An approach to a lightweight low-cost smart parking system with use of simple convolutional neural networks

Innopolis University

Thesis submitted to The Innopolis University in conformity with the requirements for the degree of Bachelor of Science.

presented by

Alena Yuryeva

supervised by

Alberto Sillitti

Date

An approach to a lightweight low-cost smart parking system with use of simple convolutional neural networks

(optional) dedication.

Contents

List of Tables

List of Figures

Abstract

In this work, we present a lightweight, fast and straightforward system for parking space occupancy detection based on a simple convolutional neural network processing images captured from a stationary camera in real time.

The system is fully working and nearly ready for production, open for public usage, and well-suitable for small businesses in developing countries. The system has sufficient accuracy and speed for everyday use and is extensible enough to be a basis for more complex intelligent transport systems. We have tested it against the state-of-the-art solutions to determine its viability, as well as tried out possible optimizations.

Chapter 1

Introduction

Technology has been easing people lives for thousands of years, thus letting them proceed the life more efficiently. Examples can be found as early as a discovery of methods to start a fire 1.5M years ago. This discovery helped the first people to get warmer, cook better food, and create culture communicating together in a more comfortable setting.

Then there was abacus to ease count, the wheel to make transportation faster. Fast forward a few thousand of years, and we have an internal combustion engine which was to transform the machinery, nuclear energy, first computers all of these improved human lives significantly. New tools are being created daily to optimize the functioning of the humanity, to save more time, to produce more goods. From small villages, humanity rose to cities. Now we want technology to take over even more parts of our lives, with smartphones and omnipresent internet. We have become lazy and want things to be done autonomously, seeking for more and more simplification of life.

Smart gadgets are now an everyday part of our lives, with nearly everyone having a smartphone, smartwatch, and probably even some high-tech home appliances. Smart houses are thus also slowly becoming a norm, with quick solutions for infrastructure. However, this is not the end, as what futurists often talk about is smart cities. It is the future reality we will eventually run into, something that is going to change our lives forever even without any breakthroughs in modern physics. Smart cities are an epitome of connectivity and humanly possible optimization.

1.1 Overview of emerging and trendy tech and supposed usage

The technical progress marches on, leaving a trace of buzzwords only a few people can truly understand, and a bunch of funny toys which later prove to be vital for the technological development, science, and society. Or just fade away, clearing the place for new ingenious inventions.

1.1.1 Virtual reality and augmented reality

Say, there is virtual reality, which refers to a specific type of reality emulation. We all learned about five humans senses at school, and there are also plenty of other senses such as a sense of balance, and all together they perceive information which we then call a reality. Thus there is the thought that arises, namely that whatever we percept, we can then believe it is real. Then the idea is to generate an input that would somehow stimulate one's senses with made-up information that is not there so that the perception changes and our brain thinks this input is real. In other words, virtual reality aims to present our senses with a virtual environment generated by a computer which can somehow be explored.

So technically virtual reality is a three-dimensional, computer-generated environment created for a person to explore it and interact with it. It is usually done using headsets, special gloves, and other similar equipment. It is not an easy task, as our senses need to be in sync, and if anything is even a little off, we are going to notice that. The reality, in fact, consists of tiny subtle details brought all together, which makes significant difficulties for making true immersion possible. The synchronicity done right, however, forms a so-called sense of presence.

The applications for virtual reality seem to be numerous. First of all, there is the entertainment area, with immersive films and video games. However, also, there are such things as:

- Arts
- Sports
- Medicine
- Architecture

The number of applications is infinitely expandable. Whatever is too dangerous to do in reality, can be done in virtual reality, especially as costs of the technology keep going down. Perhaps one day it can become a basis for communicating the way we have never known before.

The notion of augmented reality is also quite similar. This is a sort of middle ground between actual reality and virtual reality, which finds probably even more applications.

1.1.2 Data science: ML, AI, Big Data

Another hype technology that is associated with futurism is artificial intelligence (AI) which started as the simulation of human thought process by computer systems, which includes such processes as learning, reasoning, and self-correction. This term was coined as early as 1956, further changing its meaning and becoming an umbrella term covering many various areas, as AI was developing in directions that were not predicted at the beginning, and winning over human intelligence capabilities in several fields.

It was hypothesized that there could be weak AI, suited for a particular task, or a strong AI, which can generalize human cognitive abilities. However, at present, only weak AI has been developed.

Another classification comes from Arend Hintze, and includes four categories:

- Reactive machines, that can analyze the situation and make decisions, but cannot use past experiences to guide these decisions. They have a narrow purpose, an example being Deep Blue — a chess program built in the 1990s.
- Limited memory — a kind of AI which uses past experiences to inform current decisions, yet these experiences are not stored permanently. Some autonomous vehicles have been designed this way.
- Theory of mind — this level of AI would take into account the fact that other actors have their own beliefs, desires, and intentions which will affect the decisions made. This kind of AI is currently being developed, with especially numerous attempts to model theory of mind in the game development industry.
- Self-awareness, meaning that a machine would have a sense of self and use it to infer what others are feeling. This type of AI does not yet exist and is unlikely to be created in the not-so-distant future.

Some areas currently covered by the term AI include:

- Robotic process automation
- Machine learning
- Machine vision
- Natural language processing
- Pattern recognition
- Robotics

Examples of areas where AI can make a breakthrough:

- Healthcare — as a means to automatize medical diagnosis, medical feedback, and ease interactions with patients.
- Business — robotization of highly repetitive tasks, improvement of analytics, services for interaction with humans
- Education — automatized grading, adaptation to the needs of students, additional support. One day, AI can even mostly replace teachers, completely transforming the education field.
- Finance — financial analytics and consulting automated trading

- Law — search through documents, automated consulting
- Manufacturing — aims for total robotization of processes.

One of the areas mentioned above is machine learning, which powers all the automated analytics — it is the technology which is nowadays used virtually everywhere. The ultimate goal of the field is getting computers to learn and use information, improving over time autonomously, based on observation and real-world interactions. Regardless of learning style, all machine learning algorithms depend on representation, evaluation, and optimization, with a fundamental goal to generalize beyond what has been seen. Research in this area is often done straight in the industry, on real-life application, and often breakthroughs are obtained while using the technology in the fields it has never been used before.

Still, machine learning is not about complete automation, and thinking this way will make one miss the valuable insights that can be obtained from data. It is more useful as a fast and efficient way of working with big loads of information, which can be then analyzed by a human data scientist, helping make more informed decisions and contributing to faster problem-solving.

The rise of machine learning allowed to operate with what is now called the big data efficiently. The concept was coined in the 2000s and defined by such characteristics as incredible volumes, unprecedented speed of data streams, a wide variety of data formats, the variability of data flow velocity and complexity if the overall picture.

Amounts of produced information are growing exponentially, and so does the need to handle this information. This issue is becoming vital for cost and time reduction, product development and informed decision making. With the power of big data, one can:

- Detect frauds before it affects the organization
- Perform fast and accurate risk evaluation
- Determine the best moments for businesses to perform actions
- Determine causes of failures and shortcomings in near-real time.

With that much potential, there are areas which benefit the most from the use of big data:

- Banking — boosting customer satisfaction, minimization of risks and frauds and other advanced analytics
- Education — identification of students who need assistance, ensuring of adequate progress, better evaluation and support
- Government — analytics which helps to manage utilities, run agencies, or, for instance, prevent crime

- Healthcare — analyzing patient records, treatment plans and prescription information to speed up the process
- Manufacturing — boosting the quality of output while minimizing waste
- Retail — best ways of marketing, efficient transaction handling, strategic decisions

1.1.3 Internet of Things

Let us, for now, pause with the talk of how newest technologies can work with information and move on to the most down-to-earth parts. We are talking about the Internet of Things (IoT) — a new step in the Internet development, such that amount of connected devices outstrips the population of Earth. This epoch is believed to start around 2009, and in the future, it means the internet occupies the real world and devices start working without human help.

The term was coined in 1999 by Kevin Ashton when he proposed optimizing business processes using RFID (radio frequency identification) labels. In practice, now IoT-systems involve smart devices coupled with cloud platforms to manage them, served by data storage, processing, and protection systems.

IoT is possible due to a whole stack of technologies, both hardware, and software. There are multiple protocols of interaction, along with the concept of M2M (machine-to-machine) interaction. In the b2c segment, it manifests itself as wearables, smart home, smart clothes, smart TV and even smart devices for pets. In b2b, however, there are such things as connected and automated cars, smart city, usage-based insurance, smart workspaces, smart energy grids, smart factories, IIoT, agricultural systems, location-based marketing and many others.

There is an opinion that IoT is a purely marketing term, and technologies stayed the same. Whether it is true or not is not the point of this paper, yet it seems that what is called IoT may soon lead to new futuristic future.

1.2 Smart city infrastructure

Smart city infrastructure turns out to be being built with IoT devices helping combat the strain of city growth. Cities continuously face new challenges as the world population explodes along with crawling urbanization that has already made the majority of world population city dwellers. While an increase in the population allows city economies to flourish, high population density also put a strain on resources. Additional support is needed to ensure economic growth, which makes cities turn to IoT.

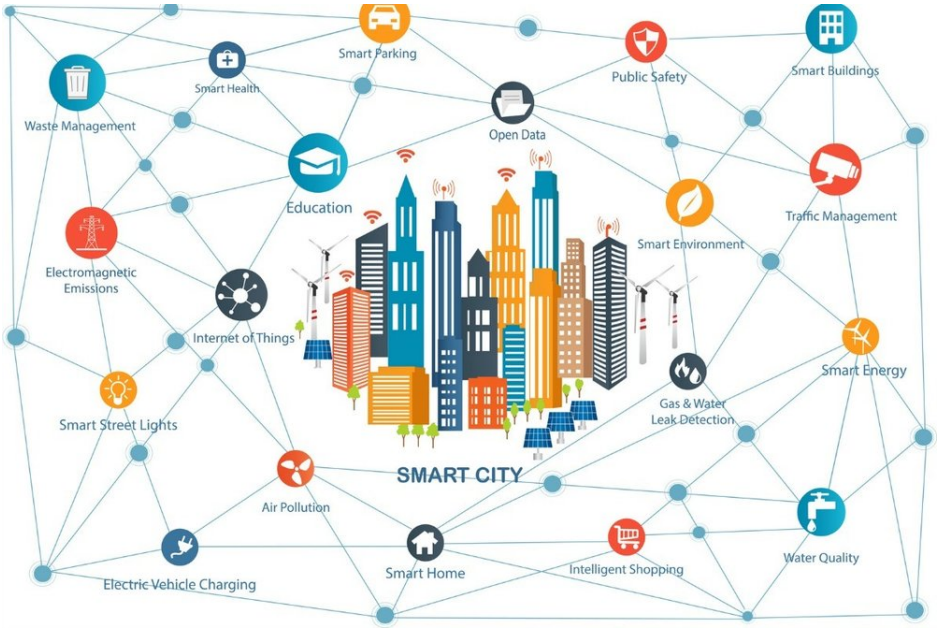
According to the United Nations:

> By 2030, the number of cities with 1 to 5 million inhabitants is projected to grow to 559 and 731 cities will have between 500,000 and 1 million inhabitants.

from "World Urbanization Prospects".

Some cities grow by 40-60% over just a decade. Such population explosions quickly lead to such problems as traffic congestion, environmental problems, and safety issues. Some attempts to solve these problems with the aid of brand new tech have already been made, with, for instance, Las Vegas turning to the Cisco Smart+Connected Digital platform to aggregate and process data from IoT devices in the city.

Figure 1.1: Smart city



With such centralization of information collection, such tasks as crime prevention, surveillance, trash pickup optimization and traffic lights automation can be solved. It is predicted that in 2020, the world is going to have over 50 billion devices connected to the Internet, with over 20 billion of them located in cities [1].

The data analysis enables the city to prevent problems instead of solving them. It can start small, with, for instance preventing trashcans from overflows. It also can get as global as ensuring better ecology in the city in such creative ways as adaptive traffic light control. Along with better infrastructure, it is also better service for people, meaning here such services as healthcare.

With such a broad variety of issues to solve, it is essential that all the data be gathered together, and enough access to this data is given to all existing applications. Open data policies have already proved to reduce costs [2]. Moreover, if the data is open and systems can interoperate, it is possible to connect smart infrastructures of different cities, thus having a gain in efficiency in spe-

cific fields through geo-specialization. The geographic boundaries can this way fade entirely with time. Furthermore, data exchange could enrich consumer experience, with analytics that takes into account multiple facets of data, thus giving a more vibrant picture of the world, leading to better decisions.

On the other hand, data privacy advocates currently argue against data sharing, as data privacy is already a significant issue for IoT, with so many security breaches the area suffers. There are already cases of massive hacking, such as Mirai botnet cyberattack in 2016. This issue calls for complex data access policies that would protect information from leaking to the wrong hands, probably based on location, time and many other factors.

The most severe ongoing issue that prevents smart cities from becoming mainstream is the global fear of change. It can be relatively easy to implement the systems but making companies rethink their business models is an entirely another story. One should have all the answers regarding why new approaches are better than current ones. In the end, the main thing is starting from somewhere, instead of analysis paralysis. The key is to find a problem, then solve it, then repeat infinitely.

1.2.1 Smart building

Aside from all the common infrastructure, one of the fundamental blocks of a smart city is a smart building — another concept that is yet to flourish to its fullest.

The first buildings in history were quite primitive, made from stones, sticks, animal skins and other natural materials. As different from modern buildings as is probably possible, the purpose of these buildings remained the same for thousands of years — to provide a comfortable space for the people inside.

Modern buildings have much higher complexity, with multiple structures, systems, and technologies being used. Each component of the building has been improving over time, so that now all the systems like ventilation, heating, security, lighting and so on can be chosen separately. Nowadays, it is not enough, as building owners start having more global concerns such as the impact of the building on the environment. This issue makes the industry start changing so that buildings of the future are made in an integrated, dynamic and functional way. The systems that constitute the building, now have to minimize energy cost, ensure robust power supply and reduce environmental impact.

More practical needs imply that systems constituting a smart building must hold the maximum productivity for its occupants, regulating the perfect illumination, temperature, air quality, security system, sanitation and many more, while keeping the lowest cost and environmental impact over all the lifetime of the building. Reaching this vision would mean that these factors have to be taken into account from the design stage. The smart building must control all the internal information technology systems, so that it is possible to optimize them overall, not just one by one. It is much more than inside-the-building systems — it is also about smart power grids and interaction with occupants.

The truly smart building would hold plenty of functions, which include connecting building systems, people, and technology, as well as connecting to the global environment and smart power grid.

Modern buildings already have some of these systems, but they are however disjoint, communicating over obscure proprietary protocols making data sharing hard or impossible, and those losing ways for possible overall optimizations, such as using data from security systems to turn off lights and reduce cooling when occupants are not present. Achieving connectivity would require lots of third parties which now compete to start working cooperatively.

Plenty of ways to reduce cost become possible with smart buildings, such as optimized cooling and ventilation equipment, matching occupancy patterns to energy use, proactive maintenance of equipment, dynamic power consumption, and so on. These features all require open access to information for all parts of the building.

Another issue is the connection to the global environment. It is of course already useful when the building can optimize itself, yet connecting it to a more massive infrastructure in terms of information flow can further optimize costs. It would require some middleware which converts all gathered information to a uniform format, to produce meaningful analytics and let functional executives see a big picture.

Connection to smart grids is another direction where costs might be decreased, with smart grids sending requests to reduce power consumption when energy costs go high or when reliability is jeopardized. It also means more open and honest electricity charges. For instance, there can be the centralized change in heating expenses as a reaction to weather forecasts as they become more reliable. Another usage is again analytics, helping building operators to make better decisions.

So overall, smart buildings are an important, although not the only part of a smart city vision as it is.

1.2.2 Information flow

From what has been already said, information exchange plays a crucial role in a smart city of the future. To ensure this exchange, new solutions are being developed to utilize existing infrastructure more efficiently. While for industrial nations this means expansion, for developing countries this is even more important since it might aid the basic construction of infrastructure, and also ensure that this infrastructure is made the right way from the beginning, thus accelerating economic growth.

Smart cities can optimize resource consumption, thus more or less compensating for resource scarcity. Energy, space, money and time can be tuned to be efficiently used dynamically so that they fit the current demand and resources are not wasted. To achieve it, it is crucial that infrastructure is interconnected in terms of information exchange, to create a holistic model of the environment and correctly use data on status, demands, and capacities, so that technology fits seamlessly into the everyday life.

Here are some essential requirements and possibilities for the information flow:

- The city should be a platform for everyones communication to make everyday life more comfortable, as information is probably one of the most valuable resources of today and tomorrow.

- Mobility must be ensured through traffic flow monitoring so that communications become more efficient.

- The power supply system needs to be converted into an intelligent energy information system, which would also provide information on availability and consumption, which among other things would make power consumption more thoughtful.

- For the public safety of citizens, prevention of force majeure situation is to become a priority. Furthermore, the safety of communication systems should also be ensured

- The interconnection between all means of transport in the city for the sake of coordination could allow efficient and wait-free traffic flow as well as lower noise and carbon dioxide emissions.

- As a means for public health, doctors, hospitals pharmacies, and health insurance companies should be even more interlinked than they are now, with equal access to the critical data, which would allow faster and more individualized care for each patient.

- Governmental processes are expected to be kept to a minimum while also being transparent. The smart city would connect citizens, businesses, and institutions with each other, allowing access to public data and increasing participation.

- The safety should reach far beyond single buildings, ensuring universal safety, providing information on the power supply, as well as letting residents control their private living spaces from everywhere and adjust them individually for their needs.

- The culture can also be enriched through multimedia-based communication leading to more fruitful social, economic and cultural exchange and therefore more extensive cultural education.

- One of the critical concepts of the smart cities is networked knowledge exchange which will among all other things contribute public education and awareness with open access to information.

1.2.3 Material goods flow: delivery, logistics

Information is undeniably a vital part of the smart city, yet the concept is even more about the material world, and material goods also need to be transported, leading us to the question of logistics. This question increases in importance as cities grow, while according to UN, 54% of the world population is now urban, and it is predicted that by 2050 the number is going to be 86% for developed countries and 64% for developing countries. The number of online shoppers grows by dozens of millions of customers every year, which places heavier emphasis on delivery. Moreover, in US delivery trucks currently dominate up to 26% of road capacities in cities [3].

The presented statistics suggest that soon the issues of traffic congestion, carbon dioxide emissions, pollution, and noise can become much sharper than they are today. To cope with these problems, multiple cities turn to the smart city paradigms with increased logistics management. A proper logistics management is in fact not possible over just one company, as it needs to take into account the whole pictures, forcing society to build tight collaboration between governments, business, academia, and many more stakeholders.

1.3 Role of transport

With such a heavy emphasis on logistics in the smart city infrastructure, all transport-related questions are in focus. The city wellbeing increasingly relies on smart transportation, which is a must for a modern society. Without proper transportation management, the city life will halt. It is essentially a big part of the quality of life in modern cities, thus there is an increasing demand for intelligent transportation systems (ITS). The growth rate of this industry is estimated to be around 25.1% per year over the next five years. An intelligent transportation network at its highest should have the following features:

- Public transportation management, to encourage more people to use public transport through effective automation, planning, and management of public transport, where data analysis is the key. It would also allow an individual to be informed about vehicle schedules and change, as well as provide better security.

- Route information, to provide travelers enough data to inform their choices to facilitate a comfortable and secure trip. All information is to be provided detailed and real-time, for easy access of drivers to all the information they might need.

- Safety and vehicle control, which would assist vehicle operators through information transfer, warning of possible collisions, compensating for reduced visibility through sensors or scene recognition, so that the number of accidents is reduced as well as damage from accidents that are impossible to avoid for some reason.

- Electronic timetable, to assist travelers with information on arrival and departure times, delays, cancellations, transfers and transport connections, so that they can make informed decisions and even last minute readjustments during the trip.
- Electronic payment system and single fare card, so that the customers do not have to waste time buying tickets for different means of transport, and have one card to pay for everything.

With these five components, the intelligent transport system can be considered to be done. However, all these novel features that are to be introduced need also to be based on some technologies needed to implement all parts of this complex infrastructure. Here are some of them:

- Advanced tracking system. Even nowadays, most of the vehicles have an embedded GPS system to help navigation. However, these systems might gain a much more extensive usage, as they offer two-way communications and can thus be used to gain valuable statistics, coordinate vehicles, prevent speeding, provide emergency services and many others.
- Advanced sensing technologies. Intelligent sensors might be implanted not only into vehicles but also into road infrastructure. These might be used to ensure safety, gather valuable statistics regarding road situation, help traffic control, and law enforcement
- Advanced video vehicle detection, which can insanely help traffic management, identify emergencies, provide security and gather statistics.
- Advanced traffic light system, which may offer flexible traffic light management which can adjust to traffic situations, time of the day and special event without the need for manual management
- Emergency e-call vehicle service. In case of emergency situation, the vehicle must be able to connect to nearby emergency centers to automatically provide valuable information, as well as help driver to connect with a trained operator. This feature has already become mandatory in all new vehicles in Europe.

Intelligent transportation systems can provide plenty of significant benefits to the world as a whole, substantially affecting global economic and ecological situation. Here are some of the benefits which might be obtained by the city which is implementing one of these systems in practice:

- Minimized pollution. Intelligent transport systems, in general, promote the more massive use of public transport, thus lowering carbon dioxide exhausts due to reduced private car usage. It is possible to do with real-time information on public transport, aside from improved accessibility of public transport itself. Also, with intelligent transport systems, it becomes possible to stimulate carpooling, bike sharing and clean fuel usage.

- Security and safety. With vast amounts of data collected through sensors, GPS, CCTV, and internet connection, emergency services can be provided efficiently, while most of the accidents can be prevented. Such amounts of surveillance may also help to prevent terrorism.

- Increased demand for mobile apps. The new approach to transportation is going to depend a lot on smartphones and mobile apps for transfer of all kinds of information to the end user, which is going to influence mobile application market positively

- Smart parking solutions. Today, parking is a massive pain for cities, affecting every city resident. With the right infrastructure, internet connectivity and practical usage of security cameras can minimize the negative impact of parking problems on the city. Good infrastructure and overall informedness will one day aid the infrastructure to handle ever-increasing vehicle streams.

- While experiments with intelligent transport systems have already been launched in USA and Europe, it is still a challenge for developing countries due to the lack of finance, illiteracy, lack of IT infrastructure, unsensible city planning and infrastructure and other similar issues. To make any moves in this direction, it is essential to first invest in the general education of the population which would affect their mentality, and then launch governmental initiatives.

1.3.1 Smart cars (emerging needs and solutions)

Let us talk about smart cars in more detail now. Today, the rise of self-driving cars is imminent, with all advances in car manufacturing, Wi-Fi, and power of smart devices. According to the Forbes journal, it is predicted that by 2020 about 10 million self-driving cars are going to be in use. This innovation would save people time previously spent driving and parking, resulting in better usage of precious time of our lives.

Responsibilities are going to shift, now being more heavily placed on manufacturers who will need to improve self-driving technologies to the point of perfection, while driving is one day going to stop being an essential skill it is now. It is estimated that in nearest future smart cars will be able to save their users about 50 minutes daily, meaning more time to work, improve oneself and communicate with friends or family.

Vehicles are going to be able to avoid collisions by exchanging their positions on the road, avoid traffic congestion, and save over a trillion dollars spent on the fuel wasted in traffic jams annually, as well as reduce carbon dioxide generation. In the future, it is going to be possible to increase the vehicle autonomy by sending it somewhere remotely controlled. Along with infrastructure sensors, it will also be possible to cut down on time, spacing and fuel costs, for example through information on available parking spaces, which is going to eliminate up to 30% of traffic congestion.

About 90% of traffic accidents are caused by a human error due to stress, fatigue, recklessness or distracted driving, at the cost of about 1.3 million human lives every year. Smart cars can eliminate the human factor by handing decision making to smart systems which are permanently alert and responsive. However, this is an open question when this will indeed become possible. Accidents tend to happen even with the most reliable technology, so the influence of these changes on the insurance industry is yet to be seen, and mass controversies are inevitable.

