



Norwegian University of Life Sciences
Faculty of Science and Technology
and SINTEF Digital

Philosophiae Doctor (PhD)
Thesis 2021:73

Explainable and Data-efficient Learning for Visual Guidance of Autonomous Agri-robots

Forklarbar og dataeffektiv maskinl ring for
visuell styring av autonome landbruksroboter

Marianne Bakken

Explainable and Data-efficient Learning for Visual Guidance of Autonomous Agri-robots

Forklarbar og dataeffektiv maskinl ring for visuell styring av
autonome landbruksroboter

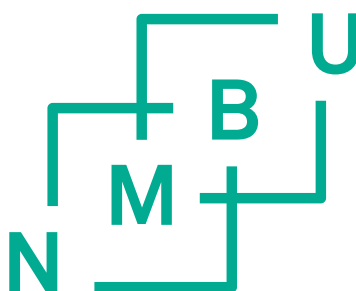
Philosophiae Doctor (PhD) Thesis

Marianne Bakken

Norwegian University of Life Sciences
Faculty of Science and Technology

and SINTEF Digital

 s (2021)



To Sigvald

SUPERVISORS AND EVALUATION COMMITTEE

Pål J. From (Main supervisor) Professor at Faculty of Science and Technology, Norwegian University of Life Sciences, Norway

Richard J.D. Moore (Co-supervisor) Senior Researcher at SINTEF Digital, Norway

Johannes Kvam (Co-supervisor) Researcher at SINTEF Digital, Norway

François Chaumette (First opponent) Senior Research Scientist at National Institute for Research in Computer Science and Automation (Inria), France

Alexander Binder (Second opponent) Associate professor at University of Oslo, Norway

Ingunn Burud (Committee coordinator) Professor at Faculty of Science and Technology, Norwegian University of Life Sciences, Norway

PREFACE

This thesis is submitted in partial fulfilment of the requirements for the degree of *Philosophiae Doctor* (Ph.D.) at the Norwegian University of Life Sciences (NMBU). I started this work after four years as a research scientist at SINTEF. In parallel with this work, I have continued with a 25% position at SINTEF and participated in research projects there. This research was conducted at NMBU and SINTEF from 2017 to 2021 as an institute PhD funded by the Norwegian Research Council ¹, resulting in a collection of five papers included in this thesis. The last three papers are partially funded by internal grants at SINTEF.

The topic of this thesis is the application of learning-based methods within vision-based guidance of agricultural robots. This is a topic that is cross-disciplinary and applied in nature, and involves both machine learning, computer vision, robot guidance and practical fieldwork. Because of this, the background chapters span a broad range of topics, and they are covered with varying depth to keep them concise but relevant for the research problems addressed in the papers. The intention has been to present it in a way that is accessible for an audience with a background in one of these disciplines or a related field.

Acknowledgements

I could never have done this without all the fantastic people that have been a part of my life at SINTEF, the university and outside work these four years. I want to thank everyone for being there in their own way, and I wish I could mention everyone here. Some have had a more direct impact on this work, which is worth mentioning.

First of all, I want to thank all my supervisors for joining me on this journey, and SINTEF Digital for choosing me as one of their candidates for institute Ph.D. grants. In 2017, I was working primarily with vision for UAVs, and the original topic of this project was to apply deep learning for guidance of autonomous drones. Thus Richard Moore became my supervisor from SINTEF, and has actively supported me all the way from the beginning, which I am

¹Grant number 259869

Preface

very grateful for. I was happy to finally have the chance to collaborate with Pål From and his team at NMBU and Saga Robotics. I gradually worked more with agri-robots than drones, and I must thank everyone at the NMBU robotics group and Saga Robotics for help with the robot setup and for providing a great social community. Especially Lars Grimstad, Vignesh Ponnambalam and Tuan Le. I will also thank Simen Myhre and Per Fredrik Saxebøl for access to their farms and practical help during fieldwork.

Working on a Ph.D. is not always smooth sailing, and about halfway through this journey, I hit some rough seas. I am extremely grateful for all the support from my colleagues, especially Helene Schulerud who helped me shift to a more sustainable gear. Another crucial factor was the inspiring and productive collaboration with Johannes Kvam, who generously offered me the opportunity to collaborate on exploring his idea on the visualisation of neural networks. Apart from that, I have the best colleagues in the world at the Smart Sensor Systems department, who are always eager to share their abundance of knowledge and do the silliest things just to have fun. I will also thank SINTEF Digital, particularly Mats Carlin, for having faith in me and extending the funding of my Ph.D.

Last but not least I am grateful for support and patience from friends and family. My husband Sigvald (also best friend, home office colleague, programming consultant and reviewer), deserves more praise than I am able to convey here.

Thank you.

Marianne Bakken

Oslo, July 2021

CONTENTS

Supervisors and evaluation committee	iii
Preface	v
Abstract	xiii
List of publications	xix
1 Introduction	1
2 Agricultural robotics	5
2.1 Agricultural robot applications	5
2.2 Guidance of agricultural robots	6
2.3 Robot and camera geometry	13
2.4 Experimental robot setup	16
3 Machine learning fundamentals	19
3.1 Learning a model	19
3.2 Neural networks	21
3.3 Convolutional neural networks	22
3.4 Training neural networks	25
3.5 Deep network architectures	26
3.6 Regularisation	27
4 Machine learning in practice	29
4.1 Data-efficient learning	29
4.2 Supervision strategies for agri-robot guidance	31
4.3 Explaining deep neural networks	32
5 Summary of papers	39
5.1 Paper I: End-to-end Learning for Autonomous Navigation for Agricultural Robots	39
5.2 Paper II: End-to-end Learning for Autonomous Crop Row- following	40
	vii

Contents

5.3	Paper III: Robot-supervised Learning of Crop Row Segmentation	40
5.4	Paper IV: Principal Feature Visualisation in Convolutional Neural Networks	41
5.5	Paper V: Applied learning for row-following with agri-robots	41
5.6	Relevant papers not included in the dissertation	42
5.7	Scientific contribution	42
	Bibliography	43
	Papers	52
I	End-to-end Learning for Autonomous Navigation for Agricultural Robots	53
II	End-to-end Learning for Autonomous Crop Row-following	61
III	Robot-supervised Learning of Crop Row Segmentation	69
IV	Principal Feature Visualisation in Convolutional Neural Networks	79
V	Applied learning for row-following with agri-robots	97
	Appendices	113
A	Supplementary material Paper IV	115
B	Supplementary material Paper III	123
B.1	Implementation of label projection for segmentation . . .	123
B.2	Label projection	123
B.3	Field map	124
C	Video material	125
C.1	Paper III	125
C.2	Paper V	125

