) reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   (b) make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **21.05.2019**