1.3.2 Parking (emerging needs and solutions)

As mentioned above, parkings are nowadays a major pain of cities and city dwellers. They are a constant source of frustrations for everyone who depends on transportation to be able to perform their functions and roles, thus making innovative smart parking services a key priority for smart cities. It is possible to determine whether a parking space is occupied or not with cameras or sensors engrained into the ground.The data might be then sent to the global cloud hub and be used for coordination so that a real-time parking map is created. Drivers could then access the map from their mobile phones, while parking control officers could use it to identify parking violations. It is crucial to understand who and how is affected by parkings so that the potential for smart parking solutions becomes evident. For drivers, parkings are often seen as a necessary evil, influencing the choice of places to go. Parking enforcement agencies are often perceived negatively, despite the fact that only 5% of parking violators are summoned to court. However, for business, the most crucial issue is the choice of location, so that the parking is easy enough, or else they risk need for relocation, underperforming or even shutting down. Finally, for residents, parkings are a big deal if one uses their car daily. Difficulties in parking make people create nontrivial and potentially dangerous alternative parking arrangements, park far away from their living places, or even choose to move. All cities heavily depend on tax revenue they get to generate necessary public services and support businesses. They must balance parking rules enforcement with a nontrivial task to not alienate drivers, visitors, and businesses due to overzealous law enforcement in order not to lose revenue. It is possible to make much innovation and go creative in the area, if we consider correctly all the paths which can be taken, some of them being:

- Sponsored meter time extension. Everyone dreads parking tickets which run out of time at the worst moment. So a better idea would be to notify a driver or even a merchant that the time is expiring so that they can prolongate the time easily through smartphones.

- High value, high priority enforcement. Not all parking violations are equal, so it would be reasonable to identify those violators who can affect public safety to a greater extent. Such an approach helps prevent real problems before they become disrupting.

- Parking incentives to drive business growth and economic development. The fundamental idea is to give merchants partial control of the parkings so that they can provide incentives to make customers come at specific days and visit their organizations more. The increased visitor traffic would thus allow getting more significant revenues.

- Efficient citywide parking space utilization. The collected data might be used to dynamically adjust parking times and prices so that spaces which are almost always empty can be utilized, and spaces which drivers usually compete for can provide more revenue. It can even go as far as coordinating businesses to host events on the days on which a low parking load is typically expected or encourage suburban citizens to use more public transport.

However, all these smart solutions have the same basis, namely the need to determine whether a parking space is empty or occupied, which is, although quite simplistic, can be even called the core problem of smart parking.

1.4 Economical issues

All these beautiful futuristic explanations surely sounded like a fairytale till now. However, in reality, making a smart city is economically problematic. One Chinese parking slot sensor currently costs about $180 on Alibaba. Another example is Libelium-Metiora Smart parking Sigfox kit which costs 2850 and allows to equip ten parking spaces. Other systems are just as expensive, and this is not what most of the possible stakeholders can afford to use. The path to living full of high tech turns out to be more expensive than anything.

Still, the humanity absolutely needs to move on to the direction of the smart city to improve economy and ecology, as can be seen from descriptions above.

1.4.1 Economical state of developing countries

Furthermore, developing countries do not currently live their best years. In 2016, their economic growth slowed down to 3.6%, the slowest since the global financial crisis in 2007–2008, due to weak global trade and financial instability. Some economies even suffered contractions, though China and India saw robust growth. Average growth is expected to reach 4.7% in 2018, but still, it is not a very promising number. According to UN "World Economic Situation Prospects": The countries might require some help in the form of monetary and fiscal policies. This issue is the reason why solutions good for the first world just wont work in these countries. There is an urgent need to think of something cheaper so that it can be adopted there.

Even if the state does not enact any policies regarding smart city solutions, they still can be beneficial for the private segment, especially for the small business.

1.4.2 Companies cannot afford expensive software

Small businesses, however, most often cannot afford the software they need, as the prices are often quite unpleasant even for the first-world countries, and for developing countries, they become outright cosmic. Furthermore, small businesses have fewer capabilities than big corporations to afford solutions they need.

So they need to handle this in different ways. They may choose to look for open source solutions which are free for everyone, yet there are some drawbacks in them. Some can be only used for non-commercial purposes, and other can lack quality. Then there are solutions which cannot be used because of the language barrier. Also, some of them require at least basic technical literacy not everyone has.

So other companies turn to piracy. This way is sometimes more accessible, and sometimes more difficult, yet it helps to obtain products which are often of a higher quality, more mainstream and more familiar. It is also tied to many difficulties. First of all, they might have significant problems with the law and face fines. They can be denied some functionality available only with proper licensing, and the software will lack stability and reliability this way. Finally, some people consider piracy plain unethical, which can alienate some customers.

Some businesses choose to opt out of both and do it all by hand, using old ways. This way is free and reliable, yet often takes way more time and leads to stagnation, preventing an increase in profits.

There is also such a solution as Software as a Service, where one can obtain a powerful service in the cloud with a pay-as-you-go policy. This one is becoming more popular over the years, yet over 50% of companies are not yet ready to use such services due to concerns on privacy and security, as well as the reliability of such systems.

While Software as a Service is a powerful solution of the future, for now, the open source remains what is probably the most feasible option.

1.4.3 Companies cannot afford top-tier hardware

Of course, we all want to be the owners of the very best hardware, but the truth is the costs of such hardware are well beyond the opportunities of an average business. One wants a latest top-tier iMac Pro with all features — fine, if they want to spend over $13500 on it, let it be. Except this can be an annual income of three or four security guards somewhere in Russia.

Let us now get real. A business will aim for as much economy as possible, buying minimal configuration required for functioning. In our case, this should be a video server, as we must take care of companies and individuals which cannot afford expensive and not always reliable sensors and opt for cameras for their simplicity and relative cheapness. After all, if one adds more slots to their parking, they could either buy more sensors or relocate a camera.

So in 2018, here is how a typical video server for a small business in a developing country would look like:

CPU: Intel Core i3-6100 — 7000 RUR
HDD: 2TB — 5000 RUR
RAM: 4GB — 3000 RUR
Motherboard: LGA 1151 — 3000 RUR
Case: 2000 RUR
Keyboard & Mouse: 1000 RUR
Power Supply: 2000 RUR
Display: 20 inches — 2000 RUR

The whole set totals about 25000 RUR, or about $420, being over 30 times cheaper than the top hardware. This data has been found on the internet using information on retail websites.

However, this is just hardware alone! Other expenses include an operating system and others software, as well as cameras and reserve power supplies.

Apparently, the described computer is outright weak, and is what an average geek would call outright garbage. The point is that our smart system must be able to run on such hardware and thus should be lightweight enough for that.

This work has been done by a team of two bachelor students so that we have a joint project but separate thesis papers. There are many repetitions ahead, yet the teamwork allowed us to produce amounts of work which are not that easy to perform alone.

Chapter 2

Systematic Literature Review

2.1 Rationale

Following the establishment of the end goal of this work, we started researching the field to find out if there were attempts to solve the outlined problem, what issues they focused on, as well as the extent of success of every attempt that occurred. To keep the research thorough and reliable, we focused on tapping into exclusively academic sources.

Moreover, due to a literal explosion of amounts of available information in recent years, we had to find a way to limit our research so that we fit into deadlines of our thesis work, while also keeping the review academically valuable. This issue was the reason why we chose to perform a systematic literature review, which need not be comprehensive but also provides valuable statistical information that proved to be vital in our attempt to determine best approaches. While SLR is said to be way more effort-consuming to perform than a classical literature review, it also provides a handy framework for structurization of information, that allows for an extent of consistent control over the quality of information we are obtaining.

2.2 SLR protocol development

2.2.1 Research questions

In this literature review, we attempted to answer such research questions that they include maximum relevant information which we need to know to implement the system that suffices the requirements put on the proposed system for it to fit the cause that drove us towards this field. Here are those we pitched upon:

- **RQ1:** What technologies, tools and methods are suitable for implementing a smart parking system?
- **RQ2:** Which efficiency parameters were used to evaluate described systems?
- **RQ3:** What results were obtained in the course of evaluation?.

2.2.2 Searching process

This section provides the specification of searching process, including used search resources, keywords and search queries, study inclusion and exclusion criteria, evaluation scheme and the related methodology.

Resources

The search included electronic sources only. We mostly used the following sources and tools:

- IEEExplore Digital Library
- ResearchGate
- Google Scholar
- ACM Digital Library
- Arxiv.org

This list is far from exhaustive, as there were some papers which we have found over the course of the unstructured search. It is, furthermore, quite senseless to try to track each paper to some exact source, since even listed sources are often crosslinked or contain different versions of the same paper.

Search queries

For the search, we first needed to identify the key topics we needed to perform our research on. The first one was, apparently, smart parking. Also, due to us having to implement a standalone software system, software design was another one that came naturally. The other topics were trickier. Over the course of pre-search, we saw that all similar systems employ either sensors of cameras and since sensors are relatively expensive while cameras are everywhere, we decided to tap into camera-based systems. This decision would mean that another topic was computer vision. The next one was machine learning, due to being the most widespread group of methods used for the task accomplishment. This approach always requires plenty of training data to perform fine-tuning of predictive models; thus another topic came to be datasets and data collection. So overall, here is how the final list of key topics looks:

- Computer vision
- Machine learning
- Software design
- Smart parking
- Data collection on vehicles

For each topic, key concepts were written down, then synonyms found. As a result, all searches were performed using OR-combinations of the following queries:

- (automatic OR smart OR IoT OR internet of things) AND (parking OR parking lot OR parking slot OR parking space) AND (utilization OR occupancy OR tracking OR detection OR monitor OR management) AND (sensor OR camera OR video)
- (trigger OR event OR conditional execution OR decision making) AND (patterns OR design principles OR implementation OR usability OR user-friendly OR GUI OR user interface)
- (vehicle OR car OR moving object) AND (data OR dataset OR metrics OR information)
- (machine OR supervised OR unsupervised OR deep) AND (learning OR classification OR regression) AND (algorithm OR dataset OR evaluation OR application)
- (computer OR machine OR automated) AND (vision OR recognition OR image processing OR scene reconstruction) AND (moving object OR shape OR vehicle)
- parking AND lot AND occupancy AND detection AND computer AND vision

The queries were edited to fit in the search opportunities of each system, but overall the semantics was left the same. We performed the search across titles as well as abstracts.

2.2.3 Inclusion and exclusion criteria

To decide which papers will be included in the review the following inclusion criteria were employed:

- Found using search queries specified above
- Related to at least two of the five specified key topics
- Written in English

- Published not earlier than in 2000
- Journal and repository articles, conference proceedings, master's and doctoral degree dissertations
- Peer-reviewed
- Primary studies or systematic literature reviews

Papers that did not fit at least one of inclusion criteria were excluded from the review, as well as duplicates.

2.2.4 Quality assessment

We have developed a set of questions to assess the quality of included papers. The yes answer yields 1 point, partially yields 0.5 points, no yields 0 points. The points obtained on every question are then added up. The questions are:

1. Are objectives clear and specific?
    - 1 point if research questions are clear explicitly stated, or if the proposed system concept is described comprehensively
    - 0.5 points if there are some objectives of research that are related to the field
    - 0 points if no objectives are stated, or objectives are hard to determine
2. Does the research satisfy the objectives appropriately?
    - 1 point if research answers precisely the question that was stated or implements the concept that was proposed
    - 0.5 points if research is related to objectives but went somewhat off track
    - 0 points if research is unrelated to objectives
3. Is research process clear and reproducible?
    - 1 point if paper specifies clear steps, methods, technologies, and data, along with all necessary references and sources, needed to reimplement the research
    - 0.5 points if minor details are lacking, such as a dataset that is not readily obtainable, that can be inferred or replaced, or obtained via a few questions in an email
    - 0 points if it is impossible to restore the sequence of actions, or critical details missing, such as an algorithm or technologies used
4. Were the results assessed properly in a paper?

- 1 point if provided results allow to fully reestablish the structure of the dataset in terms of its labels (i.e., results in a form equivalent to providing a full confusion matrix)
- 0.5 points if published results can be used to somewhat evaluate the solution, but the full dataset structure cannot be restored
- 0 points if provided results are not sufficient to give an independent evaluation of the solution

5. Were the results summarized to provide a clear conclusion?

   - 1 points if results were summarized in an appropriate and meaningful way
   - 0.5 points if results were over-summarized or under-summarized, or there was an apparently more meaningful way to summarize the data
   - 0 points if no attempt to summarize data was present, or if conclusion seems unrelated to the data

6. Are there any comparisons with alternatives?

   - 1 point if comparison with other solutions is performed, with advantages and disadvantages clearly stated
   - 0.5 points if only comparison with previous iterations of the same work was performed, or if the comparison was not comprehensive
   - 0 points if no comparison provided.

So basically the score range is 0 (the worst) to 6 (the best).

Apart from evaluation, we needed to retrieve some amount of meaningful information required to decide on our future track in this project, so for this reason, we have compiled a paper questionnaire, which then was used as a structured tool to retrieve the data by answering pre-specified questions. For the questionnaire, we have created the following questions:

- What is the dataset used in this study?
- In what ways was this dataset preprocessed before feeding it to the predictive model?
- What methods were used for prediction?
- What was predicted? (the range of output values for the predictor)
- What were the results of prediction?
- Is there any information on the processing speed?
- Is there information on any additional features of the system?

2.2.5 Search and synthesis strategies

For papers found using resources and queries specified above, papers were evaluated according to inclusion and exclusion criteria, as well as to questionnaires, by both researchers. In case of disagreement, we discussed the application of criteria and agreed on conclusions. The synthesis was performed both qualitatively and quantitatively. Statistics about the papers were gathered, such as key topics, years of publication and quality assessment. Also, we classified the found information by methods of data processing, prediction, and evaluation.

2.3 Results

2.3.1 Search Sources Overview

First of all, we are going to represent some information regarding the resources used to search papers, to aid assessment of the work done. In Table 2.1, we discuss advantages and disadvantages of the sources.

Table 2.1: Sources advantages and disadvantages.

Source	Advantages	Disadvantages
IEEExplore Digital Library	Published papers can mostly be assumed to be peer-reviewed	Paywall
ResearchGate	Open access; well-formatted, full-color articles	Temporal delays of registration, no confidence in reliability of studies
Google Scholar	Diverse sources, helpful in finding open access articles	Some sources can be unreliable, and some behind the paywall
ACM Digital Library	A good source of interesting and high-quality papers in IT, convenient means of search	Paywall
Arxiv.org	Open access, diverse papers	Very likely to be not peer-reviewed, sometimes outright poor quality, non-intuitive search

2.3.2 Excluded Papers

There were thousands of paper found with our queries. Nevertheless, the time constraints did not allow us to look up them all, so we opted to only look through first 100 papers in every search. We threw out everything that did not fit inclusion criteria, that was, fortunately, possible to do due to the nature of criteria, which are generally possible to determine by metadata. As a result, we have gathered 111 papers, which were reviewed one more time during the

quality assessment. After review, 25 more papers were discarded, leaving us with 86 acceptable papers. In Table 2.2, we present information on reasons the papers were excluded. So, hereinafter we are only going to work with 86 remaining papers in this review.

Table 2.2: Reasons for exclusion

Reason to exclude	Quantity	Percentage (of 111 papers)
Not a primary study	4	3.6
It wasn't peer-reviewed/Draft	5	4.5
Duplicate	5	4.5
Not in English	2	1.8
Not conference proceedings or journal articles	3	2.7
Other reasons	6	5.4
Total	25	22.5

2.3.3 Studies Classification

Now, let us first present some statistics on quality and content of studied papers, and then the overview of the said content. We have a Table 2.3 that organizes information by major topics we have. Note that these topics are not exclusive, as one paper may belong to several major topics, so percentage will not add up.

Table 2.3: Key topics distribution

Topic	Years	Number of papers	Percentage
Computer vision	2003-2017	50	58.1
Neural networks	2014-2017 (one paper 2003)	21	24.4
Machine learning	2006-2017	48	55.8
Scene reconstruction	2009-2017	10	11.6
3D	2001-2016	8	9.3

Table 2.4 describes statistics by years of publication. For the sake of simplicity, we split all papers to five-year periods, with the last one being 2016 to present time, or a shorter one.

As we can see, number of relevant papers grows approximately twice every five years, and since we got almost as many papers for 2016 and 2017 as for previous five years, it is safe to conclude that interest in the topic is rising rapidly (Table 2.4). Furthermore, according to our observations, the overall quality of research is steadily improving over the years. Speaking of quality, in Table 2.5 we present statistics on quality assessment.

The good news is latest research papers have significantly improved in quality, so QA scores are mostly 4 or above (Table 2.6). Nevertheless, even papers

Table 2.4: Distribution by year

Years	Quantity	Percentage
2000-2005	7 [4–10]	8.1
2006-2010	16 [11–26]	18.6
2011-2015	32 [27–58]	37.2
2016-now	31 [59–89]	36

Table 2.5: Quality assessment statistics

QA score	Quantity	Percentage
0-2.5	37	43
3-3.5	19	22.1
4-6	30	34.9

with poor QA score can be useful in providing valuable information, even though we would not fully trust the results regarding the efficiency of applied technologies.

Table 2.6: Quality averages by year

Years	Quantity	Average Quality
2000-2005	7	2.2
2006-2010	16	2.7
2011-2015	32	2.9
2016-now	31	3.9

Let us now look at the content of studies we gathered. In Table 2.7, there are some statistics regarding methods of feature extraction employed.

Table 2.7: Features extraction methods usage

Feature used	Number of papers	Percentage	Years
Background subtraction	9	10.5	2006-2016
SIFT	8	9.3	2015-2016
HOG	7	8.1	2012-2017
BoF	4	4.7	2015-2017
GIST	2	2.3	2015-2016
SURF	2	2.3	2012-2016
Other	18	20.9	2003-2017

As we can see, all these methods keep being used after their introduction, and overall, a wide variety of features are employed. The most widely used are background subtraction, SIFT, and HOG, and what is important, HOG keeps being used even in 2017. Also, though it cannot be seen from the table, it is worth noting that HOG was mostly used in cases similar to our problem.

The Table 2.8 is dedicated to machine learning methods and statistics of their usage so that it is clear which of them are the most widely used.

Table 2.8: Machine learning methods usage

Feature used	Number of papers	Percentage	Years
SVM	24	27.9	2007-2017
Neural network	21	24.4	2014-2017 (one from 2003)
k-(something)	13	15.1	2008-2017
FCM	9	10.5	2009-2016
Bayes-based classifier	5	5.8	2006-2017
Logistic regression	5	5.8	2014-2017
Other	10	11.6	2006-2017

Here it is easy to see that some methods are older than others, but most of them keep being used up to 2017. Fuzzy clustering, logistic regression, and Bayes-based classifiers do not seem to be very popular here, while SVM and neural networks are more widespread.

One could notice that information in the table dedicated to features does not add up to the total number of papers. This issue can be explained by the fact that not all methods require usage of features, so, for instance, papers dedicated to neural networks did not usually describe any feature extraction.

Speaking of methods for evaluation, the Table 2.9 displays statistics on efficiency measures used in papers. These are used to evaluate the quality of prediction in a meaningful way, thus allowing to compare different methods.

Table 2.9: Measures of efficiency usage

Evaluation measure	Number of papers	Percentage	Years
Accuracy	54	62.8	2005-2017
Recall and Precision	8	9.3	2012-2017
ROC curve	8	9.3	2010-2017
Confusion matrix	7	8.1	2013-2017
mAP	4	4.7	2015-2017
F1 score	2	2.3	2014-2016

Here one can even see, along with conventional measures, some more exotic, such as mAP (mean average precision). Also, it is evident that accuracy is overall prevailing.

Note that numbers here are not expected to adapt. The reason for that is that some papers did not contain proper evaluation, and other opted to use several methods at once to show different aspects of prediction.

Note also that there were papers related to work with sensors, not cameras, which were of general interest, but did not fit our purpose due to relatively high

expenses on sensors and vast amounts of extra work required to mount them correctly. Nevertheless, a few of them employed machine learning as part of the approach and were quite amusing and beneficial for our cause.

Now with all the information we have gathered, and summaries we obtained and presented in this section, it seems feasible to infer more general and high-level facts from the information we have.

2.4 Discussion

In this section, let us discuss the overall picture we got while gathering information in existing studies. First of all, most of the papers did not look very trustworthy after quality evaluation. Two of the most common flaws were lack of reproducibility, which cast doubts at the validity of all the work, and underreporting of results, which also made checks on whether results are actually that good, problematic at best. While we understand that those experiments might well have been valid and thorough, and recognize efforts of fellow researchers, the way such studies are reported makes them only a barely reliable source for reference. Another problem seems to be systematic underreporting of negative results, which means that while good practices are well-known, the research community is forced to make all the same mistakes all the time, which apparently makes our work way harder.

Let us now answer questions from the paper questionnaire we designed, which were not answered as tables yet.

There were numerous datasets used in studies we explored, including the following:

- PETS
- PASCAL VOC [90]
- Microsoft COCO: Common Objects in Context (COCO) [77]
- VIVA
- Caltech 101 [91]
- ILSVRC [92]
- PKLot [57]

Some of these proved to be useful on some stages of our work later on.

Considering the prediction outputs, multiple cases are possible:

- The binary classification was among the most widely used, especially in tasks aimed to detect parking occupancy. In this case, values were EMPTY or OCCUPIED.
- Other studies used commonly exploited picture datasets, where the task was to classify an object into one of many categories (multiclass classification).

- There also were those aimed to detect an object, in which case, multiple labels could be assigned to a sample which contains multiple objects

- Finally, studies related to license plate detection had models that were to construct a value from the picture, and in this case, the task did not amount to a simple classification, and value range could have been quite broad.

Results of prediction, in their turn, also vary massively, with accuracy ranging from 62.5% in cloudy weather [18] to 99.4% [79]. The range of processing speed was even more widespread, being between 0.5 fps for VGG-based Fast-RCNN Ren2017 and 59 fps for the SSD system [59]. This range is, however, not exhaustive, as many papers did not contain any information on performance, so it only was reported in cases where results are scientifically valuable in the sense that they allow the scientific community to get closer to real-time object recognition. It is well-known that in fact, old neural networks could process a picture for up to 7 minutes.

Also, there are some papers worth noting for special features of the systems described. These include 3D-based parking space detection [38], unbelievably high prediction speed [59] and radar-based prediction [48].

As can be seen from the Table 2.4, this area becomes gradually more and more of interest for researchers, so we can interpolate that the interest is going to keep growing, attracting more people into the process of finding new, better solutions, especially since there are numerous problems in the field.

As for quality, from the Table 2.6 it is clear that there is a certain tendency in the fact that reporting quality improves with time, so one may hope that in the future it is going to be easier to rely on the background work.

The most popular methods of feature extraction seem to be background subtraction (supposedly due to simplicity), SIFT (which may be explained by amounts of information this feature vector preserves) and HOG (which may have something to do with the fact that it is best suited to represent shape). However, background subtraction is believed to be rather time-consuming due to the need to operate on a large number of pixels.

As for machine learning realm, the most widely used solution seems to be the application of artificial neural networks, especially CNNs, due to them being excellently suitable for work with pictures, without the need for actual feature extraction. SVM is almost just as widespread, probably due to the simplicity of the model, as well as the fact that this class of predictors aims for the optimal interclass border. Of methods for evaluation, accuracy beats all records, which is explainable by the fact that it is a simple measure that can be easily applied to multiclass classification.

From the statistics we gathered, it is easy to see that once a given method comes in use, it stays in use for most of the cases. The dates for early use are, however, nonrepresentative as they do not seem to correlate with dates those methods were invented.

2.5 Conclusion

As mentioned in the introductory part, our goal is to develop a system that is fast enough to work relatively real-time, undemanding in terms of computational resources in the process of use (and also in the process of training, as we do not have many computational resources), and reasonably accurate. The idea is to employ a simple solution not yet found in papers, and compare it to existing solutions in terms of both accuracy and performance for evaluation. Based on these goals, the research question for the rest of the work is the following:

RQ: How the proposed system compares to state-of-the-art works covering smart parking in terms of accuracy and performance?

Chapter 3

Methodology

3.1 Motivation

When one considers the idea of this thesis, it is becoming clear that the system is going to have quite a complicated structure in order to do everything it is supposed to do efficiently, and technologies that are to be combined are all nontrivial in and of themselves. Machine learning and computer vision are both highly sophisticated fields, so it is crucial to determine the methods we are going to use and ensure a good understanding of theoretical background behind them.

The facts mentioned above become the incentive to consider thoroughly describing the methodology of our work to, first, ensure maximal reproducibility of the results, as our final intention is to enable the system to be widely used in industry and for the public good with only cosmetic changes and overall broad adoption. Another reason is that we need to give evidence persuasive enough that usage of the proposed system can be expanded to the multitude of other use cases, and in the long run, serve as another step in development and usage of IoT.

Here is a brief plan of what we are going to cover in this chapter, for readers convenience:

- General strategies for data collection
- Discussion of issues of data preparation
- Suitable computer vision methods for image segmentation and feature extraction
- Suitable machine learning models
- Overview of software development techniques

3.2 Data collection

3.2.1 Rationale for data collection

One cornerstone of the development of systems that involve machine learning is data collection, since it includes training, and training needs data. Still, just having terabytes of data will not necessarily save an aspiring researcher, since one will also need to have it structured and be able to make sense of it to make predictive models work.

One of the most significant issues with data is that all datasets are flawed, that makes data preparation a crucial step of the process. Over the course of data preparation, one should make data more suitable for machine learning, and in broader terms, data collection is also a part of data preparation process, and often the most time-consuming step, where one might spend months on preparing the data before building the first algorithm.