LIST OF FIGURES

1.1	Example agricultural applications	2
2.1	Thorvald robot applications	6
2.2	Agri-robot platforms	7
2.3	Greenness index	12
2.4	Robot coordinate frames	13
2.5	Camera models	15
2.6	Robot setup	17
2.7	Strawberry field datasets	18
3.1	Overfitting and underfitting in machine learning.	20
3.2	A two-layer neural network	21
3.3	Illustration of a three-layer CNN	23
3.4	Illustration of the convolution operation	24
3.5	Training and validation loss	25
3.6	Illustration of a bottleneck architecture for image classification	26
3.7	Encoder-decoder architecture for segmentation	26
4.1	Randomisation test for explanation methods	37
B.1	Illustration of the label projection principle	123

LIST OF TABLES

2.1	Overview of navigation sensors	8
4.1	Comparison of explanation methods	34

ABSTRACT

To feed a growing world population and achieve the goal of zero hunger², we must develop new technologies to improve farm productivity and sustainability. Agri-robots can be a part of this solution, but new research is needed to provide reliable and low-cost autonomous operation across the broad spectrum of agricultural environments. Combining low-cost RGB cameras for vision with the recent advances in deep learning is a promising direction that can enable easier adaptation and lower hardware costs than existing solutions.

We explicitly tackle two of the main challenges faced when applying deep learning in robotics: learning from data of limited quantity and/or quality, and making neural networks easier to understand for humans. Thus, the main objectives of this work are to develop and apply methods that are more data-efficient and explainable than state-of-the-art in learning-based visual robot guidance, and to apply this insight to guide agri-robots in the field.

These topics are explored through five papers. First, we investigate the properties of an established end-to-end learning strategy for guidance and apply it in crop row following. Although promising at first, the black-box nature of this approach and inherent difficulties for debugging led to two different strategies; 1) a more explainable network architecture with a new supervision strategy for this task, and 2) a novel visualisation method to better understand visual features in convolutional neural networks. Finally, we unite these strategies in a new hybrid learning approach for row following that is both robust, data-efficient and more transparent.

The main contributions of this thesis are 1) *Increased explainability* through the development of a novel feature visualisation method, which provides explanations that are complementary to existing methods, 2) *Increased data-efficiency* and adaptability of learning-based crop row following through a new supervision approach which eliminates the need for hand-drawn labels, and 3) *New insight into applications* of learning-based methods in the field, by testing several supervision strategies on a real robot in the field, and considering the whole pipeline from data collection to predicted steering angle.

²The second UN sustainability goal, <https://www.un.org/sustainabledevelopment/hunger/>

ABSTRACT IN NORWEGIAN/SAMMENDRAG

For å brødfø en voksende verdensbefolkning og oppnå målet om å utrydde sult³, er vi nødt til å utvikle ny teknologi for økt bærekraft og produktivitet i landbruket. Landbruksroboter kan være en del av løsningen, men vi trenger ny forskning for å oppnå pålitelige autonome operasjoner til en lav pris innenfor et enormt spekter av ulike miljøer. En lovende retning er å kombinere billige RGB-kameraer med de nylige fremskrittene innenfor dyplæring, som kan gi løsninger som er enklere og mer tilpasningsdyktige enn de som eksisterer i dag.

I dette arbeidet ser vi spesielt på to utfordringer som oppstår når man anvender dyplæring innenfor robotikk; Å lære mest mulig fra data med begrenset mengde og kvalitet, og å gjøre beslutningene til nevralt nett enklere å forstå for mennesker. Hovedmålet er å utvikle og anvende metoder som er mer dataeffektive og forklarbare enn eksisterende læringsmetoder for radfølging, og anvende denne innsikten til å utvikle et system for autonom styring av landbruksroboter ute i åkeren.

Disse temaene har blitt utforsket gjennom fem artikler. Først undersøkte vi egenskapene til en etablert alt-i-ett (end-to-end) læringsmetode for styring, og tilpasset metoden til visuell radfølging i åker. Selv om de første resultatene var lovende, viste det seg etter hvert at metoden var vanskelig å forstå og feilsøke. Dette motiverte oss til å utvikle to nye metoder: 1) En ny veiledningsstrategi som gjør at vi kan bruke en mer transparent nettverksarkitektur som er lettere å forstå, og 2) en helt ny visualiseringsteknikk som viser hva slags egenskaper konvolusjonsnettverket har lært. Til slutt forener vi disse to metodene i en hybrid læringsstrategi som er både robust, dataeffektiv og mer transparent.

De vitenskapelige hovedbidragene i denne avhandlingen er som følger 1) *Økt forklarbarhet* gjennom utvikling av en helt ny teknikk for visualisering av egenskaper som er komplementær til eksisterende metoder, 2) *Mer dataeffektive* og tilpasningsdyktige metoder for radfølging gjennom nye måter å veilede dyplæringen på som eliminerer behovet for manuell merking av data, og 3) *Ny innsikt innen anvendelser* av læringsbaserte metoder i felt, gjennom testing av flere ulike strategier med en ekte robot i jordbæråkeren, og ved å betrakte hele systemet under ett fra datainnsamling til estimert styringsvinkel.

³FNs bærekraftsmål nr. 2, <https://www.fn.no/om-fn/fns-baerekraftsmaal/utryddesult>

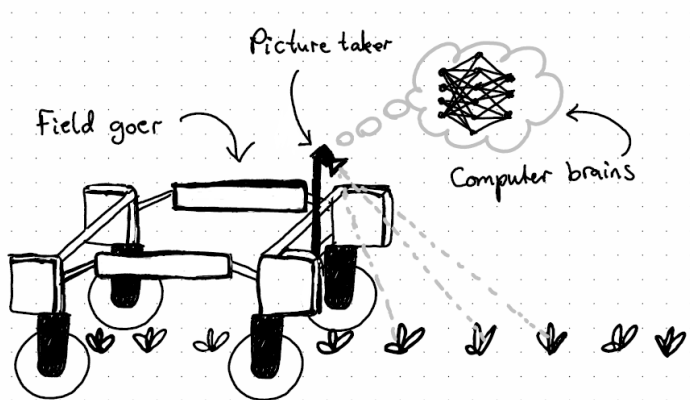
ABSTRACT IN SIMPLE ENGLISH

Sometimes, explaining your work in the simplest language gives new insight. This is written according to xkcd's Simple Writer⁴ with a vocabulary of only thousand words.