There are three main variants of where to obtain data:

- Downloading an existing dataset, or combine several downloaded datasets.
- Performing field data collection with subsequent processing
- Combining the first two variants

3.2.2 Existing data sets

The first approach is quite straightforward. There are plenty of datasets out there on the internet — for pictures there are PASCAL VOC [90], COCO [77], and many others — and once one has downloaded one, they have a general purpose dataset, which is more than enough if they only want to compete in accuracy measures.

This method is quite quick and easy, as there is no need to spend time and resources to collect data from real life. At most, some reformatting is required, but the data is there, and it is already labeled. On the other hand, there are multiple drawbacks. There might be no data for the given study, or the data might be non-representative, some important information might be lacking, or there can be plenty of outright unnecessary samples. In the end, one might end up spending not less effort solving these issue than they would with manually collecting the data.

3.2.3 Collecting data sets

So this is basically why the second approach exists. A general purpose dataset might not always fit the specific purpose, and often there is just no dataset that would be sufficient. This issue makes a case for manual data collection, and this is, to put it nicely, not the most straightforward task. Big companies frequently hire some people specifically for that matter.

In other words, it is a long, expensive and resource-consuming process. On the other hand, it yields dataset which, if assembled right, will ideally fit the purpose.

So what is needed to collect a decent dataset, for it to be representative enough?

1. First of all, it is crucial to determine the goal of data collection. The more thoroughly the problem to solve is formulated, the easier it will be to determine and lead data collection. There are plenty of possible goal options, and all of them are good if the goal is well-formulated:

   - *Classification.* One needs to answer either a binary yes-no question (empty/occupied, ill/healthy, red/blue), or a question that has a finite predetermined number of possible answers, or categories (5 breeds of cats, ten genres of music, and so on). The right answers will need to be provided as labels.
   - *Clustering.* One needs to classify data into some classes they do not yet know, with rules that are currently unknown. The main difference from the previous variant is that one does not know in advance principles of division into groups. This situation might occur, for example, while trying to group customers by their behavior.
   - *Regression.* One needs some numeric value to be yielded. Examples can be the estimation of reasonable price, or of the number of items to prepare for a given day.
   - *Ranking.* One needs to rank objects using features. This approach is used in systems that recommend movies, music, or goods to a user.

   Most likely the problem falls into one of these categories, but even in other cases, the key is to avoid overcomplication and state a simple goal.

2. Establish data collection mechanisms, or rules on how to collect, store and process data.

3. Collect data based on predetermined rules.

4. After the data collection, formatting is required to make it consistent. The formatting here is about format consistency of data recording. It is especially true for data gathered from multiple sources by many people that it is essential to ensure consistency of date formats, money formats and so on. Another aspect of data consistency is all values conforming to the set constraints.

5. Reduce data to what is necessary. While it can be tempting to include as much data as possible, common sense should guide a data scientist into reducing data to amounts that are sufficient for the task but at the same time not overwhelming.

- One of the approaches is attribute sampling. One can determine which values in a record are critical, and which do not contribute much, thus reducing the size of the record by throwing out unnecessary values. Note however that domain expertise plays a prominent role here since one might need to be an expert in the domain the task is related to, to determine whether a given value is meaningful or not.
- Another approach is called record sampling. Mostly, this means removing records with missing, erroneous or less representative values, so that prediction becomes more accurate. One might also need to randomly shuffle the dataset and retrieve only a fraction of it if there is too much data.
- Aggregation of values might also help to reduce the number of records. For instance, instead of using daily values, one can add or average them over weeks or months and thus drastically reduce the amount of data to work with.

6. Clean data. At this stage, it is time to look at remaining missing values and do something to them. Concerning machine learning, it is better to have assumed or approximated values instead of just missing ones. The right way to fill the gaps, however, is heavily dependent on the situation:
   - Substitute missing values with dummy values such as 0 for numeric data, or other default values.
   - For numeric data, substitute missing data with dataset mean or median values.
   - For categorical values, use the most frequent categories.
   - Predict missing values based on other features.
7. Decompose data. In some way the opposite to data reduction, this step means that some values are too complex and decomposing them might capture relationships better. An example can be the retrieval of a day of the week from the date.
8. Rescale data. Often there is a need to rescale numeric values in a dataset, so that they do not overweight other features, or so that features are weighted in a right way. Several ways to rescale are the following:
   - Normalize to zero mean, unit variance (so-called standard scaling)
   - Normalize to a range (min–max normalization), often 0 to 1 (0–1 normalization)
   - Decimal scaling, or multiplying to some power of 10
   - Normalize the whole sample to the unit norm
   - Logarithmic normalization, used for data with skewed distributions

9. Discretize data. Some numeric data can work more efficiently once grouped into categories instead. Say, there is not much difference in heights of 167 cm and 169 cm, so these can be grouped. Another good example is binarization of the image, where all pixels are either made black or white.

3.2.4 Combine methods

Finally, the third approach goes as follows: several pre-collected datasets are taken to retrieve the necessary information, the manually collected data is added, and all the data is processed so that the format is uniform. It is sometimes faster than the fully manual data collection, provided that sources of data be known. If they are unknown, on the other hand, there is a risk that the process will consume just as much time as manual data collection.

3.2.5 Data generation

One more approach to finding a dataset is data generation, which can be automated or semi-automated, random, with given distribution, or conforming to specific rules. Bright sides of this method are that one does not need where to find data and only needs to know some rules that hold. Also, it is often easier and cheaper than field data collection so might be a reasonable option in some cases, especially if the suitable dataset does not exist. However, there are some areas in which data generation is quite tricky, like computer vision where data generation often boils down to 3D modeling. Many kinds of data, such as pictures, music, or texts, can not yet be generated in acceptable quality. Furthermore, this approach can quickly lead to dataset being not representative or too artificial.

3.3 Data preparation

As mentioned above, it is vital to prepare data adequately before they are feed to a model. The reason for that is this way we can decrease the training time, as well as resources needed for the process, or even increase the accuracy.

When working with a picture, there could be two fundamental approaches to processing it. One can either process the picture as a whole, or split it into regions of interest and process the work separately. The first approach allows one to work with context at a price of increased complexity, while the second one is simple, yet makes one sacrifice accuracy in many situations where context would be helpful.

Now, considering the predictive part, there are also two approaches to data input. The one is feeding the raw picture to a predictor, which would imply more time spent training and more time on prediction. Another one is extracting features fist to make them be an input for the model — this would in its turn require more coding efforts that might not necessarily lead to improvement.

Even if we decide to use raw pictures as predictive model input, it will still require some transformations we need to perform. We need to decide whether we get rid of color or not, apply appropriate scaling which is needed because many models require a constant number of features in an entry, as well as perform particular transformations at regions of interest, an example being perspective transformation.

3.4 Computer vision

Computer vision is a specialized approach to the creation of machines that aim to detect, track and classify objects on pictures or videos to understand the picture. The technology is a mix of image processing methods, pattern recognition, and attempts to represent the semantics of imagery, and is used for such tasks as detection, segmentation, localization, and recognition of specific objects in images, object tracking, scene reconstruction and content-based image retrieval. Many of these tasks are accomplished via operations on so-called ROI (regions of interest), which are superpositions of points and curves with specific properties. Now let us look at some key notions of computer vision:

- Bounding boxes are boxes of a predetermined shape that fully enclose objects that are being checked for collision, and ideally are minimal of all that hold this property.
- A contour is merely the boundary of an object in an image
- Then there are edges, ridges, interest points and blobs
  - Edges are usually one-dimensional and represent the boundaries between objects or just image regions
  - Interest points are essentially points or small clusters of points
  - Blobs are similar to interest points but are more region-like and less point-like.
  - Ridges are, however, more specific and serve more or less as axes for elongated objects

Another important notion to consider is the structural similarity index (SSIM). It is a method used to measure the similarity between two images, created for prediction of perceived quality of images on television. It was a big step from such techniques as mean square error (MSE) or peak signal-to-noise ratio (PSNR), the difference being that it is a perception-based model that takes into account biases of how people perceive the image, in terms of contrasts and colors.

Let us now say a few words about features, which are blocks of numerical information that, although at times looks as nonsense for humans, represents the most significant properties of a target object while also having a significantly

smaller dimensionality of properties than an object itself. We are going to cover features applicable to images.

There are two categories of features worth considering, namely local and global features. Local features are used to describe critical points of the image, while global features aim to generalize the entire object. Each task predictably has its specific requirements put on features, which is the reason there were created plenty of them.

One of the methods of feature extraction is Bag of Features, or BoF, which is essentially Bag of Words (BoW) translated onto image feature extraction. Bag of words is a sparse vector of the number of occurrences of each word of a document set dictionary in a document. Overall, BoF is a method that has more advantages than disadvantages despite not being very efficient.

Another widely used technique for getting a fixed size vector representation is the use of the Fischer vector, which is an improved version of Fischer kernel method and is pooling of image features into a vector representation. The technique has been used in image classification, showing quite decent results while having less dimensionality compared to other approaches.

One more standard feature descriptor is the histogram of oriented gradients (HOG), used mainly for object detection. The idea behind HOG is that one can use edge directions of gradient distributions to describe the local shape of an object efficiently. The feature vector is robust to photometric and geometric transformation, while sensitive to rotation, and is suitable for human detection, especially face detection, mainly through binary classification.

One could also use background subtraction, also known as foreground detection. It is a technique designed only to leave foreground objects for future processing. It is a popular and powerful method, with its principal problem being the determination of what background is. It can be determined by frame difference, running average, running Gaussian average, or Kernel Density Estimation.

More features to go, and now let us consider a classical algorithm for feature extraction called scale-invariant feature transform (SIFT), which is used in junction with local features. The descriptor helps to identify interest points and is quite robust due to containing way more information than is needed minimally. This property also makes it resource-demanding. SIFT is widely used in tasks of individual identification of wildlife, video tracking, gesture recognition, 3D modeling and object recognition.

For the sake of object recognition, there is another feature descriptor, called shape context. It takes into account distributions over relative positions, which makes it compact, robust and accurate. It has been empirically shown shape context is invariant to outliers, noise, and deformations, while also being able to provide rotational invariants when it is necessary. Shape context has been successfully used in tasks like trademark retrieval, 3D object recognition, silhouette similarity-based retrieval, and digit recognition.

Now, let us take a look at the GIST descriptor, which was initially proposed in 2001 [93]. It was created to be a low-dimensional representation of an image, while not requiring segmentation, and is notable for being incredibly compact

and fast to compute while holding plenty of valuable information. This fact makes GIST efficient for duplicate detection, scene alignment and scene categorization in databases containing millions of images.

Another kind of digital image features designed for object recognition is haar-like features, which consider parts of adjacent rectangular regions, which are assigned values, based on sums of all pixel intensities, to further subtract on region value from another. It is observed that certain objects will have specific haar-like features in some areas. Thus every Haar-like feature is a weak classifier. Due to specifics of the method, any Haar-like feature can be calculated in constant time.

Overall, there is a notably significant amount of methods for extracting features from images for different purposes, and thus only a fraction of them has been covered, more specifically, those features that were the most widely used for problems similar to ours.

Most of the features are robust to scaling, and all except the bag of features are more or less robust to noise. Only background subtraction is sensitive to both rotation and translation, and haar-like features are rotation-sensitive, so overall, one can easily find feature descriptors with needed levels of robustness.

3.5 Machine learning

Machine learning is a class of artificial intelligence methods that are notable because instead of straightforward problem solving, learning by observing solutions of a multitude of similar problems is exploited. In other words, the algorithm learns ways to solve problems in practice, hands-on, just like a human would, only much faster, say, hours instead of years. After the learning stage has passed, the algorithm becomes able to solve similar tasks, based on the learning experience. This fact also saves a programmer from the need to know how to solve a given task, as they leave the task of finding dependencies and patterns to an algorithm, and the only things they need to do is appropriately prepare data and adjust the model, that saves the time spent coding.

There are numerous machine learning algorithms, which can be classified in a variety of ways, like by availability of data, type of predicted data, or a principle of work of the algorithm. Say, there are clustering algorithms for unsupervised learning, when we do not have labels for datasets, or prediction algorithms for supervised or semi-supervised learning, where at least part of a dataset available is adequately labeled. On the other hand, there is the regression, which outputs numbers on a real scale, or classification, that assigns a label from a finite set of possible discrete labels. Then there is binary classification, where only two classes are considered, multiclass classification, where there are more than two classes, and multilabel classification, where each object can be assigned more than one label. Finally, algorithms can be based on linear equations (linear predictors), polynomes (polynomial predictors), trees (such as decision tree or boosting algorithms), or perceptrons (neural networks).

Now let us consider all the notions listed above in ordered fashion and dive

deeper into details.

3.5.1 Unsupervised vs. supervised

In practice, most of the machine learning one would run into is supervised learning. The point of this broad category of approaches is mapping a set of input variables (x) to an output variable (Y). One uses the algorithm to approximate the mapping function between this variables to provide an appropriate output to every input. Knowing correct answers, we can adjust the result iteratively and stop once acceptable level of performance is achieved.

There are two kinds of supervised learning:

- *Classification* — output is categorical. Typical use cases include mail sorting, and medical diagnostics, as well as object, pattern, speech and handwriting recognition.
- *Regression* — output is a real value. Examples are the prediction of prices or sizes.

Unsupervised learning, on the other hand, means that one only has input data (X) which is not labeled by output values in any way. The goal of this approach is to learn distribution or structure of data to extract more information and get useful insights, so, since there is no output data present, algorithms are left to themselves to discover patterns in datasets.

Just like with supervised learning, unsupervised learning can be further grouped into two kinds:

- *Clustering* — aims to discover natural groupings of data. Used virtually everywhere, for example in ecology to describe groupings of individuals inside a population, in physical geography to cluster physical properties, or for image segmentation.
- *Association* — aims to discover rules that describe large portions of the given data, in the form of relations between variables. Typical use cases are market basket analysis, malware detection and e-learning.

3.5.2 Classification vs. regression

Classification, as was said above, is the task of mapping a function (f) from input variables (X) to output variables (Y) with a discrete finite domain. Output variables, in this case, are often called labels, or categories.

Examples of applications of classification models:

- Medical image analysis
- Optical character recognition
- Video tracking

- Speech, handwriting and pattern recognition
- Geostatistics
- Credit scoring

By definition, classification is the systematic distribution of researched objects, phenomena, or processes, by sort, kind, type, any significant features, for the sake of research convenience; grouping of primary concepts and a particular sort of ordering, based on an extent of similarity. Alternatively, on the other hand, a set of objects ordered based on some principle, where objects have some classificational properties that are chosen to determine how similar or different they are.

Regression, on the other hand, is the task of mapping a function (f) from input variables (X) to continuous output variables (Y). Y can be integers or floating-point values, which often represent quantities. Because of that, the quality of prediction is a function of the error in prediction.

Some algorithms have the word regression in their names, but not all of them are regression algorithms. For instance, while linear regression is a regression algorithm indeed, logistic regression as in fact a classification algorithm.

Some algorithms can be used exclusively either for regression (e.g., linear regression) or classification (e.g., logistic regression), while other algorithms, such as decision trees or artificial neural networks, can be applied to both cases with minor modifications.

3.5.3 Binary classification

Binary or binomial classification is the task of classifying elements into precisely two groups, examples being:

- Medical testing whether a patient has a given disease
- Pass or Fail tests
- Deciding whether a page or an article should appear in search results

Most often, sizes of categories are not equal, and errors are skewed - that is, sometimes False positives are more critical, and sometimes False negatives are. This kind of classification is quite easy and needs simplest models. In fact, many machine learning methods are initially created for binary classification and then adapted to multiclass classification through one-versus-one or one-versus-all strategies. On the other hand, one can often need more labels to work with than just YES or NO.

3.5.4 Multiclass/multilabel classification

Multiclass classification as classification where there are more than two classes to sort items into. One item can belong to one class. This variation of classification

is more flexible, yet some classifiers are not naturally tailored for anything except binary classification, so clever workarounds are needed.

Let us now make one step forward in complexity and consider multilabel classification. Now, there are many categories, but any object can have multiple labels — even all at once. This feature is usually modeled with binary classifiers for each category that answer the question whether an object belongs to a given category or not. An example is an answer to the question What is in the picture?.

3.5.5 Probabilistic approach

Often, classification models do not just yield labels but instead predict continuous values that serve as probabilities of a given example belonging to each of the classes. These probabilities can be interpreted as confidence or likelihood of prediction. A predicted class is then assigned the class label with the highest probability. With binary classification, it is common to set another arbitrary threshold between classes. For example, if the prediction is whether a patient has cancer or not, the threshold can be as low as 10% which means that patient having more than 10% probability of cancer will not be classified as healthy. From this example, it is clear that such manipulations are done when the cost of type I error is different from the cost of type II error, and the threshold is chosen to minimize this cost.

The idea of probabilities yielded instead of bare labels leads us to another two important notions — discriminative and generative models.

Discriminative models, or conditional models, are those that model the dependence of target variables (Y) on observed variables (x). From the probabilistic point of view, the probability function P(y—x) is what is being modeled. This function can be used to predict Y given X. Unlike generative models, that are considered more thoroughly below, discriminative models do not model enough information to generate more samples. However, for tasks that do not require the joint distribution, such as classification and regression, discriminative models are often more preferable, due to superior performance explained mostly by the lesser amount of variables to compute. On the other hand, they lack the flexibility required for more complex tasks, and also only support supervised learning. Ultimately, the choice of generative versus discriminative model is dictated by needs in every specific application.

Now, let us talk about generative models — these are models that generate all values for a phenomenon, both target variables computed from observed data and values that can be observed. In other words, they generate both outputs and distributions of inputs, often given some hidden parameters. Generative models thus specify joint probability distributions over observations and for target values, given the possibility to yield all probability combinations that arise in a data set. These combinations are used either as an intermediate step for prediction or direct data modeling.

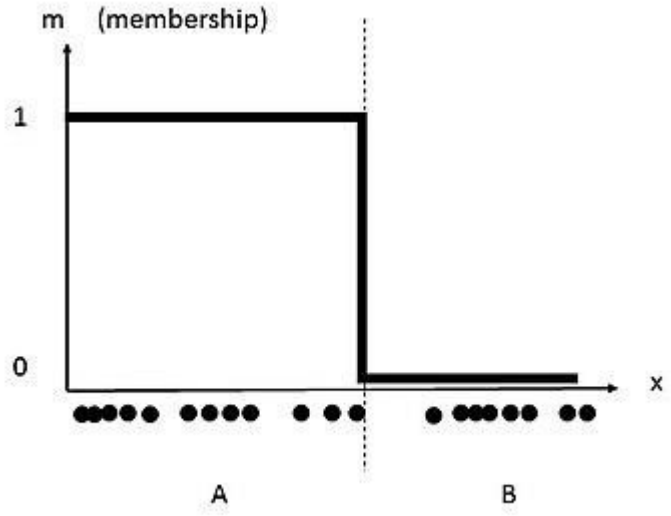
3.5.6 Clustering

As was mentioned above, clustering pretty much consists of dividing values into classes which are not predetermined, where the number of classes is often unknown in advance. There are two kinds of clustering — hard clustering and soft (fuzzy) clustering. Hard clustering — each object belongs to exactly one cluster. Soft clustering — each object belongs to a cluster to a certain degree (in other words, we have probabilities of belonging to a cluster) For more details, let us consider an example of monodimensional data (Figure 3.1):

Figure 3.1: Monodimensional data

Let us now split this dataset into two clusters. For that, a threshold is selected, and resulting clusters are labeled A and B. This way every data sample will either have a membership coefficient of 1 or 0. The graph of this coefficient is shown on Figure 3.2.

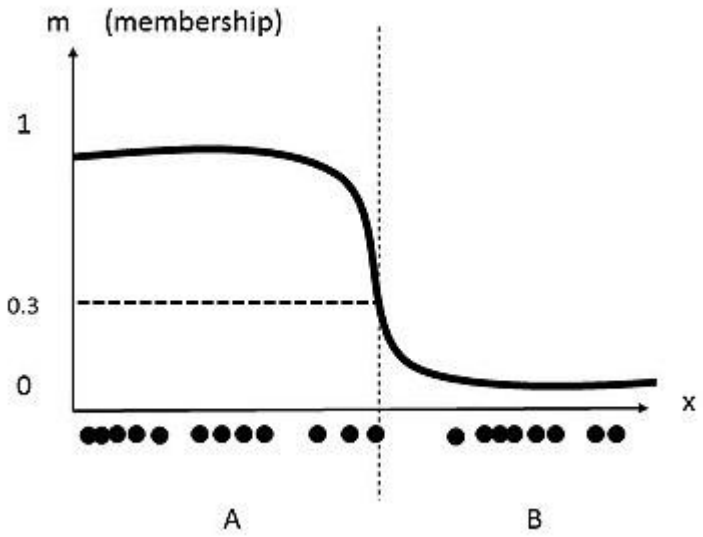
Figure 3.2: Hard clustering



In fuzzy clustering, on the other hand, each sample can be a member of multiple clusters, to an extent. The membership coefficient is thus a value in the range between 0 and 1. Now, on the Figure 3.3, a threshold is redefined, using

c-means fuzzy clustering. New membership coefficients are generated based on cluster centroids.

Figure 3.3: Fuzzy clustering



The point highlighted in the image, for example, has a membership coefficient of 0.3 with respect to cluster A. Thus fuzzy clustering is a kind of clustering where an object can belong to more than one cluster simultaneously, to an extent. That is, instead of cluster labels, objects are assigned coefficients for every cluster, such that sum of all coefficients is 1 for every object. For instance, fuzzy c-means is a fuzzy version of k-means.

Let us now consider a variety of clustering methods, such as k-means, k-median, and k-center.

The k-means is the most popular of them, created as early as the 1950s. The algorithm aims to minimize the sum of quadratic deviations of elements from cluster centroids:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{3.1}$$

In fact, k-means, k-median, and k-center are all very similar and follow the same pattern, to the extent that it is safe to say that they are all the same algorithm, only with slightly different functions to optimize. In each of them, k starting points are chosen as cluster center, and then all other points are assigned to the closest of the clusters. Then, centers are rearranged to minimize a particular function, that will be considered later. After that, a new check

performed on all object to reassign them to clusters they are closest to. The process is then continued until no object is assigned a different cluster compared to the previous iteration.

Now considering the function that we need to minimize, it is the very difference between these three algorithms. For k-means, it is a sum of squares of distances between a center and objects of a corresponding cluster, or L2 loss function. For k-medians, it is a sum of distances between a center and objects of a corresponding cluster, or L1 loss function. So finally, for k-center, it is the radius of every cluster or distance between the center and the cluster member located furthest from the center.

What is good or bad about these methods? Let us first start with disadvantages. The first issue is that Euclidean distance measures or any similar metrics will not most probably weight underlying factors by their importance, so at the very least, it is useful to use weighted versions of these metrics. Then, one will need to go through significantly more iterations if you want to achieve a slightly better result, that is sometimes even disappointing. Lastly, as with every clustering method described here, one needs to specify the number of clusters in advance (that is inconvenient and will probably later make us abolish these lousy methods and find something that fits our needs better).

3.5.7 Distance-based classifiers

For any two N-dimensional vectors, numeric or categorical, it is possible to define the distance. It is some function of two vectors(or points), which is numeric, non-negative, symmetric, defined in such a way that d(x, x)=0, and conforms to triangle inequality. In this case, the distance will be a measure of similarity, or rather, the dissimilarity between vectors. Common numeric distance formulas shown in the Table 3.1.

For categorical vectors the computation of similarity is more uniform, being

$$sim(x, z) = \sum_{d=1}^{D} sim(x^d, z^d) \tag{3.8}$$

with component similarity functions being as easy as 3.9

$$sim(x^d, z^d) = \|[x^d = z^d] \tag{3.9}$$

or as complex as 3.10

$$sim(x^d, z^d) = \begin{cases} 0, & x^d \neq z^d \\ K(p(x^d)) & x^d = z^d \text{ for some} \downarrow K(u) \end{cases} \tag{3.10}$$

where

Table 3.1: Common numeric distance formulas

Name	Formula	
Euclidean	$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{\sum_{i=1}^{D}(q_i - p_i)^2}$	(3.2)
L_p	$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt[p]{\sum_{i=1}^{D}(q_i - p_i)^p}$	(3.3)
L_∞	$\max_i =_{1,2,\ldots D} \mid p^i - q^i \mid$	(3.4)
L_1	$\sum_{i=1}^{D} \mid p^i - q^i \mid$	(3.5)
Canberra	$\frac{1}{D}\sum_{i=1}^{D} \frac{\mid p^i - q^i \mid}{\mid p^i + q^i \mid}$	(3.6)
Lance-Williams	$\frac{\sum_{i=1}^{D} \mid p^i - q^i \mid}{\sum_{i=1}^{D} \mid p^i + q^i \mid}$	(3.7)

$$K(p(x^d)) = \frac{1}{p(x^d)^2} \tag{3.11}$$

or

$$K(p(x^d)) = 1 - p(x^d)^2 \tag{3.12}$$

The link between similarity and distance can be easily defined, for example, as 3.13

$$sim(x_{num}, z_{num}) = F(\rho(x_{num}, z_{num})) \text{ for some } \downarrow F(u) \tag{3.13}$$

e.g.

$$sim(x_{num}, z_{num}) = \frac{1}{1 + \rho(x_{num}, z_{num})} \tag{3.14}$$

With distance being a measure of dissimilarity, it is natural to try to use distance-based algorithms for machine learning, using values for vectors close enough to determine the output for a given vector.

One of such algorithms is called K-Nearest Neighbors (K-NN). In this algorithm, for a given instance, one determines K training instances closest to one we are predicting. Then, voting or weighted voting among them is employed to determine the output.

KNN is simple to implement, fast to train, flexible due to the multitude of variations of both distances and weights, handles multiclass classification naturally, as well as regression, and with enough data, applicable quite well in practice. On the other hand, this algorithm is incredibly slow in prediction, being linear both in terms of training set size and vector length, consumes much memory since all training instances must be stored, and only works when it is possible to apply a meaningful distance function.

KNN is widely applied in the following areas:

- Concept search, i.e., searching for documents with similar topics
- Recommender systems
- Face recognition systems

3.5.8 Linear classifiers

In the field of machine learning, a common task to solve is to find a mapping between inputs and outputs. Linear machine learning models, thus, employ linear functions 3.15 as approximations of these mappings:

$$y = F(\vec{w}\,\vec{x}) = F(\sum_j w_j x_j) \tag{3.15}$$

Typically, linear classifiers can only solve linearly separable problems, i.e., such that there exists a hyperplane dividing the classes. However, even in non-linear cases, it is possible to make linear models work for the cause by introducing kernel functions to work instead of a conventional scalar product.

Linear classifiers are often used when prediction needs to be fast, due to them being faster than most other models while also having acceptable scores, especially with large-dimensional and sparse vectors.

3.5.9 Support Vector Machine

The first model for us to consider is Support Vector Machine (SVM). Unlike distance-based methods, it aims for the most optimal class boundaries and thus achieves better accuracy, while also being more complex, due to more information being taken into account. It is essentially a linear model with hinge loss and L2 regularization. This method aims to find such a class border hyperplane that it is maximally far from both classes. Usually, it is done by setting a condition that margin should always be no less than one, or in case of nonseparable classes, less than one minus specific constant that is also optimized. In addition to linear classification, SVM can also perform non-linearly with use of alternative kernel functions instead of standard scalar product, implicitly mapping input features to more dimensions. On 3.16 one can see how optimization task for SVM looks like.

$$\frac{1}{2C} \mid w \mid^2 + \sum_{i=1}^{N} [1 - M_i(w, w_0)]_+ \to \min_{w,\xi} \tag{3.16}$$

SVM is pretty widely used everywhere where machine learning is in use at all, and for some good reasons. First and foremost, it is the fastest existing method to find discriminative functions. Also, the method is fundamentally based on solving a quadratic programming problem in a convex area that implies there is always the only solution. So what is best, the method performs the search for the broadest possible band, which allows for straightforward and sure classification in the long run. On the other hand, there are certain drawbacks to consider, namely noise sensitivity, as well as the strong dependence of results on data standardization and normalization. Furthermore, there is no general approach to choosing a kernel function in cases where classes cannot be linearly discriminated, so whenever one faces such a situation, they need to choose a kernel manually and hope for the best. Finally, this method proves to be the slowest across all the linear models.

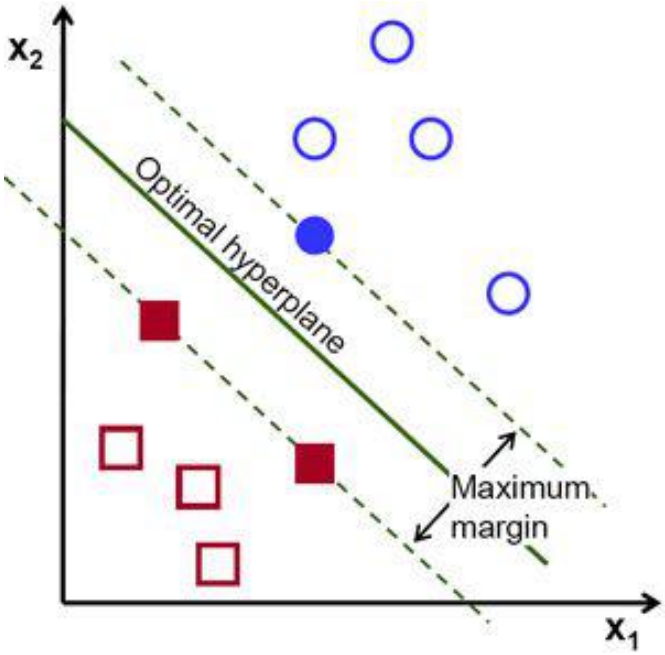
The most successful applications of SVM include:

- Face detection
- Text and hypertext classification
- Image classification
- Bioinformatics — protein classification, cancer classification
- Handwriting recognition

3.5.10 Logistic regression

Let us now consider another classification method, called logistic regression. It is especially suitable when the classification is binary and is a type of regression analysis. It is a predictive method, which means it outputs probabilities

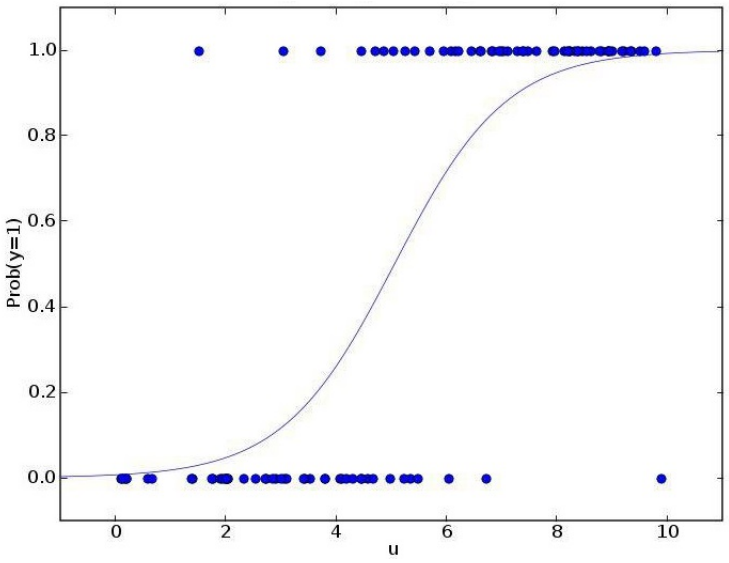
Figure 3.4: Support Vector Machine



of labels, not labels themselves, that has some particular advantages over discriminant analysis. In fact, there are so many advantages that they deserve a separate paragraph. To start with, feature values can freely be bounded or non-interval. No assumptions about features are made. As such, there is no assumption that error terms are normally distributed, or that variance is homogenous. Moreover, the outcomes do not need to be normally distributed. Furthermore, it is possible to add power terms or explicit interaction, and, as it could have already become apparent, nonlinear effects are also handled quite simply. In fact, linear relationship between features and outcomes is not assumed at all, and as a result, the method is strongly robust in the sense that equal variance for each group is not needed, as well as normal distribution for features, that does not have to be assumed.

With so many advantages, it might seem surprising that any other methods are used at all, but still, the main drawback for logistic regression is that it needs way more training data than discriminative methods even just to get halfway stable, let alone marginally meaningful, results. The lowest amount of data in a training set ought to be several times higher, and that is the cost of such a flexible instrument to use. On 3.17 one can see how optimization task for logistic regression looks like.

Figure 3.5: Logistic regression



$$\sum_{i}^{n} \ln(1 + e^{-\langle w, x_i \rangle y_i}) + \rightarrow \min_{w} \tag{3.17}$$

Most successful applications of logistic regression include:

- Image segmentation and categorization
- Geographic image processing
- Handwriting recognition
- Prediction whether a person is depressed or not based on the words they use

3.5.11 Naive bayes

Naive Bayes classifiers are the whole family of probabilistic classifiers based on Bayes theorem with strong independence assumptions made for the features. This kind of classifiers is around since as early as the 1950s and is extensively used for text categorization. With appropriate processing, it can beat or at least compete with SVMs and even more advanced methods in certain fields, including among all others medical diagnosis. This family of classifiers is highly scalable, with appropriate training performed in linear time, in one iteration. On the other hand, when features are not strongly independent, the method will fail miserably.

The classification function for Naive Bayes shown on 3.18.

$$\widehat{y}(x) = \arg\max_{w} p(y)p(x^1 \mid y)p(x^2 \mid y)...p(x^D \mid y) \tag{3.18}$$

3.5.12 Decision tree

A decision tree is a decision algorithm using a tree-like graph or model of decisions and possible consequences. For splitting the nodes, various decision rules are employed, such as categories, ranges or conditions, sometimes even multivariate conditions.

Decision trees are simple, interpretable, able to implicitly perform feature selection and handle all kinds of features naturally well, even being able to cope with missing data, which makes the algorithm quite valuable. On the other hand, these models are not entirely accurate, prone to overfitting and do not support reinforcement learning.

Use cases of decision trees are numerous:

- Star galaxy classification
- Nonlinear dynamic systems
- Medical diagnostics
- Amino acid sequences analysis
- Estimation of development effort for a given software module
- Medical text classification

3.5.13 Combinations

Sometimes, one may decide to combine multiple models into an ensemble, to either reduce the risk of overfitting or increase resultativity. There can be similar or different models, working consequently or in parallel, or with any other arrangement. Two popular methods of combining models are boosting and bagging.

Boosting is an algorithm for primarily reducing bias, as well as a family of algorithms to convert weak learners (slightly better than random) to strong ones. It has grown from a big question: is it possible to produce a strong learner from many weak learners?

The procedure is sequentially building an ensemble in such a way that the next algorithm corrects mistakes of already existing ensembles. This algorithm is thus greedy. For the last decade, boosting remains one of the most popular machine learning methods, along with neural networks and SVMs, due to simplicity, universality, flexibility and high generalizability.

Decision tree boosting, for example, proved to be one of the most efficient classification methods. Many experiments showed that error rate on the training

set was declining almost infinitely, with test set error rate continuing to decline long after the training set error rate fell to zero. It was a shocking discovery, overthrowing the common belief that too complex models always reduce the quality of prediction. The phenomenon was later theoretically explained.

Bagging, also called bootstrap aggregating, on the other hand, is an ensemble meta-algorithm used to prevent overfitting by reducing variance, as well as to achieve stability in results. It is a special case of model averaging approach — all models are trained independently, with the result being determined by voting. The idea is that classifiers do not correct each other, but instead compensate each other in voting. The base classifiers must either be based on different methods or be trained on different sets, to be independent.

3.5.14 Random forest

Let us now talk about a specific example of an ensemble, called a random forest. The catch is to construct a multitude of decision trees on training randomly and output the mode of classes for classification, or an average for regression. Randomness here exists to correct decision trees proneness for overfitting.

Random forest helps to overcome many problems that exist with decision trees. The overfitting is reduced due to averaging, the variance is decreased, and thus random forests almost always perform better than single decision trees.

On the other hand, disadvantages are indeed numerous. Random forest is more complex, less interpretable, more challenging to implement, and more computationally expensive. Sometimes, reduced risk of overfitting compensates for all these drawbacks. Applications for random forests in computer vision include:

- body part classification
- head pose estimation
- pedestrian detection
- body pose estimation

3.5.15 Neural networks

Artificial neural networks (ANNs) are systems inspired by biological neural networks animal brains consist of. They are an advanced method to avoid task-specific programming. An ANN consists of connected nodes, also called artificial neurons. Each connection can transfer information between nodes. A node can then process the signal and send it further across the network through connections.

Most commonly, a signal is a real number, and the output of each neuron is a function of its inputs (or commonly, a function of a sum of its inputs). Typically, nodes and connections also have weights that adjust during the learning process, which are responsible for the strength of the signal. Nodes can also have a

threshold that cuts all inputs lower than a particular value. Typically nodes are arranged into layers, where each layer is dedicated to one kind of transformation.

Initially, the goal of the ANN approach was to solve problems in a way similar to how a human would solve them. However, soon the focus shifted to more specific tasks, thus forging the drift away from biological similarity. Now, ANNs are used in a wide variety of ways, including medical diagnosis, games, machine translation, speech recognition, and computer vision.

3.5.16 Neural network building blocks

For ANNs, three key components matter:

- Network Topology
- Adjustment of weights or Learning
- Activation functions

Let us now look at each of these components in detail.

A network topology is the arrangement of nodes of a network, with layers and connecting lines. By topology, several kinds of ANNs can be outlined, such as

- Feedforward networks
- Feedback networks

Let us first look at feedforward networks — this is a kind of non-recurrent network that can be divided into layers, with nodes in a layer being connected to nodes of the previous layers, with different weights upon them. The signal here only flows in one direction, from input to output. Simplest feed-forward networks are single-layer, which means they only have the input layer fully connected to the output layer, with nothing else in between (Ex. on Figure 3.6).

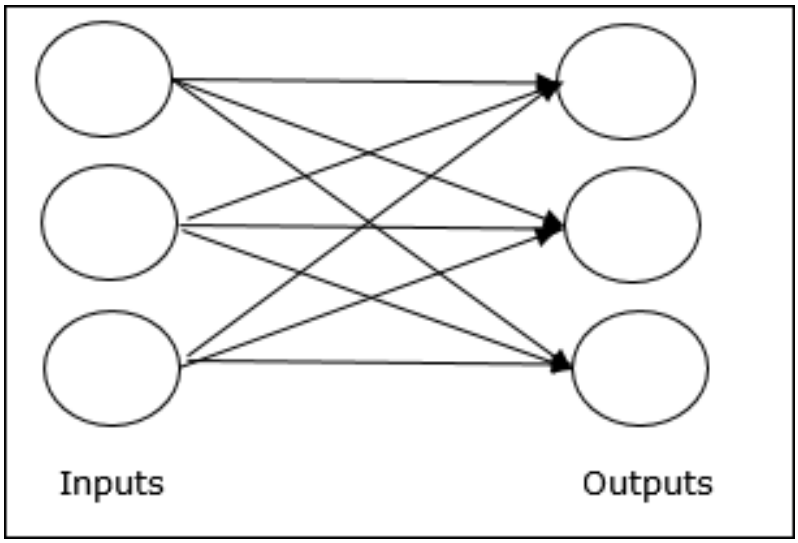
Multilayer networks (Ex. on Figure 3.7), however, have some number of hidden layers in between, differently weighted, so are able to process more complex data.

Now, let us look at feedback networks, which have some feedback paths, allowing the signal flow in both directions using loops, which makes a network non-linear dynamic system, which continues forcing the flow until equilibrium is reached.

For example, recurrent networks are feedback networks with closed loops (Ex. on Figure 3.8). Two well-known types of recurrent networks are fully recurrent networks and Jordan networks. Fully recurrent networks are quite simple in architecture, as every node is connected to every node, with all nodes working both as input and as output.

With Jordan networks, the matters are even simpler than that, with outputs going back to inputs as feedback (Ex. on Figure 3.9).

Figure 3.6: Example of single-layer feed-forward network



After the network architecture is finalized, it is time to think about adjustments of weights, also called learning. It is the method of modifying the weights of connections between the network neurons. One can distinguish supervised learning, unsupervised learning and reinforcement learning.

With supervised and unsupervised learning, the situation is just as straightforward as for any other predictor. With reinforcement learning, however, it all becomes more interesting (Ex. on Figure 3.10). It is the learning used on a pre-trained network to strengthen it over some critical information. It is similar to supervised learning but requires a pre-trained model, yet training sets are way less in this case.

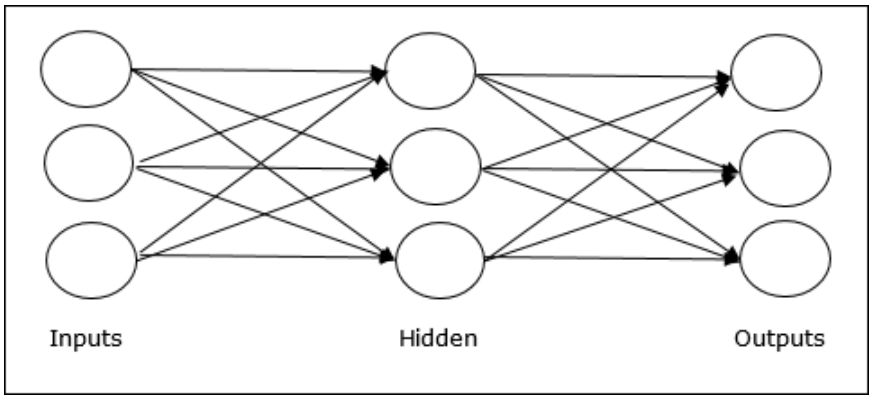
The network receives some feedback from the environment, yet the feedback is evaluative, not instructive. After that, weights are adjusted further.

Let us now move on to activation functions — the last critical component of neural networks. They can be defined as extra efforts required to convert the input to output, or functions performed on input, and are applied to nodes, not a network as a whole. The simplest of them is a linear activation function or identity function 3.19

$$F(x) = x \tag{3.19}$$

Other popular activation functions include sigmoids, which can be binary or bipolar. Binary sigmoidal function renders the output to be between 0 and 1, and is strictly increasing 3.20

Figure 3.7: Example of multilayer feed-forward network



$$F(x) = sigm(x) = \frac{1}{1 + exp(-x)} \tag{3.20}$$

Bipolar sigmoidal function, on the other hand, fits the output into (-1; 1) range, while also being strictly increasing 3.21

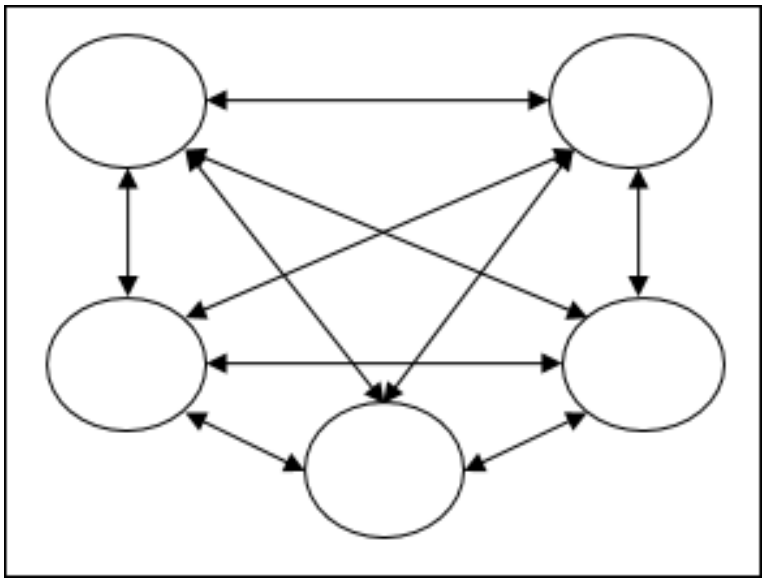
$$F(x) = sigm(x) = \frac{1 - exp(x)}{1 + exp(x)} \tag{3.21}$$

3.5.17 ANN applications

Here is a longer (but in no way exhaustive) list of successful ANN applications:

- black-box models in geoscience
- cancer diagnostics
- email spam filtering
- machine translation
- data mining
- automated trading systems
- gesture, speech, handwritten and printed text recognition
- object recognition
- face identification
- radar systems

Figure 3.8: Example of feedback recurrent network



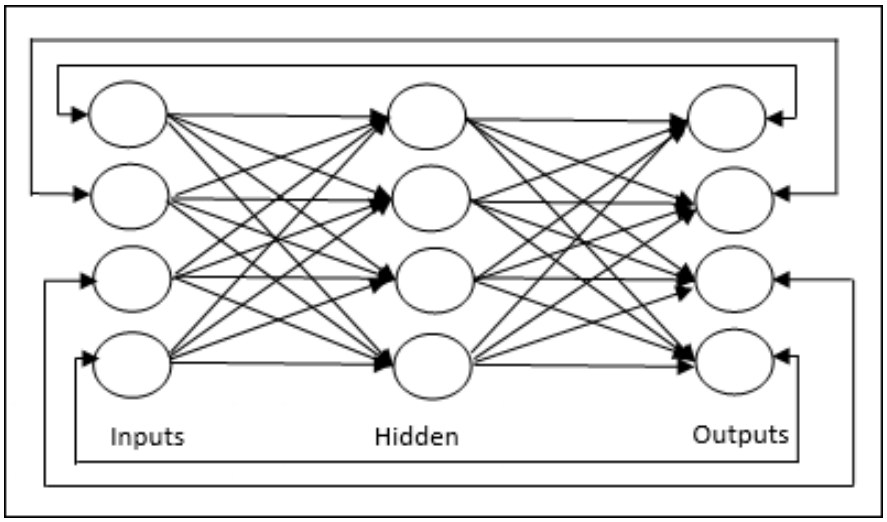
- game-playing and decision-making
- quantum chemistry
- trajectory prediction

With current rate of progress, the actual areas of use seem to be going to start expanding almost exponentially.

3.5.18 Convolutional neural networks

Let us now talk about one particular kind of ANNs. Convolutional neural networks (CNNs) are deep, feed-forward ANNs, which have been successfully applied to the field of computer vision. CNNs use a particular kind of multilayer perceptrons designed in a way that would require minimal data preprocessing, and make a network naturally invariant to translation. CNNs were inspired by principles of information processing in the animal visual cortex. Individual neurons only react to stimuli in a restricted area, but such areas for different neurons overlap, together constituting the whole visual field. A good advantage of CNNs in computer vision is that these networks learn the filters that were explicitly programmed in previous algorithms, making them independent of a lot of efforts and prior knowledge. Aside from visual recognition tasks, CNNs proved to be applicable to natural language processing and recommender systems.

Figure 3.9: Example of Jordan networks



3.5.19 Region-based CNNs

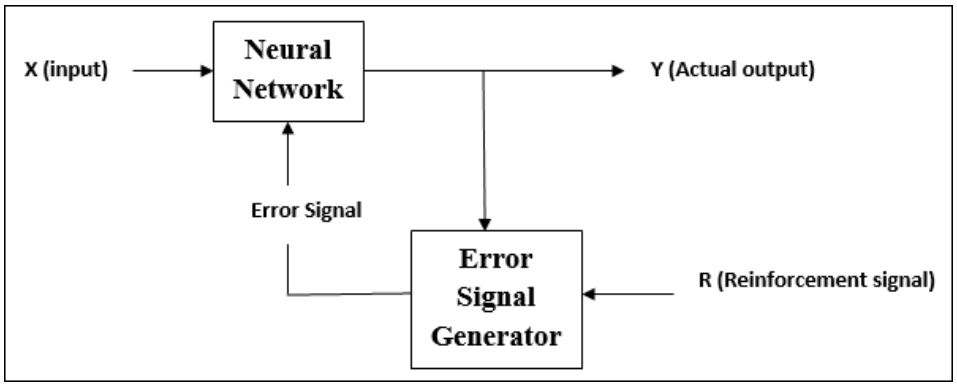
Now, we are going to discuss a well-known family of CNNs, namely R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN, all recently created by a group of researchers over the course of several years.

The R-CNN takes an image as input, correctly outputting bounding boxes and labels for all principal objects in the image. The way it makes the prediction is putting plenty of boxes into an image and seeing if any of them contain an object. This process is called region proposal, which is done through the method called Selective Search — taking windows of different sizes, traversing the image through them, and then trying to identify objects by grouping pixels by some parameters.

After proposal creation, the region shape is reformed into a square of a standard size, to pass through a modified variation of AlexNet. In the final layer, an SVM starts working to classify an object. After the object is found, the final task is to tighten the bounding box to reproduce actual dimensions of an object. For that, a simple linear regression is run on the region proposal to produce coordinates. While R-CNN achieves significant accuracy, it is also painfully slow, both since every single region proposal has to be passed through AlexNet, making up several thousand passes per picture, and also the need to train all models separately and then arrange them into a working pipeline.

In 2015, the problem of speed was finally solved, yielding a new model called Fast R-CNN. The catch is, most of the proposed regions overlap significantly, so computations are repeated for hundreds of times. Thus there should be a way to speed up the computations by only making one pass of the whole picture to the CNN. This way was found soon, and called Region of Interest Pooling

Figure 3.10: Example of reinforcement learning network



(RoIPool) – now for each region proposal, only specific features needed to be taken out of the CNN output.

Moreover, the Fast R-CNN was trained jointly, as if it was a single solid model, or to be more exact, started using the single network for all three tasks – feature extraction, classification, and bounding box regression.