What this work is about

In this work, we make field-goers find their way using picture takers and computer brains. To teach the field-goer not to drive on the small red food things, we show a lot of pictures to the computer brain and tell it what is where. This is very boring and can take a long time, so we found a way to make the picture taker tell the computer brain what is where, so we don't need to do it anymore. To know if the computer brain has made a good choice, we told it to draw a picture to show us what it is thinking.

⁴A Controlled Natural Language created by Randall Munroe, used in Up Goer Five <https://xkcd.com/1133/> and [Munroe, 2015], and further analysed in [Kuhn, 2016].



LIST OF PUBLICATIONS

Papers included in the dissertation

Paper I

Marianne Bakken, Richard J. D. Moore, Pål From. “End-to-end Learning for Autonomous Navigation for Agricultural Robots”. Workshop on Robotic Vision and Action in Agriculture at ICRA 2018. <https://research.qut.edu.au/future-farming/projects/icra-2018-workshop-on-robotic-vision-and-action-in-agriculture/>

Paper II

Marianne Bakken, Richard J. D. Moore, Pål From “End-to-end Learning for Autonomous Crop Row-following”. IFAC-PapersOnLine 52.30, Special Issue for 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019. (h5-index: 52) <https://doi.org/10.1016/j.ifacol.2019.12.505>

Paper III

Marianne Bakken, Vignesh Raja Ponnambalam, Richard J. D. Moore, Jon Glenn Omholt Gjevestad and Pål From. “Robot-supervised Learning of Crop Row Segmentation”. IEEE International Conference on Robotics and Automation (ICRA) 2021. (h5-index: 94)

Paper IV

Marianne Bakken, Johannes Kvam, Alexey A. Stepanov, Asbjørn Berge. “Principal Feature Visualisation in Convolutional Neural Networks”. Lecture Notes in Computer Science, vol 12368, Proceedings of European Conference on Computer Vision (ECCV) 2020. (h5-index: 144) https://doi.org/10.1007/978-3-030-58592-1_2

Paper V

Marianne Bakken, Johannes Kvam, Richard J. D. Moore, Pål From. “Applied learning for row-following with agri-robots”. Submitted to Computers and Electronics in Agriculture, July 2021. (h5-index: 61)

List of publications

Relevant papers not included in the dissertation

Paper VI

Vignesh R. Ponnambalam, Marianne Bakken, Richard J.D. Moore, Jon Glenn Omholt Gjevestad, Pål J. From. (2020) “Autonomous Crop Row Guidance Using Adaptive Multi-ROI in Strawberry Fields” *Sensors* 20, no. 18: 5249. <https://doi.org/10.3390/s20185249>

Paper VII

Jonatan S. Dyrstad, Marianne Bakken, Esten I. Grøtli, Helene Schulerud and John Reidar Mathiassen. “Bin Picking of Reflective Steel Parts Using a Dual-Resolution Convolutional Neural Network Trained in a Simulated Environment.” *IEEE International Conference on Robotics and Biomimetics (ROBIO) 2018*. Received the T. J. Tarn Best Paper in Robotics award. <https://doi.org/10.1109/ROBIO.2018.8664766>

Other relevant contributions

Marianne Bakken, Johannes Kvam, Asbjørn Berge. “Fast reasoning visualization for deep convolutional networks”. Presentation at Northern Lights Deep Learning Workshop 2020.

Marianne Bakken, Richard J. D. Moore, Pål From. End-to-end Learning for Autonomous Navigation for Agricultural Robots. Presentation at Northern Lights Deep Learning Workshop 2019.

Marianne Bakken and Silvija Seres. “Når robotene får øyne”. An episode of the popular science podcast Lørn. <https://www.lorn.tech/podder/0366-n%C3%A5r-robotene-f%C3%A5r-%C3%B8yne>

CHAPTER 1

INTRODUCTION

We stand at a point in history where the separation between the digital and physical worlds is blurring. Plummeting hardware costs and the increased computational power at everyone’s fingertips enables an explosion of new technology and applications, like wide-spread video conferencing with augmented reality that can hide your messy background at the home office or suddenly turn you into a cat. This crossroad can also be a game-changer for more pressing matters, like producing enough food.

One of the major global challenges we are facing today is to feed a growing world population whilst battling a changing climate. This is related to the second of the UN’s sustainable development goals, *Zero Hunger*¹, which states that “Increasing agricultural productivity and sustainable food production are crucial to help alleviate the perils of hunger”, naming technology development as one of the targets on the way to achieve this ambitious and important goal. In agriculture, there is an enormous technology gap between the large-scale industrialised monocultures on one side and the smaller and more diverse farms on the other (Figure 1.1). The first category suffers from a lack of precision, spraying large amounts of pesticides on the whole field when only a fraction is infected. The second often relies on large amounts of seasonal manual labour, for instance during the short and intense strawberry season in Norway. Automating agricultural practices with fleets of agri-robots can improve farm productivity and sustainability at both ends of this spectrum by increasing precision and efficiency. However, the wide variety of agricultural applications and environments makes this a challenging task that is beyond the current state-of-the-art within robotics research.

A key technology for future large-scale deployment of agri-robots is low-cost navigation solutions. Current systems typically rely on accurate global positioning, which works well for open fields, but fails in indoor environments and requires expensive equipment on every single robot. Other solutions are based on scanning LIDARs which typically works well for more confined spaces with vertical structures. This works well in polytunnels or orchards, but it will usually require tailor-made algorithms for a specific farm or crop type. Future large-scale agri-robot fleets need sensing solutions that can operate on all kinds of crop types and environments with minimal setup cost. One possible solution to this problem is probably already in your pocket: Images from common RGB cameras contain an extreme amount of information, the challenge is to convert it into something useful. To detect crop rows in camera images, it is common to look for something green to separate plants from the brown ground. However, this is sensitive to seasonal changes, and is not always

¹<https://www.un.org/sustainabledevelopment/hunger/>

1. Introduction



Figure 1.1: The diversity in agricultural applications is almost limitless. From the top, row-wise: Pesticide spraying with an aeroplane in a wheat field in Argentina, worker picking strawberries in a polytunnel in Norway, manual harvest in a flooded rice field, transport during broccoli harvest in Norway. Photo credits, from top and row-wise: Santiago Nicolau, Stian Tandberg, World Bank Photo Collection, Saga Robotics.

the most distinguishing feature, as illustrated with the all-green sea of broccoli leaves in Figure 1.1.