Still, even with these improvements, the region proposal has become a bottleneck, due to Selective Search being quite slow. Soon, a team from Microsoft Research found a way to make this step nearly cost-free, in the next generation of the system called Faster R-CNN. Because region proposals depended on image features output by the CNN, researchers decided to use these result for region proposals instead of employing a separate algorithm.

The next step was performed just recently by a team of researchers from Facebook AI that included the original author. The new architecture, named Mask R-CNN, allowed solving the problem of image segmentation instead of just extracting bounding boxes. The Faster R-CNN got modified with an auxiliary branch which outputs a mask showing whether a pixel belongs to an object or not. The branch was just a Fully Convolutional Network on the top of the feature map output by the CNN. For now, this family of algorithms is a huge breakthrough in the field of object recognition.

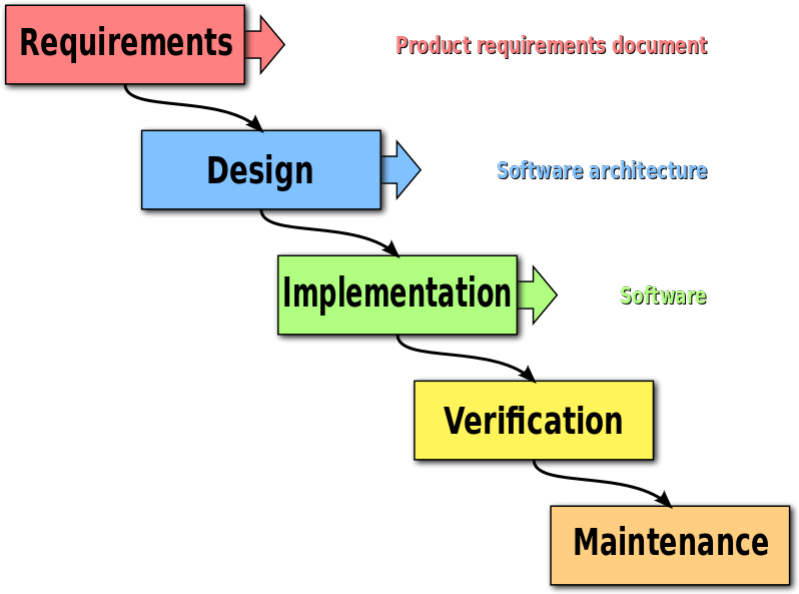
3.6 Software development techniques

Since the task for this thesis project is to develop a system with some functionality, there is an urgent need to consider different software development methodologies to be confident regarding what techniques we are going to use and why.

3.6.1 Waterfall model

The first one, and the earliest methodology stemmed directly from engineering (namely in manufacturing and construction), is the waterfall shown on Figure 3.11. The model is linear, meaning that all stages are passed consequently and not revisited.

Figure 3.11: Waterfall model



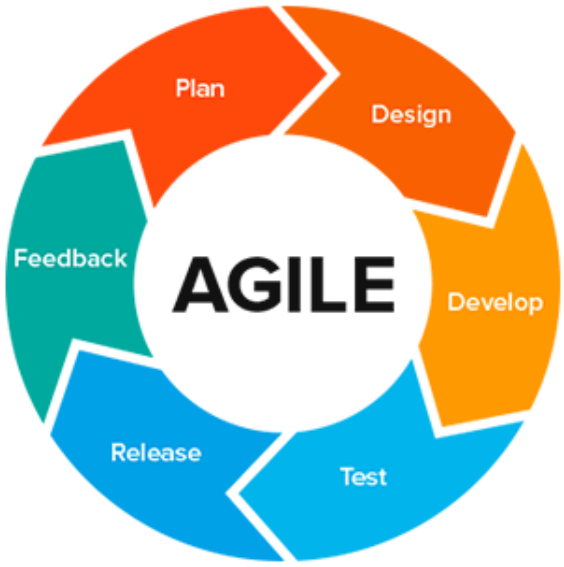
The logic is that time spent early on specification and planning can reduce costs at the later stages, along with the time spent coding. Thus the approach is about going through each stage thoroughly, with even large projects having very detailed specifications. Another argument for the use of the waterfall model is a heavy emphasis on documentation, as well as source code, which allows for easy developer interchangeability. It is also easily understandable, highly structured and provides identifiable milestones. This fact makes it perfect for cases where requirements are fixed, the end product is stable, and employed technologies are well-known.

However, it is not very practical when it comes to most of the products developed in the industry. Most of the time, clients tend to dynamically change the requirements after seeing some version of the product, leading to substantial redesign and so on, which implies a drastic cost increase. Furthermore, some details of constraints and requirements can only be discovered in the process of coding or testing, making the team to redesign the system again. These facts led to further modifications of the approach, as well as to the rise of entirely new methodologies.

3.6.2 Agile software development

A relatively newer software development methodology, Agile is nowadays one of the most popular approaches to software engineering. Within this technique, requirement and solutions are expected to continuously evolve through collaboration, with such features as adaptive planning, evolutionary development, early delivery, and continual improvement, so that all changes are addressed urgently.

Figure 3.12: Agile software development



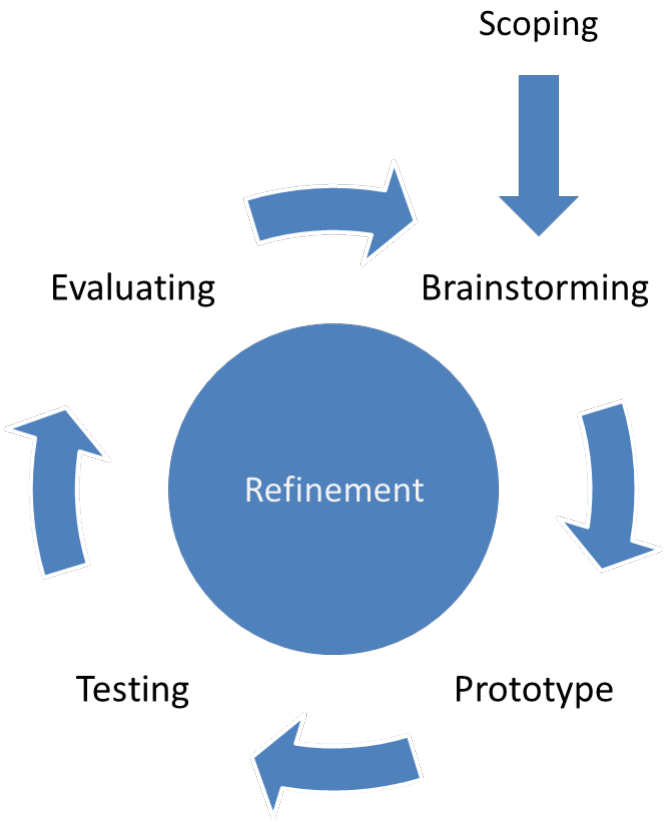
The best parts of agile software development are the acceptance of uncertainty existing in requirements and issues, faster review cycles leading to the better product, greater flexibility in releasing features, and less upfront work implying no more need to produce lengthy documentation while developers are doing nothing. However, everything comes at a price, and the price here is that agile principles are often misunderstood leading to pure chaos. Examples are dysfunctional choices made with no reference to check against, the need for specific organizational structure allowing for deep interaction, lack of predictability which might lead to the loss of strategic vision, the inherent difficulty of the creation of entirely cross-functional teams, as well as reduced scalability of the approach.

While anecdotal evidence strongly suggests that agile indeed helps to produce better products, some empirical studies have not found such connection.

3.6.3 Software prototyping

Software prototyping is about creating prototypes of software applications, or incomplete versions of software to be produced, as the name would suggest. It is similar to prototyping in such fields as manufacturing and mechanical engineering. A prototype need not resemble the final product, as its purpose is to simulate only certain aspects of the software being developed.

Figure 3.13: Software prototyping



Prototyping as a valuable practice in a sense that it allows to obtain useful feedback on the early stages of product development. The prototyped parts can be compared to specification so that changes are rapidly made if deemed necessary. It also allows for insights regarding project planning and time estimates, so that milestones are set realistically. Nevertheless, the convenience of the method is dimmed by increased development time and costs unbearable by tight budget teams, and reduced freedom for software designed stemming from the architecture of the prototype.

3.6.4 Iterative development

Iterative development is about breaking down the development of a system into small chunks so that the project is designed, developed and tested cyclically. In every iteration, new features can be added, until the fully functional software is ready to be released.

Iterative development is most suitable for cases when clear requirements are present, major requirements are known from the beginning while minor ones can evolve with time, the time-to-market is limited, an innovative technology not yet learned by the team is involved, and goals can be changed over time.

Experience shows that iterative development is quite a useful technique. It allows spotting potential defects early, to have functional working versions on early development stages. It implies more time on design and less on documentation, easy progress measurement, more straightforward implementation of changes, facilitation of testing, easy identification of risks. Iterations can be mapped to milestones, a new version of a product delivered on any iteration, operating time is reduced. Still, drawbacks are also numerous. More resources are required. Successive iterations cannot be overlapped, so each one is a miniature waterfall; project management can become more intensive, and the end date is often hard to predict. Risk analysis here becomes complex enough to require a highly skilled professional.

3.6.5 Pair programming

While not being a software development methodology in and of itself, pair programming is an agile software development technique where two programmers work at the same code at the same time at the same workstation. One programmer, called the driver, is writing the code, while another performs the line-by-line review, both switching roles frequently. The observers role is also to consider the strategic view on the work, proposing improvements and highlighting future problems to solve. The intention is to let the driver free to concentrate on more tactical issues for the current task.

Advantages of the method include two points of view, natural catching of small accidental mistakes, better concentration, the possibility for knowledge combination, and development of soft skills in the process. On the other hand, it is not the best method in a case of skill disparity, can quickly turn into socializing sessions and also problems can arise when both developers are experienced and are pushing their visions of how the work should be done.

3.7 Conclusion

In the end, we decided to try and make a convolutional neural network as simple as possible, and look whether it suits our needs since CNNs are used most widely for our cause. Nevertheless, but most of the conventional architectures are incredibly complicated, yet we need to produce something that will work fast and not consume many resources.

For comparison, we decided to take HOG + SVM combination due to it being prevalent in cases most similar to our problem. Also, we decided to try some existing architectures of CNNs if there is time left.

Also, we decided to work with picture fragments, to improve performance.

Finally, considering the software development methodology, the decision was to combine prototyping and iterative development, since this combination is the most convenient for this work. This way, we split the work into tiny parts, share them, implement them, and after that — combine them into something whole, which is a prototype of a system we are trying to build. Then it is evaluated, all advantages and disadvantages found and noted, all errors and bug are fixed, and the new cycle is started.

We also decided that my colleague would be going to do all the necessary preprocessing, while my tasks would be more directly related to machine learning.

Chapter 4

Implementation

With what we found out during the preliminary research, and extensive knowledge obtained while digging deeper into concepts, now we were ready to start implementation, and here we are going to explain how the process went, what considerations were there, and what problems occurred.

We are going to cover the system requirements first, then move on to high-level system architecture and related considerations, including development plan and programming language choice. After that, we are going to dive deep into details, including data collection, data transformation, prediction, I/O and other parts. Finally, we are going to discuss some strategies for system evaluation, that were used to obtain data presented in the next chapter.

4.1 System requirements

Extensive discussion on the subject of system requirements, along with practical considerations and issues related to lack of resources, led us to the following set of requirements:

- The system should be able to capture video from files
- The system should be able to capture live video streams
- The system should capture frames from videos
- The system should split a frame into slots according to a markdown configuration file
- The system should predict the status of the given slot with an acceptable degree of correctness
- The system should output predictions in a format both human-readable and computer-readable
- The system should save the output in log files

- There should be one log file for each frame
- The log file should contain enough information to reproduce the logged state on the picture, given markdown configuration.
- The system should be multiplatform
- The time to process one frame must not exceed 2 seconds
- The system should be lightweight in terms of consumed memory and computation resources, where lightweight means that an average modern computer should be able to also handle a couple of typical applications like a web-browser, while running the system, without lags in performance.
- The system should be easy to install, launch and run
- The system should be able to show the output visually in real time
- The system should be easily extendible, with additional tools, and modifiable with other predictive models or similar improvements

As one could notice, nothing is said about the origin of configs here. This issue is also going to be discussed, along with an auxiliary tool produced specifically to make parking lot markdown configs.

There were also other requirements that were considered but thrown out:

- The system should determine whether the situation is typical or somehow strange
- The system should have convenient UI for interaction
- The system should be able to improve prediction based on the manual sorting of an unrecognized situation and relearning

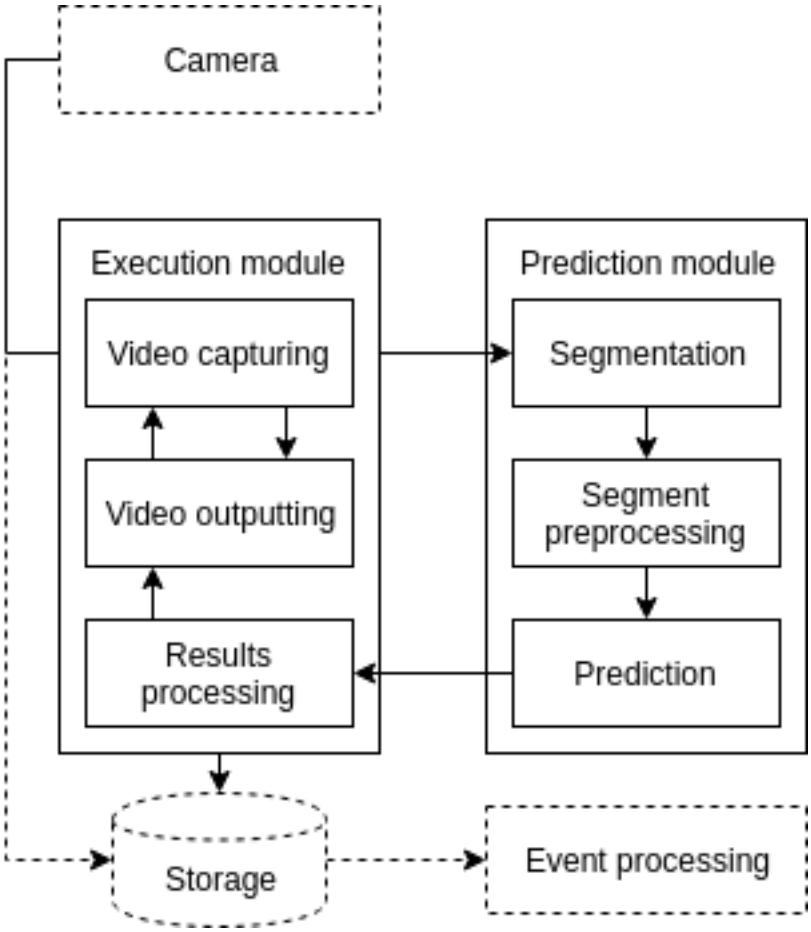
All three were thrown out because of time constraints and drastic increase in complexity of implementation.

4.2 High-level system architecture

Out of requirements listed above, a notion of the high-level architecture of our future tool emerged (Figure 4.1). It is inherently modular, consisting of an execution module and a prediction module, which should communicate with each other.

The execution module would capture a video stream from a camera and send it to a prediction module, while also outputting it on the screen with current labels visualized and continuing video capturing. After the prediction is made, it would update the labels and save them into a log, and then the cycle would start again.

Figure 4.1: High-level system architecture



The prediction module cycle has, on the other hand, much simpler work cycle. It accepts a frame from the execution module, retrieves parking slots from this frame, preprocesses the slots, predicts their status, and outputs prediction back to the execution module, waiting for the next frame to predict on.

4.2.1 Justification of modular architecture

Why did we make the structure modular? This architecture has plenty of advantages, to the extent that it is the recommended option nowadays. The good sides of this approach include:

- Fewer bugs and easier testing, since parts are decoupled
- More natural understanding of internal structure and algorithms, due to

semantically appropriate structurization

- The work can be divided more efficiently, with every developer implementing their part (we did not, though)
- Allows interchangeability of components and easy extensibility

All these features allow saving of time to develop at a price of overhead on more thorough documentation and integration testing.

4.2.2 Essential and additional components

As one could notice, some lines on the diagram are dotted. They are to represent the components we are not responsible for. We cannot guarantee an appropriate camera and storage — this needs to be done by a user. Other developers might also ensure saving of the video stream to storage. Another useful feature would be an event processing module that would parse logs from storage and use them to determine the behavior of other arbitrary devices, thus opening up more space for improvement of the smart parking.

4.2.3 The issue of parallelism

As could be already clear from the diagram and the explanation of it, to work in real time, and to handle video streams from a camera at all, the system should do two actions in parallel, the first being prediction, and the second — drawing and skipping frames while awaiting the prediction. These activities should happen at the same time, which makes a one-thread implementation unviable. Possible ways to handle the issue include:

- Use of asynchronous primitives. This is a solution commonly used in industry. However, some of the libraries that are possible to use to accomplish our goal can be incompatible with the approach.
- Implement modules as separate programs that interact with each other using pipes or queues. An older approach, which is also harder to handle, but which allows circumventing the library restrictions on asynchronous programming.

We tried to use both over the course of development.

4.3 Development plan

This work is in fact quite complex, consisting both of development part, where we need to create a new system from scratch, and research part, where we need to compare it to other existing methods to verify its practical applicability and overall quality. Thus, our plan consists of both research and development parts.

So here is the plan itself:

1. Train SVM model with use of HOG for the sake of further comparison
   (a) Train the network on existing data, get the result
   (b) Train the network on our generated data, get the result
   (c) Choose the model which works better and decide on the size of an appropriate dataset to retrain on
   (d) Retrain the chosen variation on a more prominent segment of the respective dataset
2. Try to use YOLO and (if time allows) other existing computer vision solutions
   (a) Try to use a pre-trained solution and look how it performs on our data
   (b) Retrain the network on a dataset customized for our needs and check on existing parking data we have
   (c) If it works, find more data and train the model on a better dataset
3. Develop the target system
   (a) Make and train a simple CNN
   (b) Implement functionality to cut segments out of frames
   (c) Implement the architectural (integrational) part
   (d) Add optimizations
   (e) Add support for video
   (f) Parallelize
   (g) Repeatedly refactor
4. Test all the models we have, compare them in terms of performance and predictive quality, and put the best model into the system

At the same time, we would have to develop an image labeling tool to produce config, which we are going to cover later. This tool is the reason why we have a notion of our generated data in this plan.

4.4 Justification for programming language

For the system we were creating, there were three choices of programming language:

- C
- C++
- Python

A choice that small is explainable by the fact that these are the languages for which the libraries we would need for the project exist to the fullest. Most other languages either do not have appropriate frameworks for computer vision and machine learning, or only have restricted APIs or poor documentation for these frameworks.

Thus, we decided to stop on Python, due to the following reasons:

- As developers, we have the most experience with this language.
- It is well-suited for prototyping.
- Has an incredible amount of data science frameworks
- More human-readable
- Cross-platform interpretable
- Safer than C/C++ due to lack of direct work with memory
- Extensible in C/C++

These features are especially valuable due to rigid time constraints put on us, which means we would not be able to test every pitfall we might otherwise have run into thoroughly.

However, we are also aware of the language disadvantages:

- Slow, which makes optimization an important consideration
- Harder to test and debug than most of the statically typed languages
- Issues with memory usage, which are manually unsolvable

Due to these facts, it is proposed that possible future development teams that have more time at hand rewrite the system in C++ or similar, thus potentially increasing the processing speed.

4.5 Data collection

Now that we have gone through basic high-level considerations let us concentrate at specific parts of the work, starting with data collection, as it is something to be performed first since details of system implementation might well depend on the kind of data we end up with. Let us go through a couple of things we are going to discuss in detail in this section.We are going to get through requirements for data, then consider existing datasets, and finally consider the work with video datasets.

4.5.1 Overview of existing picture databases

As the primary data collection strategy to employ was chosen to be existing dataset usage, let us look at the most notable picture datasets we ran into over the course of the search.

PASCAL VOC 2012

A general-purpose image dataset for the prediction of 20 object classes, divided into four categories:

- Person: *person*
- Animal: *bird, cat, cow, dog, horse, sheep*
- Vehicle: *airplane, bicycle, boat, bus, car, motorbike, train*
- Indoor: *bottle, chair, dining table, potted plant, sofa, tv/monitor*

The dataset contains:

- 11530 images
- 27450 regions of interest
- 6929 segmentations

The dataset was updated annually over the course of many years and primarily exists as a dataset for computer vision challenges. The goal of the challenge is mostly to recognize objects from twenty preselected classes and classify them appropriately.

This dataset, although employed at some stage, ended up being not suitable to our needs due to its general nature.

Microsoft COCO: Common Objects in Context

A large general purpose image dataset created for the wider variety of tasks as compared to Pascal VOC, including object detection, segmentation and captioning. Includes such special features as:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation

The dataset contains:

- 330 thousand images
- Over 200 thousand of labeled images
- 1.5 million objects

Figure 4.2: Example of image from PASCAL VOC 2012 dataset

- 80 object categories
- 91 stuff categories for superpixel segmentation
- 5 captions per image

This dataset has undoubtedly been much work and was also briefly used at one stage of our research, yet in the end, it turns out it is not suitable for our work due to its generality, while we have a more specific purpose.

The PKLot database

An open-source database which containing pictures captured from three cameras on two different parking lots over an extended period of sunny, cloudy and rainy days. The dataset contains:

- 12417 images of parking lots (1280X720)
- 695900 images of parking spaces cut from the images of parking lots (of good and bad quality, examples at Figure 4.4)
- XML files providing markdown and labels for slots on all pictures

Parking lot images are organized into three directories (parking 1a, parking 1b, and parking2). Each directory contains three subdirectories for different weather conditions (cloudy, rainy and sunny). Inside of each subdirectory, the

Figure 4.3: Example of image Microsoft COCO dataset

images are organized by acquisition date. It is the dataset that proved to be the most useful for our research. Both frames and segments were extensively used for the development — spaces were our first training dataset.

Nevertheless, frames further proved to be more useful than segments, since we were not able to find the segmentation tool used to produce this dataset, so we needed to implement our own. The pictures from our output were very different and thus incompatible to those present in the dataset. So later we used frames for checks and testing while generating our segments from frames to train models.

Another issue with this dataset is, it is Brazilian and thus does not contain snowy weather entries which would help improve the prediction in many countries, including Russia.

4.5.2 Video datasets

Unfortunately, we were not able to find any comprehensive video datasets which suite our needs. There were some videos concentrated on pedestrians that were not labeled for cars at all.

Still, we managed to obtain one useful video, 2 minutes long, which we used only for testing purposes. In fact, there are plenty of similar videos out there on the internet, yet they cannot precisely be named datasets since mostly they are

Figure 4.4: Example of segments from PKLot

(a) Good **(b)** Bad

demonstrations of how existing tools work, and do not have any usable labels but instead have labels as an overlay onto a video which is impossible to get rid of. Thus, we tried to filter those videos so that only those that dont prevent the correct work of our system are used.

The video we obtained is 2 minutes 1 second long, made during a sunny day. There are small circles in the middle of each slot, which change colors depending on whether there is a car or not, but this should not influence the quality of prediction, as we predict on grayscale images, and both colors look the same in grey.

4.5.3 Pictures vs. Videos

Work with video is pretty much similar to work with pictures. With videos, it is enough to capture some frames to work with them just like we would with pictures. With video streams, that would mean skipping frames until the prediction is made. Otherwise, it is pretty straightforward.

However, it is not like that when it comes to search for a dataset. It is notable hard to find a useful video dataset on the internet, which makes manual data collection a more suitable option. Furthermore, there is no such thing as video stream dataset, so one only can test work with streams during real-life test runs.

4.6 Data preparations

4.6.1 Config format considerations

XML (Extensible Markup Language) is a markup language that defines the rules for document markup in a format that is both human-readable and

Figure 4.5: Example of image from PKLot database

machine-readable. It is a textual data format which supports a multitude of human languages, used both for document markup and for object representation in programming.

XML is a basis for many other data formats, including office documents, pictures, and formats for web data transfer. The main reason why XML is an excellent choice for markup and configuration is that it is universal and extensible, thus allowing to possibly store everything in one place and create even insanely complex hierarchies of data. On the other hand, it is quite dense and wordy and requires traversing the tree to obtain its elements, which makes coding harder and slower.

XML was used as markup for PKLot (Ex. on Figure 4.6), as well as for PASCAL VOC annotation, and this was the reason we started with this format for configs. On the first iteration, we parsed parking lot coordinates for PKLot from its XML markup, which made for quite a lot of code in terms of line count, which is to be further debugged, tested and possibly maintained, and possibly makes the implementation slower, which made us think of alternatives quite soon.