With the current success and advancement of deep learning, it is possible to learn any kind of visual feature directly from images. This means that instead of tailoring the algorithms to every crop type or season, we can train a neural network using examples labelled with the properties it should learn. This is an extremely powerful tool, which not only can learn the difference between crops and lanes, but also give steering commands to the robot. But how do we collect good examples that cover all the variation the robot will ever encounter in the field? And how can we understand the reason behind a command if it makes a mistake?

This thesis seeks to address two of the main challenges of applying deep learning for robot vision. As reflected in the title, the focus is on 1) *data-efficient* learning, i.e. dealing with data of a limited quality or quantity, 2) *explainable learning*, i.e. making the black box a bit more transparent for humans, and 3) applying this insight to visual navigation for agri-robots on real farms. Based on this motivation, we defined

the following research questions:

1. How can modern learning-based methods best be *applied* to crop-row following with agri-robots?
2. How can such methods be made more *explainable*?
3. How can such methods be made more *data-efficient*?

Attacking these questions required inter-disciplinary work in the cross-section between machine learning, computer vision and applied field robotics, which resulted in the five papers included in this thesis. The thesis is organised as follows: After this introduction, there are three background chapters that present the necessary fundamentals and position the work of this thesis with respect to the relevant research fields. The first covers different topics related to the agri-robot application, before we move on to the fundamentals of machine learning in Chapter 3. In Chapter 4, we discuss learning-based methods in practice, and present state-of-the-art on two topics: Learning-based robot guidance and interpretation of deep neural networks. Then, short summaries of the scientific papers of this work are presented in Chapter 5 together with the overall scientific contribution of this work. Finally, the papers are attached in their original version.

CHAPTER 2

AGRICULTURAL ROBOTICS

This background chapter is a bundle of topics related to the application of agricultural robots. It gives an overview of different agri-robot applications, before looking into different sensing strategies for robot guidance in such applications, where we dive deeper into traditional methods for vision-based crop row detection. Then, we briefly cover the basics of robot and camera geometry and describe the robot platform used for the work in this thesis.

2.1 Agricultural robot applications

Today, different agri-robot platforms are deployed in a wide range of applications, both commercially and for research. In large open fields, Unmanned Aerial Vehicles (UAVs) give a good overview and cover more ground in less time than conventional tractors and are used for spraying and monitoring. At the other end of the scale, mobile robots are used for precision-weeding, targeting each weed with pesticides or mechanical removal instead of large-scale spraying. Robots harvesting high-value crops like capsicums and strawberries are also under development, which could replace the huge amount of seasonal human labour this requires today. Fleets of robots can also be used for transport and other tasks to aid the human workers and make logistics on the farm more efficient.

The Thorvald robot platform (see Figure 2.1) from Saga Robotics¹ is one such agri-robot platform, which is in use in many different environments, both for research and commercial applications. The most successful commercial application so far is UV-treatment of berries and fruits to control mildew fungi². This operation is performed autonomously in strawberry polytunnels, greenhouses with rails between rows of cucumbers and tomatoes, and vineyards. Variations of the same platform are used in a wide range of other applications, for instance to collect data for efficient phenotyping in cereal fields [Burud et al., 2017], and for transport during broccoli harvest, greatly improving the working conditions for the human pickers. A gripper for automatic strawberry picking is currently undergoing pilot testing, which will reduce the need for human labour while enabling precision measurements of the quality of the berries. The Thorvald robot platform is designed to be easily re-configurable [Grimstad and From, 2017] in terms of mechanics and robot control, but the wide

¹<https://sagarobotics.com>

²<https://www.morningagclips.com/uv-system-means-lights-out-for-strawberry-pathogen/?fbclid=IwAR115M0Fc6fzvzt4PCsgn1QDer0whSQmbuY9n2Ak41m3G1leXrlc6fiqn-c>

2. Agricultural robotics



Figure 2.1: Example applications of the re-configurable Thorvald Robot platform. From top, row-wise: strawberries in field, broccoli in field, strawberries in polytunnel, tomatoes in greenhouse. Navigation sensors are highlighted with circles.

range of applications and environments require a myriad of different sensor setups, which results in large significant development costs and risk for new applications.

There is virtually no end to the variation in environments for agricultural applications, which makes it challenging to design a one-system-fits-all for navigation. Agricultural environments come with a wide variety of appearance and complexity, as indicated with the examples in Figure 2.1 and Figure 2.2, and may change drastically from season to season. In general, the environments have fewer straight and massive structures than an urban or indoor environment, but within each application there may be lower complexity and variation than what you would see in a city. Most agricultural fields have a row organisation, which provides structure useful for guidance, but the appearance is often less uniform and predictable than roads.

2.2 Guidance of agricultural robots

Localisation and guidance of autonomous platforms is a huge and rapidly growing field of research, spanning from self-driving cars to vacuum cleaner robots to tiny UAVs.



Figure 2.2: Example agricultural robot platforms with different navigation solutions. From top left: 3D row detection with BoniRob [Biber et al., 2012], vision-based (RGB) row detection in [Ahmadi et al., 2019].

In this context, guidance of agricultural robots has its own set of challenges specific to agricultural environments. In this section, we will give a brief overview of some agri-robot applications and different sensing strategies for navigation of agri-robots, to motivate why we focus on vision-based navigation in this work.

2.2.1 Sensors for navigation

To achieve full autonomy, agri-robots usually need to navigate both on a global scale, i.e. to drive from A to B and plan the route between rows, and a local scale, i.e. to stay in the right position relative to the plant row to perform the designated task. These two levels of navigation require different kinds of sensing strategies for localisation. Pure global localisation can be a successful strategy in environments that does not change over time. Relative localisation can typically provide a higher relative accuracy at a lower cost and can adapt to changes in the environment or position the robot relative to specific parts of the crops for direct interaction. Each agri-robot platform typically uses a wide range of different sensors to cover the whole spectrum of navigation tasks for different applications.

Different sensor options for global and relative localisation are summarised in Table 2.1 and is discussed further below; First sensors and strategies for global localisation, and then optical sensors that can be used to extract information about the environment for both global and relative localisation strategies.

Global Navigation Satellite System (GNSS)

Positioning with a Global Navigation Satellite System (for instance GPS or GLONASS), provide a position in a global reference system. GNSS only works in open areas with line-of-sight in the direction of the satellites, as it estimates the position based on transmission time differences between the receiver and four or more different

2. Agricultural robotics

Table 2.1: Properties of different sensors used for navigation. Above line: Global positioning sensors. Below: Optical sensors that can be used for both global and relative localisation.

Sensor	Cost	Accuracy	Resolution	Suitable environment
GNSS	low	m	-	Open, outdoor
RTK-GNSS	high	cm	-	Open, outdoor
IMU	low-mid	(varies)	-	Any
Scanning LIDAR	mid-high	cm	low	Any
Passive depth camera	low-mid	cm-dm	high	With texture and good light conditions
Flash LIDAR	mid-high	cm-dm	mid	Any
Active depth camera	low-mid	cm-dm	mid	Indoor/ limited outdoor
RGB cameras	low	cm-dm	high	Any

satellites. GNSS receivers have become common in all kinds of mobile devices, which has reduced the hardware cost and increased the position update rate, but regular receivers have a limited accuracy (around 1-2 m horizontally and 3-4 m vertically)³, which makes it more suitable for road navigation than precise robot steering. Precision can be increased to the mm-range by averaging over a long time, but that requires a stationary receiver. Another option is called real-time kinematic (RTK) positioning, which utilises the phase difference relative to a fixed base station, to improve accuracy to around 1 cm. RTK-GNSS require more expensive receivers and line-of-sight to an established base station that can provide correction signals. Correction data can also be bought as a service, (for instance CPOS⁴, which eliminates the need for a separate base station. RTK-GNSS is a good choice for navigation in open fields, but like normal GNSS it is unreliable in covered environments like greenhouses and polytunnels, or close to tall buildings. GNSS cannot provide attitude information directly, but this can be achieved with a setup with multiple antennas. The configuration of the dual-antenna setup in the upper left corner of Figure 2.1 can provide the heading and pitch angle of the robot.

The main limitation with GNSS, as with other global localisation strategies is that it only provides a global position, and no information about the surrounding world. The acquired global position must somehow be related to the robot's environment, for instance with a geo-referenced map or pre-programmed route, which is not always available for a field. Steering tractors steered with GNSS is already an established technology, and when GNSS is used from the beginning with planting or seeding, the tractor can successfully do many routine operations on GNSS-steering only.

³<https://www.gps.gov/systems/gps/performance/accuracy/>

⁴<https://www.kartverket.no/en/on-land/posisjon/guide-to-cpos>

Inertial Measurement Unit (IMU)

Almost every mobile robot carries an Inertial Measurement Unit, which estimates relative six-degrees-of-freedom (6-DOF) positions by integrating angular velocity and acceleration measured by several accelerometers and gyroscopes [Barfoot, 2017]. Again, as the sensor is common in mobile devices, the hardware cost is low and speed is high, but the accuracy varies with price. Another drawback of IMUs is that the integration process accumulates errors, and the pose estimates will usually drift over time. IMUs are typically combined with other sensors to get absolute position and limit drifting.

Scanning LIDAR

Light detection and ranging (LIDAR) perform point-wise range measurements with lasers and comes in several different forms. Traditional scanning LIDARs have a mirror that moves the laser to obtain beams of point measurements, with a very wide field-of-view. The density and speed of the point cloud vary with the price of the scanner. Scanning LIDARs have a long range and are robust to illumination changes and strong sunlight and are well suited for outdoor use. LIDARs have been applied on several agri-robot systems, for instance in [Le et al., 2019] which use it to build a full 3D map to navigate between buildings and fields on a farm, while [Biber et al., 2012] perform crop row detection with the 3D data. However, the resolution is much lower than for a camera, and it requires distinct structures in the surroundings to give features that can be useful for localisation.

Depth cameras

Depth cameras provide high-resolution depth images, which can be obtained using several different technologies, which are roughly divided into active and passive sensors.

A stereo camera is a passive technology, where regular cameras are mounted as a stereo pair. The depth image is constructed by matching visual features in the different camera views. One limitation of the stereo camera is the range, which is shorter for small baselines. Stereo requires good light conditions and texture to function properly, which is usually the case in agri-cultural settings in daytime. [Stefas et al., 2016] demonstrated successful use of a stereo rig for row navigation in apple orchards with a UAV.

Active sensors use an additional light source (often infrared) to obtain the depth image. Since it does not depend on the stereo feature matching, it can provide depth data on any surface. One such technology is Time-of-flight cameras (ToF), which use the same principle as the LIDAR scanners, but have higher resolution and give a full 2D depth image in one go. Some sensors use a combination of active and passive sensing, like the Intel RealSense D435, which project an IR pattern to assist the stereo in areas without good features. Active depth sensors have been popular

2. Agricultural robotics

for indoor robotics applications, but the light source is usually overpowered by the sunlight, and does not give reliable depth measurements outdoors. One exception is the recent Flash LIDAR technology that uses a different frequency than the typical ToF cameras.

RGB cameras

Regular RGB cameras are lightweight, low-cost and versatile sensors that can be used in several ways in the context of robot navigation. RGB cameras work in most conditions and have a higher resolution and frame rate than LIDAR and depth cameras, which make them a versatile sensor for navigation. The 2D images from a regular camera contain a lot of information but need further processing to extract features related to the 3D world that can be useful for navigation. Like stereo cameras, RGB cameras require good light conditions to provide useful information. Infrared cameras can provide images in low-light conditions that can be processed in a similar way as RGB images.

2.2.2 Localisation strategies

Data from the optical sensors mentioned above has to be processed further to get more high-level information that can be useful for localisation or guidance of a robot.

Simultaneous Localisation and Mapping (SLAM)

Simultaneous Localisation and Mapping (SLAM) systems build a map from detected key-points or features while simultaneously localising the camera pose in that (global) map. The features can be extracted from any of the optical sensor modalities mentioned above; LIDAR point clouds, depth images or regular RGB images. To use SLAM for navigation, a full mapping of the environment must be performed during the setup of the system, which can be time-consuming for large fields. The various forms of SLAM are commonly used in many mobile robot applications, and LIDAR-based SLAM has been successfully used on the Thorvald platform in polytunnels and greenhouses. It works best for overlapping robot trajectories and can suffer from coordinate drift and false matches in the self-similar environment of agricultural crop rows.