JSON (JavaScript Object Notation) is a format to serialize objects, in both machine-readable and human-readable way. Created first as conforming to JavaScript type system, now the format is mostly language-independent since most of the modern languages include the functionality for JSON serialization and deserialization.

Although JSON supports only a few data types, and the format is not entirely standardized, it also has a much simple syntax as compared to XML and is natively supported by a multitude of languages including Python. Thus it is possible to deserialize data and work with it as with native Python objects.

Figure 4.6: Example of XML file from dataset (part)

```
- <space id="3" occupied="0">
  - <rotatedRect>
      <center y="208" x="366"/>
      <size h="32" w="52"/>
      <angle d="-77"/>
    </rotatedRect>
  - <contour>
      <point y="185" x="355"/>
      <point y="186" x="388"/>
      <point y="233" x="374"/>
      <point y="230" x="345"/>
    </contour>
</space>
```

These considerations made the format our choice for both configuration (markup) files and prediction log files, as a reliable and straightforward format that is easy to use.

Of course, there were thoughts on other options for configuration files and logs. For instance, what if it was **TOML (Tom's Obvious, Minimal Language)**? It is even more human-readable and straightforward. However, on the other hand, it is not ingrained into programming languages as heavily as JSON, and furthermore, the kind of information we are including would not make TOML files any more understandable than JSON. We are bound to work with very long arrays of values, and one would not make them any more readable than they are. Also, this language only supports strings and ints, which may be enough in the beginning but makes for lots of config restrictions, especially if we store the values that are to be deserialized into objects — and we do.

Another thought was to make up our format, yet it would require writing more logic for serialization and deserialization, which we cannot afford at the moment. Why bother if we already have formats well-applicable to our task?

4.6.2 What info to include

The following information was included in XML markdown used in PKLot:

- The parking ID
- Parking space ID and status
- Rotated rectangle (parking space) coordinates in 2 formats:
  - As center, width, height, and angle of rotation
  - As four points denoting vertices

The first drawback of such structure is that there is plenty of information which does not change between frames, namely locations of slots. On the other hand, labeling is easily changed between frames. This issue is what led us to separate these two kinds of data. The unchanging info goes to config, making it one config per parking, while labels are written into logs.

Another consideration is that rectangle markdown would be practically inconvenient for the user, so it is better to make it with arbitrary quadrilaterals, which are a natural representation of how slots look like at a picture, with all perspective warps objects undergo. Thus the config should be a list of slots, where a slot is a list of 4 points, where each point is two coordinates. On the other hand, a log is a map where the slot ID is a key, and a value is either Empty or Occupied. Slot IDs are 1-based and determined from the ordering in the config.

4.6.3 Metadata and labeling issues

Over the course of the discussion, we, as many aspiring developers before us, first set a goal to create an ultimate God tool that can do everything. Thus, we considered many labels for classification, just for everything to be detected. Among them, the sanest set was:

- Person
- Animal
- Truck
- Car
- Empty parking lot

However, over the course of dataset search, as well as architectural a practical considerations, we decided to simplify the set. The first question to ask was — will the possible user care whether there is a truck, or a car, or a bus? The answer was no for most cases, so we ended up with the following binary classification labels:

- Occupied parking space
- Empty parking space

Among all other things, this choice eased or lives in a sense that there was no urgent need to perform manual data collection anymore. Instead, we could find a suitable pre-existing dataset and focus on the system implementation.

There were also other issues we ran into over the course of our research. While with SVM and our CNN everything went smoothly with markdown, during the work with YOLO we needed to transform the annotations we had into an appropriate format (PASCAL VOC annotation format). It would take time

to implement necessary scripts. We decided to not start with it yet there still arose a need for plenty of scripts for selective training, so that only data relevant for us is used. The PASCAL VOC markdown is an XML which includes:

- folder name
- filename
- description of an image source
- image size
- segmented or not
- information on objects to detect
  - object category
  - object pose
  - truncated or not
  - occluded or not
  - bounding box as diagonal coordinates
  - difficult object or not

In our data, we did not have most of this information, which would have to be generated have we gone far enough to train YOLO on a representative dataset.

4.7 Image labeling tool

One of the parts of the project was an image labeling tool (ILT) created by my colleague. The idea for it arose from the need to manually markdown parking spaces to provide easy system portability for any parking lot there can be. The tool was designed in such a way that it would allow to save its output in JSON, save and continue work, and cancel previous actions.

The whole application was written in Python, with extensive use of OpenCV framework. It supports drawing slots using a mouse and has hotkeys for cancellation and saving work, and also allows to choose to work on unfinished markdown on launch.

While ILT works much as intended, it has some problems with usability and must come with a user guide to be usable, because it is not apparent from the interface how it is used.

4.8 Image preprocessing

Before we can feed the input to the model, we should preprocess it to the most suitable form. In the very beginning, we have a frame. We would need to rescale it. We would need to blur a copy of an image at some moment to adapt SSIM to our needs. We would need binarization to locate changes between frames. Finally, we would need to cut the slots somehow, and then apply the perspective transformation to make them rectangular and same-size.

To perform the preprocessing, we tapped into some popular libraries, with most useful for our purpose being Scikit-image and OpenCV. We decided to use mostly OpenCV to keep the unity of our code, but import SSIM from Scikit-image, as only the C++ interface of OpenCV has this functionality.

First of all, let us look at the issue of segment extraction. We need to cut slots, where a slot is an arbitrary quadrilateral. Many other implementations use rectangles, yet the point is, due to perspectives, many slots look like trapezoids or parallelograms. After that, points are rotated, so that the longest side would map to the left side. Finally, we use perspective transformation to change a quadrilateral into a 40x60 rectangle.

We have been implementing our system iteratively, adding something new on each iteration. At first, we did not perform any optimizations. Then, we decided to calculate frame differences and only predict on changed slots. However, we only started succeeding when counting differences on blurred images, to make up for the noise. Finally, we merged all overlapping areas of changes.

4.8.1 Benchmarking strategy

To determine which optimizations work, and which do not look like optimizations, we decided to do the following. For each version of preprocessing, we would test it repeatedly (three times) on the same video or picture sequence, while writing down the execution time, then compare average execution time for each version.

4.9 Feature extraction

For feature extraction, we decided to use HOG from the Scikit-image library, as the SVM training need not be integrated, and Scikit-image has better documentation.

While using HOG, we have not used any normalization, as it was the only feature descriptor we used, and thus rescaling was not needed. In terms of vector size, we believe that making pictures even smaller would degrade prediction quality, while more dimensionality could negatively affect performance. Furthermore, we found out that computation of HOG on all the 700 000 slots takes around 14GB of RAM.

4.10 Prediction part

4.10.1 Library capabilities

Table 4.1: ML library capabilities

Method	Keras	Scikit-learn	OpenCV
SVM	+	+	+
Neural network constructor (+ convolutional layer)	+	+	+
Grid Search or similar	-	+	-

4.10.2 SVM

The first predictor we decided to use was SVM (support vector machine) making predictions based on HOG (histogram of oriented gradients). This combination was chosen for the first try since SVM is one of the most frequently and successfully used techniques in smart parking, besides artificial neural networks. Another reason was that HOG is representative of objects shape, which is an essential characteristic in the task of detection of the situation on parking lots.

The overall principles of how this combination works are the following. HOG is calculated on picture fragments as a direction of a gradient from light to dark, thus outlining an object; it is a vector that has different length on different picture sizes, thus for usability, scaling all pictures to the same size would be required. SVM, on the other hand, will build two hyperplanes such that they are the farthest from the class border, and in optimization, will only take into account elements with margin below 1.

So we started implementing the plan, and immediately discovered that it is tied to numerous difficulties, some of which got us stuck for really long. Let us now describe in details how it was going on. As previously mentioned, we have found a more or less acceptable dataset (PKLot) and kept working with it for a while.

Let us now remember some facts. Each image of the database has an XML file associated including the coordinates of all the parking spaces and its label (occupied/vacant). By using the XML files to segment the parking space, one will be able to get around 695,900 images of parking spaces.

So now, the task was - how to choose a subset to train the model on.

It was evident that for some early prototype we do not need to use the whole dataset for training since all we have is a laptop which is not exactly powerful, so we decided to take a subset of this dataset just to try things out.

The way we chose that subset? Well, at first we just chose one folder (corresponding to one parking lot, one weather type, one day) at random for a basic check, it contained about 5k images and performed a check that everything works at all. Then we started tuning parameters with GridSearchCV

(parameters like C parameter, which is responsible for a trade-off between error correction and regularization, i.e model complexity/simplicity, kernel, which determines shape of decision boundaries, gamma, coef0, parameters used inside kernel functions, tolerance, which determines when to stop optimization). As a result, we almost always got just the same results, well over 99% of accuracy (with roughly equal sizes of classes).

That made us think the dataset is not representative, and thus we need a larger dataset.

We decided to add more images to our subset and added them in such a way that in the end, we had a dataset with both parking lots and all three types of weather, but other than that, the choice for folders with images was manual and random. As a result, the subset became 28990 pictures.

The main problem with this thing was that it overfitted easily. We tried to fight it but were not successful initially. The overfitting happened because we only took a small part of a dataset due to the scarcity of computational resources, and the dataset was sorted. Further on, when we got more resources for training models, we tried to fix main flaws in the initial experiment.

Further on, we chose another approach. We have calculated HOG for ALL the fragments, then randomly shuffled the dataset, then took 0.1 of the dataset as a training set. By then, we already had the server, so this time the training went well. After that, we repeated the procedure with fragments we have cut by ourselves.

The way the code works? At first, all pictures of a subset are read from a directory we placed them into. Then, labels for these pictures are produced from file paths (since files are sorted into free space and occupied space categories).

Then HOG features are extracted from these segments. Then the dataset is shuffled randomly and then sent for training to SVM wrapped into GridSearchCV for parameter tuning.

GridSearchCV uses 3-fold cross-validation for evaluation, that means that a given dataset Is randomly divided into three parts, then one of parts is used for validation and other two for training, where validation is done for each part, totaling three iterations.

The ML model apparently did not show much accuracy since we trained it on so-called perfect data, small and neat. These were, to be more specific, pictures of slots that were right in the dataset we obtained, that were already cut, rotated, more or less scaled to similar (but not equal) sizes and made rectangular. The prediction was mainly in our data, that is, slots we cut and preprocessed with our tooling. These pictures were drastically different from the training set.

As for the speed, it is explainable at least with the fact that the whole field was analyzed every time, while SVM is not what we routinely call a fast predictor. Still, it was only the first prototype, so it is acceptable.

Overall, It took about 24 minutes to train each of two last SVMs. Accuracy on validation was 0.98 for PKLot segment data and 0.95 for our segment data.

4.10.3 Yolo - a story of failure

Sometime after that, we were thinking about the paradigm of processing and learning workflow and came to new conclusions. Previously, the idea was that we mark the space the camera captures into areas of interest (such as parking spaces) and then take each of such areas separately and classify with multi-label classification (has cars or not, has people or not, and so on). The drawback of this approach is, for instance, that we will not be able to recognize when a car is not parked correctly (such as into two parking spaces at once).

So we had another idea, that probably we should take a picture, outline areas of interest, but then check a picture as a whole, and find all cars with their location (coordinates of bounding boxes), all people with their location, and so on. Then with bounding boxes check which areas of interest the given object falls into (that would be defined as an object having more than a certain amount of bounding box falling into a given area. The amount could be, for instance, 40% of a bounding box for an incorrectly parked car and 80% for a correctly parked one) and assign the respective label to that area of interest. All areas of interest not containing objects of interest (or objects at all?) would then be labeled empty. For this approach, we had no idea how to use linear models and if feature extraction would help at all, so we decided to search for convolutional neural network based solutions.

That was how we started searching for neural networks out there on the internet. The thing we ran into and at first, liked, is called YOLO. It is the neural network-based system for object detection, written in C. Also there are short but clear guidelines on the website (what is good, they are clear; what is bad, they are short, while we would prefer comprehensive documentation). In general, and on general-purpose datasets, it seemed to perform right, so we took a pre-trained model, and tested it on parking data. Results were not so good at times, as can be seen from Figure 4.7, so we decided we need to take a more specific dataset and retrain the model.

For easy training of this model, two datasets are proposed on their website, namely VOC and COCO. However, these are general purpose datasets, so we could not just only take and use them. For simplicity, we took VOC 2012 and left there only images of vehicles and people, which are relevant. For that, we wrote several python scripts that deleted unneeded images and mentions of them in labeling and marking files. Overall, we got four categories and around 5.5k images, not much, but we needed something to test the thing. Then we modified for our needs and used another script, which generated marking data in the correct format for a network. Then we launched the training, but in a couple of hours, we got a failure, after only 256 images having already gone into the network. We were disappointed. If not for that fact, we would consider adapting PKLot markdown to YOLO and tried, but with our failure, we moved on to other parts of the plan fast.

4.10.4 A simple CNN (made with keras tutorial)

Now, let us talk about neural networks. Here we found an excellent Keras tutorial on fast and dirty convolutional neural network suitable for small training set sizes (and this is true since more massive training sets made final results worse). So we have implemented several models using these instructions. Now, the last one of them works well enough to satisfy our requirements regarding prediction accuracy.

Overall, we got five models, similar in structure but trained on different data. Why so many models, and what does different data mean, one might ask. The answer is, we did a certain amount of experimentation.

The structure of all models is the same, and unchanged from the tutorial version, except for a target picture size. The difference is in training datasets.

To train the first model, we used segments that are an inherent part of the PKLot dataset. We took the least recent date from each weather from each parking, resulting in about 11k samples. For validation, we took two most recent dates from each weather from each parking, resulting in about 16k samples. All pictures were resized to 60*40, with some further preparation with standard means (oversampling and randomization with added distortions for robustness). The model shows accuracy on the validation set about 91%, according to Keras report on the network learning.

The second model was similar to the first one, with the only difference in dataset size. This time, for training, we took four least recent dates from each weather from one parking, resulting in about 17k images, and for validation, four most recent dates from each weather from another parking, resulting in about 26k images. As a result of training, the model showed around 96% accuracy on validation.

The third was done a bit differently from previous two, as it came out that training the network on parking space pictures led to poor results of prediction using parking space pictures we extracted in our prototype since they were pretty much different. Thus this model was trained on our data, that was previously extracted into two folders(for empty and occupied spaces) and ordered alphabetically. Then, for the training set, we chose (moved) first 10k images from each category, using a simple bash command. By the same principle, we chose validation images, with 70k images in each category. The model trained showed almost 99% accuracy on validation, according to Keras report.

The fourth model was similar to the third one, with the only difference being a size of training dataset, which was drastically increased. From the remaining pool of images, we took 20k of occupied space images, and 40k of empty space images, which were the first alphabetically. The validation set remained the same as in the previous model. This model also showed almost 99% of accuracy on validation, but during prediction, it worked much worse than the previous model, that became the reason this one was discarded. The reason for such change is supposed to be the size of the training dataset, which seemingly was too much for this kind of network.

The fifth model was born since we found out that for one of the cameras on

the parking which had two cameras we did not cut slots for a dataset for some reason. So we found it out and decided to cut it, and got the second folder of our dataset. Thus, this time, for training set we got 5k alphabetically first images from the first part of our dataset remaining, and 5k first images from the second part, for both empty and occupied slots, totaling 20k samples. For validation, we then took 30k empty spaces and 30k occupied spaces from the first part, and 15k empty spaces and 30k occupied spaces from the second part, totaling 105k images. In practice, this is the best performing model we got so far, so this is the one that is currently used in our prototype. Another reason for that might be that previous models were trained with 1 to 3 epochs, while this one was trained with five epochs, even though on the validation set the quality deteriorates after the second epoch.

4.11 Outputting results

We chose to have two ways of outputting results: logs — to store them to be further used by other software systems, and visualization — made in real time for humans.

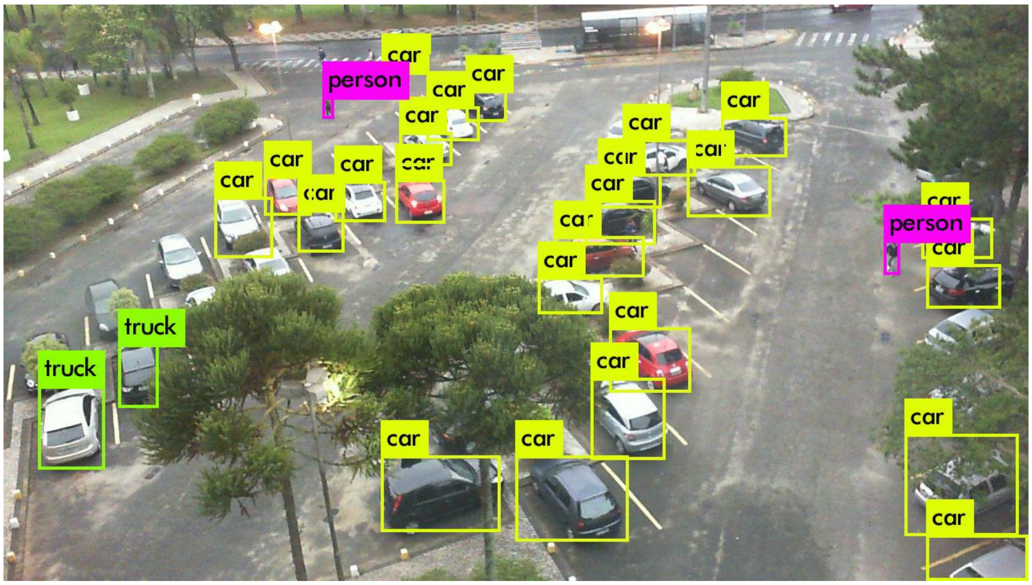
As it was mentioned before, the file format choice for the output (or log) is now JSON. The filename consists of the timestamp and file extension. The file itself is a dictionary (object), where keys are 1-based parking slot IDs, and values are labels — either empty or occupied. It is the most straightforward possible format that contains all the necessary information.

We output the video we get as an input so that people can also see what is going on in a parking lot by their own eyes. We also output labels and slot numbers as an overlay, — the green label meaning empty, and red label meaning occupied. Thus, we show slot borders as specified in markdown, either in red or green. This way was chosen because it is easy and intuitive so that humans can double check the system.

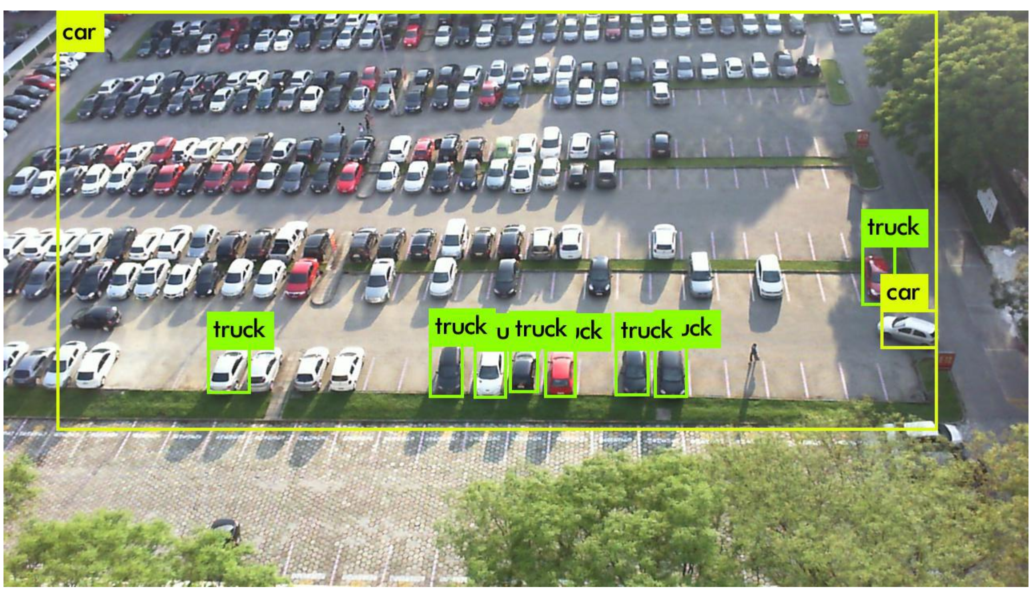
4.12 Methods of whole-system benchmarking

The last part of work would be to evaluate the whole system. For this, we decided to make a test set of pictures, large enough, such that they are all labeled. For these pictures, we could measure how long the system works to process them all, as well as calculate an overall confusion matrix, and from it, all the necessary measures. Also, we considered it necessary to calculate the average time to process one frame. The process described above is what is necessary to be done for each variation (version) of the system.

Figure 4.7: Example of YOLO predictions



(a) Good



(b) Bad

Chapter 5

Evaluation and Discussion

5.1 Overview of all setups used

Overall, here is the list of all the systems we had and need to provide performance data for:

- 3 SVM models
  - Trained on 0.1 of the PKLot slots
  - Trained on 0.1 of our slots
- YOLO pretrained
- Our systems based on CNNs
  - Model 1, with and without image difference
  - Model 2, 2 versions
  - Model 3, 2 versions
  - Model 4, 2 versions
  - Model 5, 2 versions.

Nevertheless, there is data relevant for prediction quality and relevant for benchmarking, so unnecessary checks were omitted.

5.2 Overview of evaluation methods

First of all, let us look in more detail at the metrics we are going to use to evaluate the prediction quality.

The confusion matrix is the fundamental notion of this section. With binary classification, we have some condition, and those objects that satisfy it, are called condition positives *(P)*. All the other elements are called condition negatives *(N)*. When predicting, we also assign an object either a positive or a negative label. Then there are four situations possible:

- Condition positive assigned positive — a true positive *(TP)*, or a *hit*
- Condition negative assigned negative — a true negative *(TN)*, or a *correct rejection*
- Condition negative assigned positive — a false positive *(FP)*, or a *false alarm*
- Condition positive assigned negative — a false negative *(FN)*, or a *miss*

A confusion matrix is a table, where these four cases are arranged as *TP*, *FP*, then on the next line *FN*, *TN*.

There can also be confusion matrices in multiclass classification, made using the same principle of arrangement of correct labels against predicted labels.

Confusion matrices are a sign of an honest result reporting, as they contain all the information about dataset distribution, and all the other metrics can be derived from these data.

On the other hand, sometimes they can prove to be not easily interpretable, as, for easy comparison, it is preferable that prediction quality is summarized as a single number.

Recall (5.1), also called sensitivity, hit rate, or true positive rate, is a ratio of true positive values over all the condition positives:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5.1}$$

The recall is good when it is important to identify as many condition positives as possible, yet it fails to take into account false positives which can be prevalent, degrading the prediction.

Precision (5.2), also called positive predictive value, is a ratio of true positives over all the assigned positives:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{5.2}$$

It is a measure of the fraction of selected values which are relevant. It says nothing about the number of condition positives overall and can be used to hide the fact that most of the condition positives are missed.

Finally, let us take a look at accuracy (5.3). It is a ratio of all correctly predicted values over all the values in a set:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{5.3}$$

This measure shows how many objects were predicted correctly. On the other hand, high accuracy does not always mean good prediction quality, as it also depends on the quantities of objects in categories. For instance, if condition positives are 1% of the set, the predictor that assigns all objects negative label will have an accuracy of 0.99.

5.3 Confusion matrices for all combinations

To evaluate the quality of prediction, we would first need to calculate confusion matrices. For convenience, we used the following test sets:

- To test SVMs, we randomly took 0.9 of the slot dataset we had and calculated results automatically using Scikit-learn functionality
- To test CNNs predicting without image diffs, we took one frame for each camera for each weather, a total of 9 frames, for manual calculations
- To test CNNs predicting with image diffs, we took two frames from each parking lot, totaling 6 frames. Then for each frame, we took the one captured five minutes before the given one (i.e., the closest one in a dataset) to use as an auxiliary picture, totaling 6 frame pairs, for manual calculations.

The statistics on the correct slot labels can be seen from Table 5.1.

Table 5.1: Testing sets label distribution

Test set version	Occupied	Empty	Total
CNN without optimization	201	303	504
CNN with optimization	373	213	586
SVM	338498	302173	640671

To enable a fair result reporting, we now publish all the confusion matrices, so that one can see all distributions and check the calculations. Some notable results include *CNN model 2** (Table 5.5), meaning the second neural network we trained, without the use of image difference, which is denoted by * sign. This model does not seem to be genuinely aware there can be empty slots, as there are only nine slots labeled as empty.

Overall, the spread between CNNs tested without optimizations is quite significant (Tables 5.4, 5.5, 5.6, 5.7, 5.8), while with optimizations, models perform almost uniformly (Tables 5.9, 5.10, 5.11, 5.12, 5.13).

5.4 Metrics summary

From the confusion matrices above, we have calculated all the primary quality metrics for binary classification (Table 5.14). The only exception is accuracy on validation, which was output by models themselves during training.