Crop row detection

A common localisation strategy for agri-robots is vision-based row detection. This can provide relative positioning with respect to any crop row with good precision. One of the most recent implementations on a robot [Ahmadi et al., 2019] (see Figure 2.2) combine this with visual servoing for path following. The main drawback with the traditional row detection methods is that the visual feature extractor must be tailored for every application. This is where modern methods based on deep learning can

provide a more streamlined adaptation, by learning the visual appearance directly from examples.

Learning-based row following with RGB cameras can potentially provide guidance in any type of agricultural environment with a low-cost sensor, and we believe it has a great potential to provide general low-cost guidance of autonomous agri-robots. Therefore, this is the chosen strategy for the work in this thesis. To give a more extensive overview of the related work on this topic, traditional methods are presented below, and learning-based strategies are discussed in Chapter 4.

2.2.3 Classical methods for vision-based crop row detection

Vision-based crop following has been a research topic for decades, and several methods have been successfully demonstrated for real applications using tractors and robots.

Traditionally, this pipeline typically consists of two main operations: First, *segmentation* is performed to convert the camera image to a binary *mask* separating plants from the background. Based on this, a line or path representing the row is computed.

Using traditional computer vision methods, the segmentation is typically obtained by computing some form of *vegetation index*, followed by *thresholding* to produce a binary segmentation mask. A simple and widely used vegetation index is the *Excess greenness index (ExG)* [Woebbecke et al., 1995], which is defined as

$$ExG = 2g - b - r \tag{2.1}$$

where r, g, b are the values of the R, G and B channels (in arbitrary units), normalised for each pixel:

$$r = \frac{R}{D}, g = \frac{G}{D}, b = \frac{B}{D}, \quad D = R + G + B \tag{2.2}$$

Note that $r+g+b=1$. This gives a one-channel image with high values (up to 2) when green dominates the pixel for $g > \frac{1}{3}$, and negative values (down to -1) when $g < \frac{1}{3}$. Example ExG images for a strawberry field are shown in Figure 2.3. In the ideal case (first row in Figure 2.3), this gives two distinct peaks in the histogram that can be separated with a threshold value, to produce a binary segmentation. This can be done using Otsu’s method [Otsu, 1979][Gonzalez and Woods, 2007, ch. 12], which chooses a threshold based on the maximum intra-class variance. This method works best for bimodal distributions (i.e. similar amount of background and foreground pixels) with a sharp “valley” in between. For the examples in Figure 2.3, the segmentation is successful for the cases with tidy hay-covered lanes and full green plants, but when lanes are overgrown with green shoots, or the plants turn red and brown in autumn, the ExG does not provide well-separated peaks in the histograms, and the segmentation is less successful. To get a cleaner segmentation, *morphological* operations [Gonzalez and Woods, 2007, ch. 9] are used to fill in gaps and remove noise.

2. Agricultural robotics

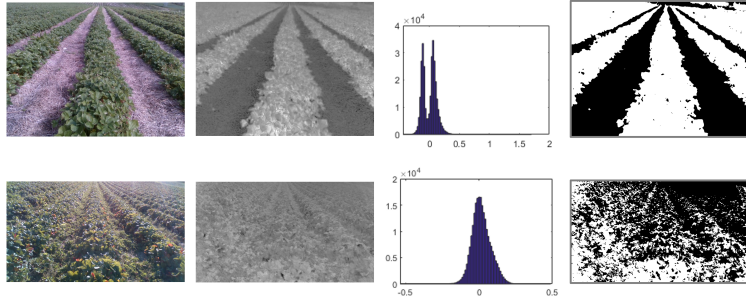


Figure 2.3: Segmentation based on greenness index on example images from a strawberry field (performed with MATLAB’s *graythresh* implementation of Otsu’s method). The columns show: 1) Image, 2) ExG image (enhanced for visibility), 3) ExG histogram, 4) Binary segmentation mask after thresholding.

Based on the binary segmentation masks, there are several approaches for line/path estimation, depending on the problem at hand. For thinner crops, the challenge is typically to extract a path through non-connected plant regions, while for fuller crops like the strawberry plants in Figure 2.3 the challenge is to find a path along the middle of one thick region. One option is to use the Hough Transform [Hough, 1962][Gonzalez and Woods, 2007, ch], a well-known method for line detection used for crop row detection in for instance [Marchant and Brivot, 1995, Åstrand and Baerveldt, 2005]. Every pixel in the binary image is transformed to Hough Space $H(s, \alpha)$ where s and α is the position and angle of the line. In essence, all pixels in the image that belongs to one line will accumulate values at the same point in Hough space, so lines can be found by thresholding this accumulator.

One drawback with Hough-based line detection is that the transform is performed on every single “True” pixel in the binary mask, which would result in slow execution and too many line candidates for the thick rows in the strawberry field. Another option is to extract *feature points* and perform regression to fit a line or (or polynomial for curved rows, as in [García-Santillán et al., 2018a]. The feature points can for instance be extracted by processing one horizontal strip at a time, but the exact technique used tend to vary based on the application in question. Common challenges are false positives caused by high weed pressure and large gaps between plants that disconnect the lines.

Current state-of-the-art in vision-based crop row detection, e.g [García-Santillán et al., 2018b, Zhang et al., 2018] builds on years of research that has optimised every single step of the process. In addition to the techniques described above, the 20-step algorithm in [Zhang et al., 2018] includes a modified vegetation index, clustering,

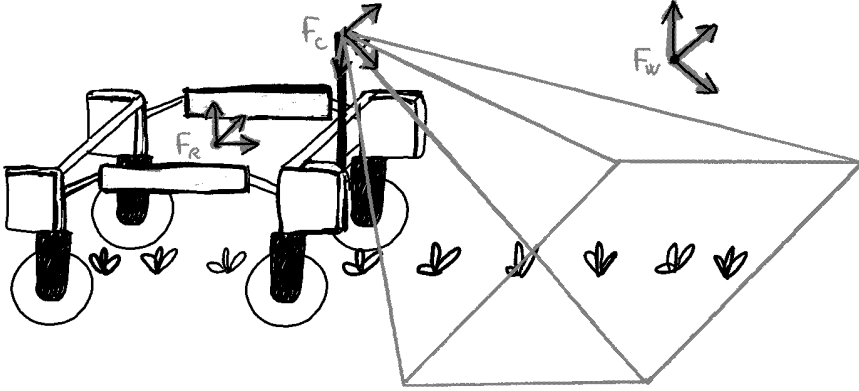


Figure 2.4: Some useful coordinate frames for a robot driving in a field with a camera: The static world frame F_W , the robot frame F_R , and the camera frame F_C .

start point extraction and a shortest path method to connect the points. There have been a few recent works that propose to simplify row detection with learning-based methods, which we will come back to in Chapter 4.

2.3 Robot and camera geometry

Since we want to use crop row detection as a means of localising and steering the robot in the field, we need to relate detected crops in the camera image to the world the robot is driving around in. This is described through *geometric reference frames* and *camera view geometry*. These two topics lay the foundation for the label generation through virtual camera views in Paper I, II and V, and the automatic mask projection in Paper III and V which is described in more detail in Appendix B.1.

This section only touches basics, and the reader is referred to [Hartley and Zisserman, 2003] and [Barfoot, 2017] for a more detailed treatment of these topics.

2.3.1 Transformations of reference frames

Consider a moving agri-robot with a camera tilted downwards as illustrated in 2.4. The configuration or state of a mobile robot in the three-dimensional world is often called the *pose* and has six degrees of freedom (DOF): three in position and three in rotation. The pose can be described as the position and rotation of the (moving) robot reference frame F_R relative to a (static) world reference frame F_W , as illustrated

2. Agricultural robotics

in Figure 2.4. Let the point of interest P be expressed with a vector \mathbf{v}_W in the world coordinate system, that can be written in *homogeneous* coordinates by appending an additional element of 1, $\mathbf{v} = [v_x \ v_y \ v_z \ 1]^\top$, such that it can be multiplied with a 4×4 matrix and be transformed to the robot reference frame [Barfoot, 2017, ch. 6]:

$$\mathbf{v}_R = \mathbf{T}_{WR}\mathbf{v}_W. \quad (2.3)$$

The *world-to-robot-transform* \mathbf{T}_{WR} is a *homogeneous transformation matrix*, composed of a 3×3 rotation matrix \mathbf{R} and translation vector $\mathbf{t} = [t_x, t_y, t_z]$:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t}^\top \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (2.4)$$

When several reference frames are involved, as for the moving robot in Figure 2.4, the transformations can be chained to obtain the *world-to-camera-transform* and express the point of interest in camera coordinates:

$$\mathbf{v}_C = \mathbf{T}_{RC}\mathbf{T}_{WR}\mathbf{v}_W = \mathbf{T}_{WC}\mathbf{v}_W \quad (2.5)$$

In practice, the *robot-to-camera transform* \mathbf{T}_{RC} is determined through calibration. Estimating the *world-to-robot transform* T_{WR} is the task of the robot localisation system. To perform a transformation in reverse, one can simply apply the matrix inverse. Thus, a point in camera coordinates can be changed to world coordinates like this:

$$\mathbf{v}_W = \mathbf{T}_{CW}\mathbf{v}_C = \mathbf{T}_{WC}^{-1}\mathbf{v}_C \quad (2.6)$$

2.3.2 Camera view geometry

The relationship between a coordinate $\mathbf{p} = [x_I, y_I, 1]$ in the image reference frame and view vector in the camera reference frame \mathbf{v}_C is approximated with a mapping called the *camera model* which can capture different levels of complexity and non-linear effects. We will start with the simple linear model of a *pinhole camera*, also called the *rectilinear* model, as illustrated in Figure 2.5, where the projection of the 3D view vector into the 2D image plane gives

$$\begin{aligned} x_I &= f \frac{x_C}{z_C}, \\ y_I &= f \frac{y_C}{z_C} \end{aligned} \quad (2.7)$$

where the focal length f is the distance from the optical camera centre to the image plane. This determines the *field-of-view* (FOV) of the camera, i.e. the angle between the outermost view vectors. The horizontal field-of-view is $\text{HFOV} = 2 \tan(\frac{w}{2f})$, where w is the width of the image sensor, and similarly for vertical FOV with the height of the sensor. Additionally, it is common to account for an offset between the camera

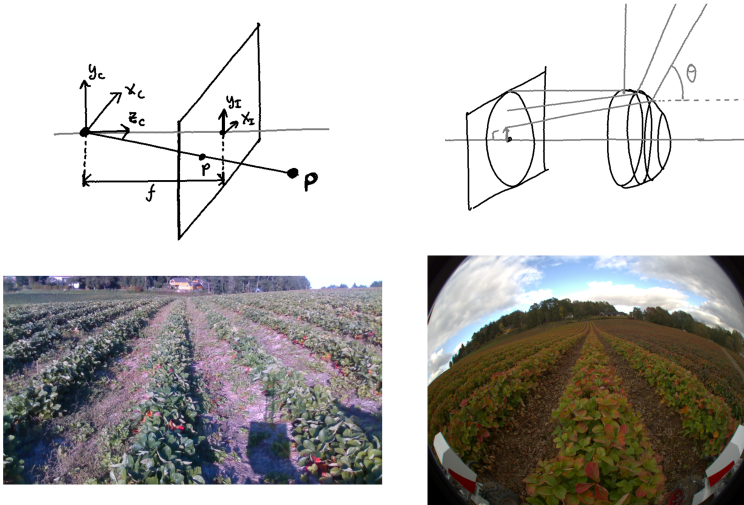


Figure 2.5: Illustration of rectilinear and fisheye projection models, and corresponding example images.

centre and the origin of the image plane, denoted by $c = (c_x, c_y)$. If the optics and the sensor is perfectly aligned, this is usually given by $c = (\frac{w}{2}, \frac{h}{2})$ where w and h are the width and height of the image sensor. Combining these effects into a homogeneous transformation matrix, we define the 3×3 (*intrinsic*) camera calibration matrix [Hartley and Zisserman, 2003] as

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.8)$$

By appending an additional column of zeros to get a 3×4 projection matrix \mathbf{C} , the transformation from 3D camera to 2D image coordinates can be expressed as

$$\mathbf{p} = [\mathbf{K}|\mathbf{0}]\mathbf{v}_C = \mathbf{C}\mathbf{v}_C, \quad (2.9)$$

which can be chained with the transformation matrices above to transform a vector all the way from world frame to image frame:

$$\mathbf{p} = \mathbf{C}\mathbf{T}_{RC}\mathbf{T}_{WR}\mathbf{v}_W \quad (2.10)$$