Table 5.2: SVM trained on PKLot segments

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 253845	FN 84653
	n′	FP 13466	TN 288707

Table 5.3: SVM trained on segments prepared by us

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 315881	FN 22617
	n′	FP 5540	TN 296633

Here it can be seen that the lowest recall falls on the *CNN model 5**, while still being almost 0.7, while among all the models with high recall the *CNN model 2** beats the way, giving assurance that almost all occupied slots are going to be correctly identified. The same model, on the other hand, has the lowest precision, meaning that over a half of spaces identified as occupied will be in fact empty. The top player in terms of precision turns out to be SVM, as the *SVM (prepared by us segments)* model has near-perfect precision, allowing us to trust that a slot predicted to be occupied will be occupied indeed.

While not an essential parameter in and of itself, validation accuracy highlights the third and the fourth CNNs, while the fifth one seems to be the least successful one. It is, however, a very misleading value in both cases, as the fifth CNN turns out to be the only one to hold a decent testing accuracy with no optimizations, while the lowest score belongs to *CNN model 2**, and the second lowest to *CNN model 4**. The absolute winner of accuracy is, however, an SVM

Table 5.4: CNN model 1 without optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 143	FN 58
	n′	FP 78	TN 225

Table 5.5: CNN model 2 without optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 198	FN 3
	n′	FP 297	TN 6

trained on our data.

5.5 Analysis

Overall, the best of all models is undeniably the SVM trained on our data. It has excellent recall, nearly perfect precision and the top testing accuracy, all over 90%, which fits state-of-the-art results.

The possible reasons for such a high result are, first of all, the properties of SVM searching the most optimal interclass boundary, as well as usage of relevant data for training, with training set size being good enough. It is hypothesized that with a training set three times as big, the results could significantly improve, but the respective model was not tested because of technical reasons.

The worst models are the second and the fourth CNN when used without

Table 5.6: CNN model 3 without optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 192	FN 9
	n′	FP 102	TN 201

Table 5.7: CNN model 4 without optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 182	FN 19
	n′	FP 221	TN 82

optimization, due to incredibly low selectivity, and as a result, low accuracy. It means the models are going to be as good as useless because of suboptimal quality of discrimination between two simple classes.

The reasons for that possibly stem from the fact that the architecture of the network is well-suited for small training sets. Both models were trained on biggest subsets from the respective datasets — the second one on PKLot slots, and the fourth one on significant amounts of our data. This fact seems to have contributed to overfitting of both models due to high similarity of objects across entire datasets.

The second best model seems to be the fifth CNN, as it shows stable and quite decent results both with optimizations and without them. It is especially crucial because as it is going to be shown below, CNNs do perform a lot faster without what was perceived to be optimizations. We believe that the best measure for determining prediction quality in our case is testing accuracy, as

Table 5.8: CNN model 5 without optimization

		Prediction outcome: p	Prediction outcome: n
actual value	p′	TP 140	FN 61
	n′	FP 14	TN 289

Table 5.9: CNN model 1 with optimization

		Prediction outcome: p	Prediction outcome: n
actual value	p′	TP 341	FN 32
	n′	FP 54	TN 159

quantities of empty and occupied slots in test sets are comparable and known, and thus accuracy can give us full information.

5.5.1 How can prediction be improved?

We believe that prediction could be improved with SVMs provided they be fast enough to predict slots close to real time. Increasing diversity in training sets could make other improvements, as well as through additional preprocessing. There is also an idea that predictions on the colored picture could make a difference to either side.

One more hypothesis is that if slot borders are defined based not on lined on the ground but instead based on the positions of imaginary 3D boxes, the prediction could also be improved by giving more pictures of whole vehicles.

Table 5.10: CNN model 2 with optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 365	FN 8
	n′	FP 63	TN 150

Table 5.11: CNN model 3 with optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 359	FN 14
	n′	FP 54	TN 159

5.6 Efficiency

5.6.1 Importance of time and memory spent

Remember we were planning to enable our system to work even on the low-cost hardware which is not very powerful. Thus it is essential that the system does not consume much resources such as CPU time and RAM, as we cannot guarantee that affordable hardware will have comfortable amounts of memory.

Also, we are going to make our system work with real-time video streams, which means that our system must be able to handle the videos at least in real-time. This is absolutely crucial if we want to see our system in production one day.

Table 5.12: CNN model 4 with optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 361	FN 12
	n′	FP 57	TN 155

Table 5.13: CNN model 5 with optimization

		Prediction outcome	
		p	**n**
actual value	**p′**	TP 356	FN 17
	n′	FP 56	TN 167

5.6.2 Benchmarking strategy

We decided to check these simple requirements directly. We benched two versions of the systems using the POSIX time tool. The memory consumption was checked through a memory profiler for Python, called `memory_profiler` version: 0.52.0.

For our work, we have used two computers. The first one is a five-year-old laptop, used for coding, testing, benchmarking, neural network training and other similar tasks:

- CPU: Intel Core i7 3630QM 2400 Mhz
- RAM: 8GB
- GPU: NVIDIA GeForce GT 635M, 2048MB

Table 5.14: Prediction quality assessment

Model	Recall	Precision	ACC on validation	ACC on testing
SVM (PKLot segments)	0.750	0.950	0.95	0.847
SVM (prepared by us segments)	0.933	0.983	0.98	0.956
CNN model 1*	0.711	0.647	0.91	0.730
CNN model 2*	0.985	0.400	0.96	0.405
CNN model 3*	0.955	0.653	0.99	0.778
CNN model 4*	0.905	0.452	0.99	0.524
CNN model 5*	0.697	0.909	0.86	0.851
CNN model 1	0.914	0.863	0.91	0.853
CNN model 2	0.979	0.853	0.96	0.879
CNN model 3	0.962	0.869	0.99	0.884
CNN model 4	0.968	0.864	0.99	0.882
CNN model 5	0.954	0.864	0.86	0.878

- HDD: 7200 rpm

The second one was a stationary PC with modern components, used for SVM training and testing. This computer was the one we also call the server:

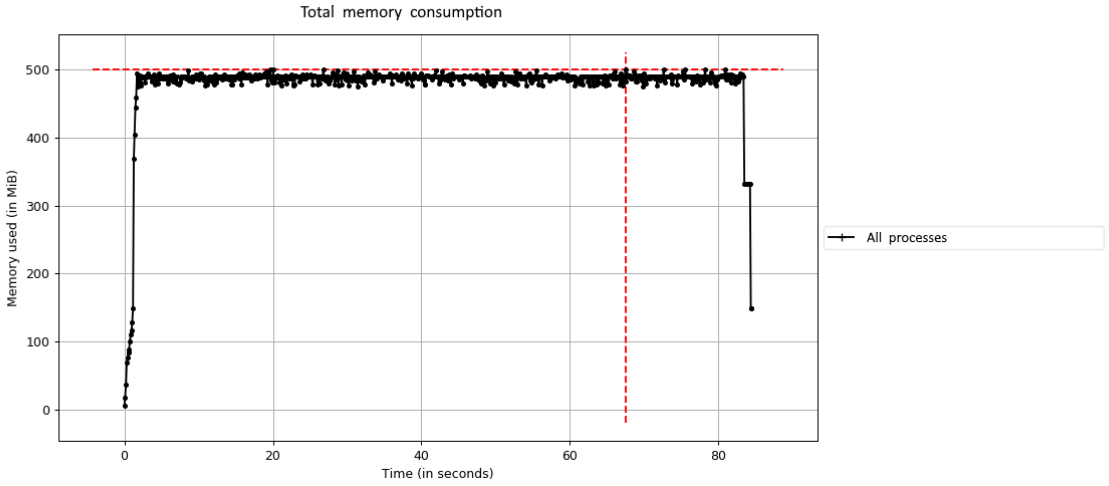
- CPU: AMD Ryzen 7 1700
- RAM: 24GB DDR4 2400 kt/s
- Storage: 1 TB cached HDD + SSD
- Swap: 10GB SSD 450Mb/s r/w + 60GB HDD 7200 rpm

The results of benchmarking were 1m40s user time for the version without optimizations, and 2m18s user time for the version with optimizations, for a video 2m1s long, on the laptop, as an average over five runs. This data implies that the so-called optimized version, in general, does not handle real time, meaning it is not suitable for the end goal. The plain version, however, is fast enough and has potential to work even on less powerful machines.

As for memory consumption, here is the situation. The total memory consumption for the plain version is at 500 MiB for the time of the system functioning (Figure 5.1), with the prediction module consuming up to 180 MiB (Figure 5.2). For the so-called optimized version, however, total memory consumption

rises as high as 640 MiB at times (Figure 5.3), with dynamic memory consumption from a predictor, being up to 300 MiB(Figure 5.4).

Figure 5.1: Whole system memory consumption without optimizations



It means that the version with optimizations using image differences was both slower and more memory-consuming, making us finally choosing the plain version as a final prototype.

Overall, the best combination seems to be the plain version of the system paired with the last of our CNNs, which was also notable for the fact that it was trained on five epochs, which is the maximum over all our attempts.

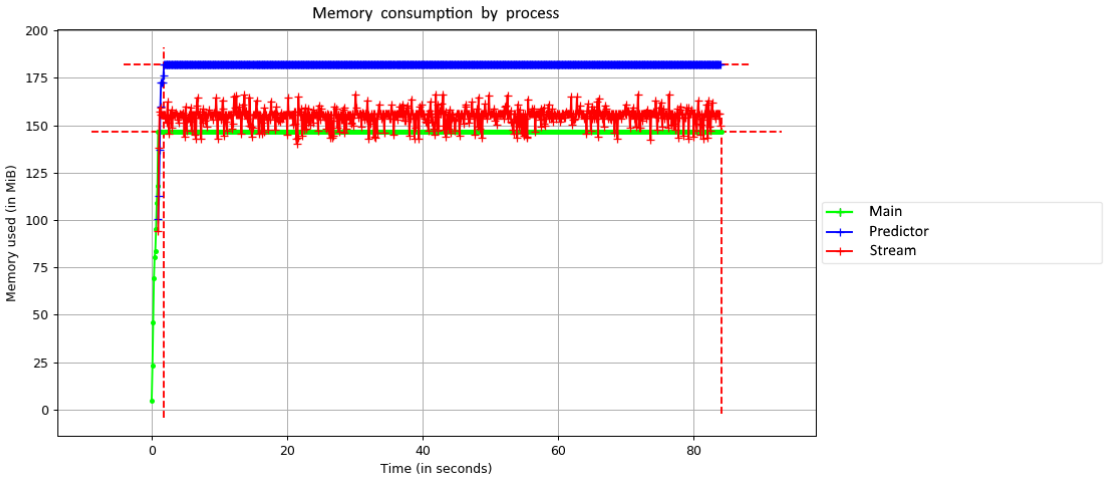
Now, considering the learning time for a bit. It took around 20 minutes to train each of the SVMs on the server, taking up around 16GB RAM, while training neural networks on the laptop was taking 1 minute per epoch on average not consuming many resources. This fact was what made us decide that probably SVMs are not the best choice since in case they need to be retrained, it will not be possible to do using a kind of computer the tool is supposed to work on.

One could further improve the efficiency in some ways, which are however nontrivial to implement:

- One could rewrite the system in C or a similar low-level language (C++, Rust). It would, first of all, make the system much faster, as Python is notable for being quite slow. Secondly, this step would reduce RAM consumption due to the elimination of runtime. Finally, writing software in lower-level languages could allow for tricky low-level optimizations, which could further increase the speed and save memory.

- It could be useful to stick to capabilities of only one computer vision

Figure 5.2: Memory consumption by process without optimizations



framework, possibly implementing all the missing functionality. It could reduce memory consumption due to fewer imports. Alternatively, one could try and implement their computer vision framework from scratch, provided they have enough time for coding.

5.7 Usability evaluation

While it is clear that our system is far from being perfect and complete due to being a mere proof of concept, and thus it is even further from perfect in terms of user experience, it is working, functioning, and functioning correctly, as a light-weight, free, extendible system. This way we proved that while straightforward solutions are not entirely accurate, they apply to the real life with some adjustments.

The next question is, how to improve the system and build on it. Of course, first of all, it is necessary to create a well-understandable, obvious, pretty, user-friendly GUI, to make the system more accessible. The second suggestion is to implement proper logging of internal processes and save the user from the need to learn how to use the terminal to use our system. Then, additional functionality can be gradually added, along with ensuring OS portability (As of now, the system is bound to work in Ubuntu 16.04, with no guarantees for any other platforms). Last but not least, it would be more convenient for the user if our image labeling tool became a system module and did not need to be launched by a separate console call.

Figure 5.3: Whole system memory consumption with optimizations

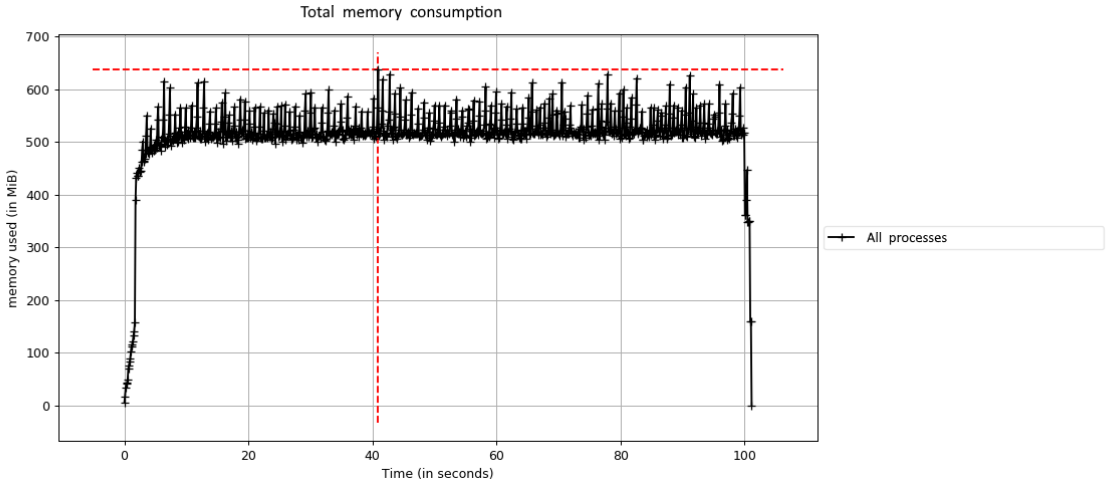
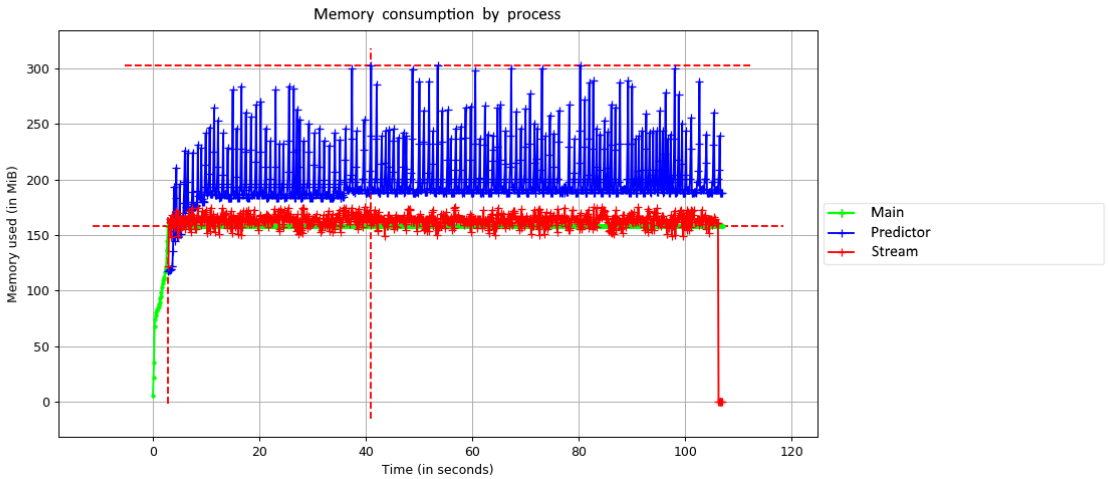


Figure 5.4: Memory consumption by process with optimizations



Chapter 6

Conclusion

Over the course of our thesis work, we have created a prototype of a camera-based system which is lightweight and straightforward, using a simple convolutional neural network. We have performed field tests, as well as compared to open state-of-the-art solutions, including systems based on SVM with use of HOG as well as YOLO. We found out that it can perform with the accuracy of about 86% in real time, detecting vehicles in parking slots in the daytime, except for the winter time which was not tested. Even though it is not the most accurate solution, it is cheap, adaptable, extendible and is a good step for parking monitoring automatization, which is, although not perfect, can be a base for more complex solutions in developed countries.

We also tested the performance to find out that not everything that is perceived to be an optimization is indeed an optimization, making space for out recommendations for more straightforward solutions.

This solution can be used for outdoors parking lots to assist parking lot monitoring, collect and analyze statistical data, as well as provide a broader network for smart transport solution in developing countries. It is way cheaper than any sensor-based solutions, being free by itself and only requiring an ordinary camera and an ordinary PC to function.

REFERENCES

Bibliography

[1] D. Evans, "The Internet of Things How the Next Evolution of the Internet Is Changing Everything," 2011.

[2] A. Zuiderwijk, Mila, P. Parycek, and M. Janssen, "Special Issue on Transparency and Open Data Policies: Guest Editors' Introduction," *Journal of theoretical and applied electronic commerce research*, vol. 9, pp. I – IX, 09 2014. [Online]. Available: https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-18762014000300001&nrm=iso

[3] Deliotte, 2016.

[4] M. Hilaga and Y. Shinagawa, "Topology matching for fully automatic similarity estimation of 3D shapes," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 203–212, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=383282

[5] E. Borovikov and A. Sussman, "A high performance multi-perspective vision studio," *Proceedings of the 17th annual international conference on Supercomputing - ICS '03*, p. 348, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?doid=782814.782862

[6] S. Nath, A. Deshpande, Y. Ke, P. B. Gibbons, B. Karp, and S. Seshan, "IrisNet: An Architecture for Internet-scale Sensing Services," *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, vol. Berlin, Ge, pp. 1137–1140, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315451.1315568

[7] C. H. Zhu, K. Hirahara, and K. Ikeuchi, "Street-parking vehicle detection using line scan camera," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 575–580, 2003.

[8] L. Bo, "Using Object Classification To Improve Urban Traffic," *Science And Technology*, no. 1, 2003.

[9] C.-H. L. C.-H. Lee, M.-G. W. M.-G. Wen, C.-C. H. C.-C. Han, and D.-C. K. D.-C. Kou, "An automatic monitoring approach for unsupervised parking lots in outdoors," *Proceedings 39th Annual 2005 International Carnahan Conference on Security Technology*, 2005.

[10] S. Midrange and P. Cars, "Vehicle Detection with a Mobile Camera," *Entropy*, no. March, pp. 37–43, 2005.

[11] H. Deng, D. Jiang, and Y. Wei, "Parking cell detection of multiple video features with PCA-and- bayes-based classifier," *Proceedings of IEEE ICIA 2006 - 2006 IEEE International Conference on Information Acquisition*, pp. 655–659, 2006.

[12] G. Toulminet, M. Bertozzi, S. Mousset, A. Bensrhair, and A. Broggi, "Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2364–2375, 2006.

[13] R. J. López Sastre, P. Gil Jiménez, F. J. Acevedo, and S. Maldonado Bascón, "Computer algebra algorithms applied to computer vision in a parking management system," *IEEE International Symposium on Industrial Electronics*, no. 3, pp. 1675–1680, 2007.

[14] S. F. Lin, Y. Y. Chen, and S. C. Liu, "A vision-based parking lot management system," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 2897–2902, 2007.

[15] G. Cormode and A. Mcgregor, "Approximation Algorithms for Clustering Uncertain Data Categories and Subject Descriptors," *roceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 191–200, 2008.

[16] A. Chianese, V. Moscato, and A. Picariello, "Detecting abnormal activities in video sequences," *Proceedings of the First International Conference on Ambient Media and Systems*, 2008. [Online]. Available: http://eudl.eu/doi/10.4108/ICST.AMBISYS2008.2827

[17] T. Fabián, "An algorithm for parking lot occupation detection," *Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008*, pp. 165–170, 2008.

[18] D. Bong, K. Ting, and K. Lai, "Integrated approach in the design of car park occupancy information system (COINS)," *IAENG International Journal of Computer ...*, no. February, 2008. [Online]. Available: http://www.doaj.org/doaj?func=fulltext{&}aId=380949

[19] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, 2008.

[20] M. Chitnis, Y. Liang, and J. Zheng, "Wireless line sensor network for distributed visual surveillance," *... Wireless Ad Hoc, Sensor, ...*, no. January, pp. 71–78, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1641890

[21] A. Subpa-asa, N. Futragoon, and P. Kanongchaiyos, "Adaptive 3-D scene construction from single image using extended object placement relation," *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry - VRCAI '09*, vol. 1, no. 212, p. 221, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1670252.1670299

[22] H. Ichihashi, A. Notsu, K. Honda, T. Katada, and M. Fujiyoshi, "Vacant parking space detector for outdoor parking lot by using surveillance camera and FCM classifier," *IEEE International Conference on Fuzzy Systems*, pp. 127–134, 2009.

[23] O. David-tabibi, N. S. Netanyahu, and M. Shimoni, "Genetic Algorithms for Automatic Classification of Moving Objects," vol. 20742, pp. 2069–2070, 2010.

[24] J. Salvador, X. Suau, and J. R. Casas, "From silhouettes to 3D points to mesh," *Proceedings of the 1st international workshop on 3D video processing - 3DVP '10*, p. 19, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1877791.1877797

[25] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrashekharan, W. Xue, M. Gruteser, and W. Trappe, "ParkNet : Drive-by Sensing of Road-Side Parking Statistics," *Challenge*, pp. 123–136, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1814433.1814448

[26] H. Ichihashi, T. Katada, M. Fujiyoshi, A. Notsu, and K. Honda, "Improvement in the performance of camera based vehicle detector for parking lot," *International Conference on Fuzzy Systems*, vol. 1, no. 5, pp. 1–7, 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5584554

[27] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C. C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, "AVSS 2011 demo session: A large-scale benchmark dataset for event recognition in surveillance video," *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2011*, no. 2, pp. 527–528, 2011.

[28] I. Kamal, "Car recognition for multiple data sets based on histogram of oriented gradients and support vector machines," *2012 International Conference on Multimedia Computing and Systems*, no. 3, pp. 328–332, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6320284

[29] K. Choeychuen, "Available car parking space detection from webcam by using adaptive mixing features," *JCSSE 2012 - 9th International Joint*

Conference on Computer Science and Software Engineering, pp. 12–16, 2012.

[30] K. Raman, K. M. Svore, R. Gilad-bachrach, and C. J. C. Burges, "Learning from Mistakes : Towards a Correctable Learning Algorithm Categories and Subject Descriptors," pp. 1930–1934, 2012.

[31] S. Gokul, G. S. Kumar, and M. Sreeraj, "Real time recognition of pedestrian and vehicles from videos," *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology - CCSEIT '12*, pp. 517–523, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2393216.2393303

[32] Z. Zhang, A. Mistry, W. Yin, and P. L. Venetianer, "Embedded smart sensor for outdoor parking lot lighting control," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 54–59, 2012.

[33] C. C. Huang, Y. S. Dai, and S. J. Wang, "A surface-based vacant space detection for an intelligent parking lot," *2012 12th International Conference on ITS Telecommunications, ITST 2012*, pp. 284–288, 2012.

[34] N. Goyette, "changedetection . net : A New Change Detection Benchmark Dataset," pp. 1–8, 2012.

[35] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 1, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2461912.2461919

[36] B. Jiang and X. Liu, "A divide-and-conquer approach to large scene reconstruction with interactive scene analysis and segmentation," *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry - VRCAI '13*, pp. 283–284, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2534329.2534372

[37] L. Lambrinos and A. Dosis, "Applying mobile and internet of things technologies in managing parking spaces for people with disabilities," *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13 Adjunct*, pp. 219–222, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2494091.2494162

[38] D. Delibaltov, W. Wu, R. P. Loce, and E. A. Bernal, "Parking lot occupancy determination from lamp-post camera images," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, no. Itsc, pp. 2387–2392, 2013.

[39] P. Almeida, L. S. Oliveira, E. Silva, A. Britto, and A. Koerich, "Parking space detection using textural descriptors," *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, no. August 2014, pp. 3603–3608, 2013.

[40] A. Chayeb, N. Ouadah, Z. Tobal, M. Lakrouf, and O. Azouaoui, "HOG based multi-object detection for urban navigation," *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, pp. 2962–2967, 2014.

[41] S. Karayev, M. Fritz, and T. Darrell, "Anytime recognition of objects and scenes," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 572–579, 2014.

[42] W. L. Hoo, T. K. Kim, Y. Pei, and C. S. Chan, "Enhanced random forest with image/patch-level learning for image understanding," *Proceedings - International Conference on Pattern Recognition*, pp. 3434–3439, 2014.

[43] A. Vora, M. A. Kumar, and K. G. Srinivasa, "Low Cost Internet of Things Based Vehicle Parking Information System," *Proceedings of the 6th IBM Collaborative Academia Research Exchange Conference (I-CARE) on I-CARE 2014*, pp. 16:1—-16:4, 2014. [Online]. Available: http://doi.acm.org/10.1145/2662117.2662133

[44] E. J. Sen, K. Deepa Merlin Dixon, A. Anto, M. V. Anumary, D. Mieheal, F. Jose, and K. J. Jinesh, "Advanced license plate recognition system for car parking," *International Conference on Embedded Systems, ICES 2014*, no. Ices, pp. 162–165, 2014.

[45] B. R. Payne, J. F. Lay, and M. A. Hitz, "Automatic 3D Object Reconstruction from a Single Image," *Proceedings of the 2014 ACM Southeast Regional Conference*, vol. 1, no. 1, pp. 31:1—-31:5, 2014. [Online]. Available: http://doi.acm.org/10.1145/2638404.2638495

[46] S. S. Mathew, Y. Atif, Q. Z. Sheng, and Z. Maamar, "Building sustainable parking lots with the Web of Things," *Personal and Ubiquitous Computing*, vol. 18, no. 4, pp. 895–907, 2014.

[47] A. K. Gupta, G. Ye, and J. Vendrig, "Hybrid Resolution Based Video Foreground Detection," 2014.

[48] R. Dube, M. Hahn, M. Schutz, J. Dickmann, and D. Gingras, "Detection of parked vehicles from a radar based occupancy grid," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1415–1420, 2014.

[49] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[50] Y. Sakai, T. Oda, M. Ikeda, and L. Barolli, "An Object Tracking System Based on SIFT and SURF Feature Extraction Methods," *2015 18th International Conference on Network-Based Information Systems*, pp. 561–565, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7350677/

[51] Jing Shao; Kai Kang; Chen Change Loy; XiaogangWang, "Deeply Learned Attributes for Crowded Scene Understanding," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4657–4666, 2015.

[52] M. Dixit, S. Chen, D. Gao, N. Rasiwasia, and N. Vasconcelos, "Supplement : Scene Classification with Semantic Fisher Vectors," p. 6964, 2015.

[53] Y. Yuan, L. Mou, and X. Lu, "Scene Recognition by Manifold Regularized Deep Learning Architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2222–2233, 2015.

[54] H. Ye, Z. Wu, R.-w. Zhao, X. Wang, Y.-g. Jiang, and X. Xue, "Evaluating Two-Stream CNN for Video Classification Categories and Subject Descriptors," pp. 435–442, 2015.

[55] H. Rajput, T. Som, and S. Kar, "An automated vehicle license plate recognition system," *Computer*, vol. 48, no. 8, pp. 56–61, 2015.

[56] F. J. Lin and H. Chen, "Improving utilization and customer satisfaction of parking space with M2M communications," *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings*, pp. 465–470, 2015.

[57] P. R. De Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich, "PKLot-A robust dataset for parking lot classification," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937–4949, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2015.02.009

[58] J. M. Menéndez, C. G. del Postigo, and J. Torres, "Vacant parking area estimation through background subtraction and transience map analysis," *IET Intelligent Transport Systems*, vol. 9, no. 9, pp. 835–841, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7305851/

[59] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[60] S.-M. Chen and C.-K. Chiang, "Rotation, Translation, and Scale Invariant Bag of Feature Based on Feature Density," *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 163–168, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7877207/

[61] H. Huttunen, F. S. Yancheshmeh, and C. Ke, "Car type recognition with Deep Neural Networks," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2016-Augus, no. Iv, pp. 1115–1120, 2016.

[62] S. M. Alwan, M. A. El-Abed, and R. N. Zantout, "Vehicles recognition from partial images," *2016 3rd International Conference on Advances in Computational Tools for Engineering Applications, ACTEA 2016*, pp. 236–240, 2016.

[63] T. H. N. Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, "Multiple Scale Faster-RCNN Approach to Driver's Cell-Phone Usage and Hands on Steering Wheel Detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 46–53, 2016.

[64] G. K. Yadav, P. Shukla, and A. Sethi, "ACTION RECOGNITION USING INTEREST POINTS CAPTURING DIFFERENTIAL MOTION INFORMATION Department of Electronics and Electrical Engineering , IIT Guwahati Department of Mathematics , IIT Guwahati , Guwahati," *Icassp 2016*, pp. 1881–1885, 2016.

[65] G. Madikenova, A. Galimuratova, and M. Lukac, "Threat detection in episodic images," *IDT 2016 - Proceedings of the International Conference on Information and Digital Technologies 2016*, pp. 180–185, 2016.

[66] L. Wu, Y. Yu, and J. Gu, "A Scene Recognition Method using Sparse Features with Layout-sensitive Pooling and Extreme Learning Machine," no. August, pp. 178–183, 2016.

[67] S. Guo, L. Liu, W. Wang, S. Lao, and L. Wang, "An Attention Model Based on Spatial Transformers for Scene Recognition," pp. 3757–3762, 2016.

[68] R. Mocan and L. Dios, "Multiclass classification based on clustering approaches for obstacle recognition in traffic scenes," pp. 1–5, 2016.

[69] C. Y. Chen, W. Choi, and M. Chandraker, "Atomic scenes for scalable traffic scene recognition in monocular videos," *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016.

[70] R. Du, S. Bista, and A. Varshney, "Video fields: fusing multiple surveillance videos into a dynamic virtual environment," *Proceedings of the 21st International Conference on Web3D Technology - Web3D '16*, pp. 165–172, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2945292.2945299{%}5Cnhttps://www.youtube.com/watch?v=5h9r2ksdJQc

[71] H. Assem, L. Xu, T. S. Buda, and D. O'Sullivan, "Machine learning as a service for enabling Internet of Things and People," *Personal and Ubiquitous Computing*, vol. 20, no. 6, pp. 899–914, 2016.

[72] V. Jain, Z. Sasindran, A. Rajagopal, S. Biswas, H. S. Bharadwaj, and K. R. Ramakrishnan, "Deep automatic license plate recognition system," *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing - ICVGIP '16*, pp. 1–8, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3009977.3010052

[73] J. Cho and J. Park, "Automatic Parking System using Background Substraction with CCTV Environment," 2016, pp. 1649–1652.

[74] L. Baroffio, L. Bondi, M. Cesana, A. E. Redondi, and M. Tagliasacchi, "A visual sensor network for parking lot occupancy detection in Smart Cities," *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings*, pp. 745–750, 2016.

[75] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car Parking Occupancy Detection Using Smart Camera Networks and Deep Learning," *Symposium on Computers and Communication IEEE*, no. Dl, 2016.

[76] I. Masmoudi, A. Wali, A. Jamoussi, and A. M. Alimi, "Vision based System for Vacant Parking Lot Detection : VPLD," *IEEE International Conference on Computer Vision Theory and Applications (VISAPP), Vol. 2., 2014.*, no. January 2014, pp. 1–8, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7294974/

[77] M.-c. Roh and J.-y. Lee, "Refining Faster-RCNN for Accurate Object Detection," pp. 3–6, 2017.

[78] I. Kamal and J. Oubaha, "Car recognition using the bag of features method," *International Conference on Multimedia Computing and Systems -Proceedings*, pp. 99–102, 2017.

[79] S. Ardianto, C.-J. Chen, and H.-M. Hang, "Real-time traffic sign recognition using color segmentation and SVM," *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 1–5, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7965570/

[80] X. Xiang, N. Lv, M. Zhai, and A. El Saddik, "Real-time Parking Occupancy Detection for Gas Stations Based on Haar-AdaBoosting and CNN," *IEEE Sensors Journal*, vol. 17, no. 19, pp. 6360–6367, 2017.

[81] N. Passalis and A. Tefas, "Learning Neural Bag-of-Features for Large-Scale Image Retrieval," pp. 1–12, 2017.

[82] X. Zhao, W. Li, Y. Zhang, T. A. Gulliver, S. Chang, and Z. Feng, "A faster RCNN-based pedestrian detection system," *IEEE Vehicular Technology Conference*, 2017.

[83] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[84] X. Wang, L. Lu, H. C. Shin, L. Kim, M. Bagheri, I. Nogues, J. Yao, and R. M. Summers, "Unsupervised joint mining of deep features & image labels for large-scale radiology image categorization & scene recognition," *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, pp. 998–1007, 2017.

[85] S. Valipour, M. Siam, E. Stroulia, and M. Jagersand, "Parking-stall vacancy indicator system, based on deep convolutional neural networks," *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, pp. 655–660, 2017. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015181563{&}doi=10.1109{%}2FWF-IoT.2016.7845408{&}partnerID=40{&}md5=f9bd5fcabd29c0897a93b2581014e103

[86] D. Neumann, T. Langner, F. Ulbrich, D. Spitta, and D. Goehring, "Online Vehicle Detection using Haar-like , LBP and HOG Feature based Image Classifiers with Stereo Vision Preselection," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, 2017.

[87] N. H. Barnouti, M. A. S. Naser, and S. S. M. Al-Dabbagh, "Automatic Iraqi license plate recognition system using back propagation neural network (BPNN)," *2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT)*, no. March, pp. 105–110, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7976099/

[88] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, "Segmentation- and Annotation-Free License Plate Recognition With Deep Localization and Failure Identification," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2017.

[89] F.-Y. Wu, S.-Y. Yan, J. S. Smith, and B.-L. Zhang, "Traffic scene recognition based on deep cnn and vlad spatial pyramids," 2017. [Online]. Available: http://arxiv.org/abs/1707.07411

[90] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[91] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[92] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[93] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

Appendix A

Code

A.1 Main

```
from multiprocessing import Process, Queue
from time import sleep
from pathlib import Path

from src.predictor import run_predictor
from src.stream import run_stream


def main():
    images_queue = Queue(1)
    predictions_queue = Queue(1)

    model_path = Path('model')
    video_path = Path('pltd.mp4')
    config_path = Path('coords.json')

    predictor_p = Process(target=run_predictor,
                          args=(model_path,
                                config_path,
                                images_queue,
                                predictions_queue))
    stream_p = Process(target=run_stream,
                       args=(video_path,
                       config_path,
                       images_queue,
                       predictions_queue))
```

```
    predictor_p.start()
    stream_p.start()

    while True:
        if not stream_p.is_alive():
            predictor_p.terminate()
            exit(0)

        sleep(1)


if __name__ == '__main__':
    main()
```

A.2 Stream

```
import json

from multiprocessing import Queue
from pathlib import Path

import cv2
import numpy as np



RED_COLOR = (0, 0, 255)
GREEN_COLOR = (0, 255, 0)
FONT = cv2.FONT_HERSHEY_SIMPLEX

def read_image(capture):
    ok, image = capture.read()
    if not ok:
        return None

    return image


def run_stream(video_path: Path,
               config_file: Path,
               images_queue: Queue,
               predictions_queue: Queue):

    with config_file.open() as file:
        config = json.load(file)
```

```
capture = cv2.VideoCapture(str(video_path))

image_prev = read_image(capture)
image_cur = read_image(capture)

images_queue.put((image_prev, image_cur))

start_predictions = Path("old.json")

with start_predictions.open() as file:
    prediction = json.load(file)

prediction = {int(k): v for (k, v) in prediction.items()}

while capture.isOpened():

    # Capture frame-by-frame
    image_cur = read_image(capture)
    if image_cur is None:
        break


    if images_queue.empty():
        try:
            images_queue.put_nowait((image_prev, image_cur))
        except:
            pass
    else:
        try:
            images_queue.get_nowait()
            images_queue.put_nowait((image_prev, image_cur))
        except:
            pass

    if not predictions_queue.empty():
        prediction_update = predictions_queue.get()

        prediction = {**prediction, **prediction_update}
        image_prev = image_cur

    for index, status in prediction.items():

        color = RED_COLOR if status == 'Occupied'
                          else GREEN_COLOR
        coordinates = np.array(config[int(index) - 1],
```

```
                                        np.int32).reshape((-1, 1, 2))

            cv2.polylines(image_cur, [coordinates], True, color)
            cv2.putText(image_cur,
                        str(index),
                        tuple(coordinates[0][0]),
                        FONT,
                        0.4,
                        color,
                        1, cv2.LINE_AA)

        cv2.imshow('Movie', image_cur)
        cv2.waitKey(1)


    capture.release()
    cv2.destroyAllWindows()
```

A.3 Labeler

```
from skimage.draw import polygon
from scipy.spatial import distance
import numpy as np
import cv2


class PicLabeler:
    def __init__(self, model, config):
        self.model = model
        self.slots = config

    def run(self, image, changes):

        self.image = image
        self.changes = changes
        self.image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
        self.changes = cv2.cvtColor(self.changes, cv2.COLOR_BGR2GRAY)

        self.height, self.width = self.image.shape
        self.pts2 = np.float32([[0, 60], [0, 0], [40, 0], [40, 60]])
        slots = []  # list of preprocessed slot images
        ids = []  # list of slot ids

        for index, space in enumerate(self.slots):
            slot, _ = self.process_slot(space)
```

```
            ids.append(index + 1)
            slots.append(slot)

        return self.predict(slots, ids)

    def iou(self, fig1, fig2):
        max_x = max(fig1[:, 0].tolist() + fig2[:, 0].tolist())
        max_y = max(fig1[:, 1].tolist() + fig2[:, 1].tolist())
        canvas = np.zeros((max_x + 1, max_y + 1), dtype=np.uint8)
        shape1 = np.copy(canvas)
        shape1[polygon(fig1[:, 0],
                       fig1[:, 1])] = 1
        shape2 = np.copy(canvas)
        shape2[polygon(fig2[:, 0], fig2[:, 1])] = 1
        intersect = cv2.countNonZero(cv2.bitwise_and(shape1, shape2))
        union = cv2.countNonZero(cv2.bitwise_or(shape1, shape2))
        iou = 0
        # assert union!=0
        if intersect > 0 and union > 0:
            iou = float(intersect) / union
        return iou

    def preprocess_coords(self, xs, ys):
        distances = []
        # calculate all side lengths of the quadrilateral
        for i in range(4):
            distances.append(
                distance.euclidean(
                    np.float32([xs[i], ys[i]]),
                    np.float32([xs[(i + 1) % 4], ys[(i + 1) % 4]])))
        # which one is the longest?
        starting_point = np.argmax(np.array(distances))
        # rearrange coordinates cyclically,
        # so that longest side goes first
        new_xs = xs[starting_point:] + xs[:starting_point]
        new_ys = ys[starting_point:] + ys[:starting_point]
        return new_xs, new_ys

    def predict(self, slots, ids):
        answer = {}
        if not slots:
            print("answer empty")
            return answer
        pred = self.model.predict(np.array(slots), 16, 1)

        # construct a JSON entity with results
```

```
        pred = pred.ravel().tolist()
        for i, one_id in enumerate(ids):
            answer[one_id] = 'Occupied' if pred[i] else 'Empty'
        return answer

    def process_slot(self, space):
        xs = []
        ys = []
        for point in space:
            xs.append(point[0])
            ys.append(point[1])
        # ensure contour is a quadrilateral.
        # This assertion failed once.
        assert len(xs) == 4
        assert len(ys) == 4
        # preprocess and save coordinates
        xs, ys = self.preprocess_coords(xs, ys)
        xs = np.float32(xs)
        ys = np.float32(ys)
        coords = np.vstack((xs, ys)).T
        # get a matrix for perspective transformation
        M = cv2.getPerspectiveTransform(coords, self.pts2)
        # transform a quadrilateral into a solid rectangle
        dst = cv2.warpPerspective(self.image, M, (40, 60))
        return np.reshape(dst, (40, 60, 1)), coords
```

A.4 Predictor

```
import json
from multiprocessing import Queue
from pathlib import Path
from time import sleep

from keras.models import load_model

from src.labeler import PicLabeler

def run_predictor(model_file: Path,
                  config_file: Path,
                  images_queue: Queue,
                  predictions_queue: Queue):
    # TODO check paths before

    model = load_model(str(model_file))
```

```
    with config_file.open() as file:
        config = json.load(file)

    labeler = PicLabeler(model, config)

    while True:
        image_prev, image_cur = images_queue.get()

        if image_prev is None or image_cur is None:
            sleep(1)
            continue

        result = labeler.run(image_prev, image_cur)
        predictions_queue.put(result)
```

A.5 Image labeling tool

```
import json

import numpy as np
import cv2


FINAL_LINE_COLOR = (0, 255, 0)

class PolygonDrawer(object):
    def __init__(self, window_name, pic, polygons):
        self.window_name = window_name # Name for our window
        self.polydone = False
        self.dump = False
        # Current position
        # we can draw the line-in-progress
        self.current = (0, 0)
        # List of points defining our polygon
        self.polypoints = []
        self.img = pic.copy()
        self.orig = pic.copy()
        self.polygons = polygons
        if self.polygons:
            for poly in polygons:
                cv2.fillPoly(self.img, np.array([poly]),
                             FINAL_LINE_COLOR)

    def on_mouse(self, event, x, y, buttons, u_args):
        # Mouse callback that gets called for every mouse event
```

```
        if self.polydone: # Nothing more to do
            return

        if event == cv2.EVENT_MOUSEMOVE:
            self.current = (x, y)

        elif event == cv2.EVENT_LBUTTONDOWN:
            # Left click means adding a point at current position
            # to the list of points
            print("Adding point #%d with position(%d,%d)"
                  % (len(self.polypoints), x, y))
            self.polypoints.append((x, y))
        elif event == cv2.EVENT_RBUTTONDOWN:
        # Right click means deleting last point
            self.polypoints = self.polypoints[:-1]
            print("Removing point #%d" % (len(self.polypoints)))

            self.img = self.orig.copy()
            for poly in self.polygons:
                cv2.fillPoly(self.img, np.array([poly]),
                             FINAL_LINE_COLOR)
            cv2.polylines(self.img, np.array([self.polypoints]),
                          False,
                          FINAL_LINE_COLOR, 1)
        elif event == cv2.EVENT_LBUTTONDBLCLK:
            # Left double click means we're done with polygon
            print("Completing polygon with %d points."
                  % len(self.polypoints))
            self.polydone = True

    def drawPoly(self):
        self.polydone = False
        self.polypoints = []

        #  disable a context menu on right button click
        GUI_NORMAL = 16

        # Creating working window
        # set a mouse callback to handle events

        cv2.namedWindow(self.window_name,
                        flags=(cv2.WINDOW_NORMAL | GUI_NORMAL))
        cv2.imshow(self.window_name, self.img)
        cv2.setMouseCallback(self.window_name, self.on_mouse)
```

```
        while not self.polydone:
            # Continuously draw new images
            # and show them in the named window

            if len(self.polypoints) > 1:
                # Draw all the current polygon segments
                cv2.polylines(self.img,
                              np.array([self.polypoints]),
                              False, FINAL_LINE_COLOR, 1)

            # Update the window
            cv2.imshow(self.window_name, self.img)

            # Wait 50ms before next iteration
            # (this will pump window messages meanwhile)
            cv2.waitKey(50)

        # User finised entering the polygon points
        # making the final drawing
        # of a filled polygon

        if self.polypoints:
            cv2.fillPoly(self.img,
                         np.array([self.polypoints]),
                         FINAL_LINE_COLOR)
        # Show final result
        cv2.imshow(self.window_name, self.img)
        # Waiting for the user to press any key

    def run(self):
        i = 1
        while not self.dump:
            self.drawPoly()
            self.polygons.append(self.polypoints)
            print("polygon #%d finished"%i)
            i += 1
            confirm = False
            while not confirm:
                rval = cv2.waitKey(0)

                if rval == ord('c'):
                    print("Confirmed")
                    confirm = True
                elif rval == ord('s'):
                    print("Saved")
                    self.dump = True
```

```
                    confirm = True
                elif rval == ord('z'):
                    print("Canceled")
                    i -= 1
                    self.polygons = self.polygons[:-1]
                    self.img = self.orig.copy()
                    for poly in self.polygons:
                        cv2.fillPoly(self.img,
                                     np.array([poly]),
                                     FINAL_LINE_COLOR)

                    cv2.imshow(self.window_name, self.img)
        return self.polygons


def typeOfWork():
    print("Type n for new work or c to continue work")
    while True:
        rval = input()
        if rval == 'n':
            return True
        elif rval == 'c':
            return False
        else:
            print("Invalid input.\n
                    Type n for new work or c to continue work")

if __name__ == "__main__":

    global_polygons = []

if typeOfWork():
    # code for new work
    # for new picture one argument - path to picture
    # parse path and pass it to imread method
    print("Print path to image file (includin file).\n
          Ex.:/home/user/folder/image.jpg")
    pathToImg = input()
else:
    # code for continueing work
    # arguments - path to picture and path to json file
    # parse both paths
    # pass picture to imread method and draw all polygons
    print("Print path to image file (includin file).
    \nEx.:/home/user/folder/image.jpg")
    pathToImg = input()
```

```
    print("Print path to json file (includin file).\n
    Ex.:/home/user/folder/labels.json")
    pathToJson = input()
    global_polygons = json.load(open(pathToJson))
    print(polygons)

img = cv2.imread(pathToImg, 1)
pd = PolygonDrawer("Polygon", img, global_polygons)
global_polygons = pd.run()

cv2.destroyWindow(pd.window_name)
for pol in global_polygons:
    cv2.fillPoly(img, np.array([pol]), FINAL_LINE_COLOR)
cv2.imwrite("polygon.png", img)
with open('coords.json', 'w') as f:
    f.write(json.dumps(global_polygons))
print("Polygon = %s" % str(global_polygons))
```

Appendix B

Licensing

B.1 Types of licenses overview

B.1.1 Creative Commons

A Creative Commons (CC) license is one of the public copyright licenses for free distribution of the content. There are several ways to choose the licensing condition, resulting in different licenses. They all contain the right to distribute the work non-commercially without derivative works. However, other conditions may differ. Four building blocks of Creative Commons licenses are:

- Attribution. While distributing, using, or changing the work, one should give credits (attribution to an author) in a manner specified by the author.
- Share Alike. Any derivative works should be distributed under the license which is no more restrictive than the license of the original work.
- Non-commercial. The work cannot be used in any way for commercial purposes.
- No Derivatives. It is forbidden to produce remixes of this work, so only the original version should be used.

Correctly combining these clauses, one can get six valid licenses:

- Attribution
- Attribution + Share Alike
- Attribution + Noncommercial
- Attribution + No Derivatives
- Attribution + Noncommercial + Share Alike
- Attribution + Noncommercial + No Derivatives

Creative Commons licenses are easy to use, provide a way to promote oneself with further ability to monetize ones work, save time spent allowing people free use of the work, and be given credit.

On the other hand, one cannot revoke permissions once the work is published under Creative Commons. Moreover, this is not a kind of license used to control the use of the software.

B.1.2 MIT

The MIT license is a permissive free software license which is also GPL-compatible. It is the most popular license on GitHub, as of 2015.

By this license, one is permitted to fully reuse the work for any purpose, provided they include the full text of the license with the distribution code. It is the main, and the only actual close of the license. Also, it means that the original creators do not hold any warranty regarding the usage of their code.

The MIT license is incredibly convenient, since it is much shorter than other open source licenses, while its only restriction is the obligation to include the full license code along with the copyright.

The main disadvantage, however, is that is it a software license, and different countries have different laws regarding what is software and what is just code, so sometimes it can be tricky to determine whether the license applies to ones case or not.

B.1.3 Apache

Apache License is another permissive free software license, created by Apache Software Foundation. It grants rights both regarding copyrights and patents, unlike other licenses which are not explicit about it. Any usage is permissible, and it is allowed to chose the license of derivative works. However, one must include the copy of the license and provide a clear Apache attribution. Also, each modified file should contain notices on the modification.

B.1.4 GPL

The GNU General Public License is a free software license, which guarantees end user the freedom to run, share and modify the software. GPL is copyleft, meaning one can only distribute derivatives under the same or similar terms. GPL is written in such a way that it allows to neutralize the laws forbidding the free software, protects users right to mess with software in whatever way, protects developers from patent threats, allows license compatibility, clarifies ways to provide the source code. The good in GPL is that it is a very ethical license, aimed at the greater good. It ensures that what is published as open source will remain open source. The disadvantages, however, are that many organizations explicitly ban usage of GPL-licensed code, some people revert from open source because of the belief that anyone should be able to do whatever he or she wants with the code, and many developers also think a lot before using

GPL-licensed components in their code. It is written in such a way that it tricks one to relicense under GPL if even one small GPL component is used.

B.2 What license we chose

Considering the explanations of the open source software licenses clarified above, we chose to distribute our system under the clauses of the MIT license, to ensure the freest possible use of our software to let the maximum amount of people, including those in business, benefit from it.