2. Agricultural robotics

2.3.3 Spherical projection

Omnidirectional lenses, or so-called *fish-eye* lenses have a very wide FOV, and can not be modelled with the linear projection above. Fish-eye lenses can be approximated by projection in spherical coordinates onto the flat image sensor, as illustrated in Figure 2.5.

$$p' = \begin{bmatrix} \frac{r}{\sin(\theta)} x_C \\ \frac{r}{\sin(\theta)} y_C \end{bmatrix}, \quad (2.11)$$

where $p' = (x'_I, y'_I)$ are centred image coordinates, $r = \sqrt{x_I^2 + y_I^2}$ is the distance from the image centre and θ is the angle compared to the optical axis. For an ideal fisheye, the relationship between r and θ is linear and can be parameterised with a constant α , corresponding to *radians per pixel*.

2.3.4 Distortion

So far, we have assumed a perfect linear projection, but for a real camera, the projection will be subject to distortions in the lens. A correction is usually performed to bring the camera model back to the rectilinear case, such that the previous equations can be used.

The most common non-linear effect is radial distortion, which is most prominent for cameras with wide field-of-view and/or short focal length. This is modelled by a non-linear function usually approximated by a Taylor expansion

$$f(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \dots \quad (2.12)$$

where r is the distance between the pixel and the image centre. The number of *distortion coefficients* $\kappa_1, \kappa_2, \dots$ are usually limited to 3 or 4.

2.3.5 Camera calibration

The parameters in Equation (2.8) and Equation (2.12) are estimated through intrinsic camera calibration. This can be performed in several ways, but the key idea is to detect features on an object where the spatial relationship between the features is known, for instance the corners of a chessboard. Collecting several images with different object placements, Equation (2.9) can then be solved through numerical optimisation. See e.g. [Hartley and Zisserman, 2003, ch. 7] for more details. For the work in this thesis, the OcamCalib Toolbox [Scaramuzza et al., 2006] was used for camera calibration.

2.4 Experimental robot setup

In four of the papers in this thesis, an agri robot platform was used for data collection and field trials. This section gives a brief overview of the robot and sensor setup, and the agricultural environments that were used for experiments in these papers.



Figure 2.6: Robot setup for field data collection and experiments. Left: The Thorvald robot platform configured for a strawberry field, as used in Paper III and Paper V with RTK-GNSS antennas (pink), fisheye camera (yellow) and Intel RealSense D345 (depth was not used for this work). Right: The Thorvald robot platform configured for a strawberry polytunnel, as used in Paper I and Paper II with fisheye camera (red) and LIDAR laser scanner (blue).

The Thorvald robot platform was used for data collection, configured for two different field types: One for open strawberry fields with crops on the ground, and one for strawberry *polytunnels* with crops on tabletops, as shown in Figure 2.6. In the strawberry field, the robot was driven manually during data collection, while in the strawberry polytunnel the robot was driving autonomously based on data from a laser scanner. The robot platforms have onboard computers that are running ROS⁵, and the camera and positioning data was recorded to rosbags.

Data was recorded over four summer seasons in different tunnels and rows at different growth stages and light conditions, as illustrated with some examples in Figure 2.7. When constructing datasets for machine learning, the camera stream was subsampled to avoid too much overlap between frames, and entire rows were reserved for the test set and not seen during training.

⁵Robot Operating System, <http://wiki.ros.org/>

2. Agricultural robotics



Figure 2.7: Example appearance from the collected datasets. Upper: strawberry fields. Lower: strawberry polytunnels.

CHAPTER 3

MACHINE LEARNING FUNDAMENTALS

As we saw in the previous chapter, traditional methods for crop row detection work well in many cases, but consist of many steps that often have to be tailored for each new application. Therefore, we have investigated learning-based methods, which can learn any type of feature from the data itself. This chapter will go through the necessary background in machine learning; It starts with some basic terminology, before we move on to the building blocks of modern deep neural networks for computer vision, and look into strategies for learning with limited data.

3.1 Learning a model

In machine learning, we attempt to *learn* a *model* from *data* that can predict the outcome of a variable based on some *features* [Hastie et al., 2009, Andrew Ng et al., 2000]. This model can range from a simple linear regression with a bias and slope, to deep neural networks with millions of weights.

We use the following notation:

$$\hat{y} = h_{\theta}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^p, \quad (3.1)$$

where \mathbf{x} is the input variable with p features, \hat{y} is the prediction of the output value or *response* y , and h is the model with parameters θ .

The parameters of the model are *learned* by optimising an *objective*, using *training data*. In the *supervised learning* case, the input data samples are accompanied by the correct output value, often called *labels* $\{\mathbf{x}_i, y_i\}, i = 1, \dots, N$, which can be used to fit the parameters of the model. In the *unsupervised case*, only \mathbf{x} is given and there are no ground truth labels to guide the learning. For the remainder of this thesis, we will consider supervised learning if not stated otherwise.

There are two main types of prediction tasks, depending on the nature of the output: *Regression* and *classification*. *Regression* is the task of predicting a quantitative output, like the price of a car, or the angle of a crop row in an image of a field. In regression, the objective is to get as close to the ground truth values as possible. A typical *objective function* (also called *cost*, *loss* or *error*) for regression is the mean squared error (MSE) overall N samples,

$$L(\theta) = \text{MSE}(\theta) = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(\mathbf{x}_i) - y_i)^2, \quad (3.2)$$

which is minimised to find the best parameters. *Classification* on the other hand, is the task of predicting a qualitative or *categorical* output, which only can take the

3. Machine learning fundamentals

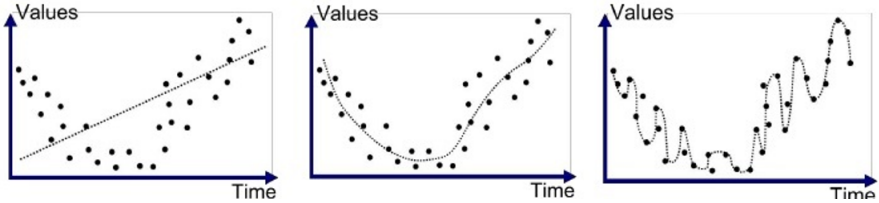


Figure 3.1: Illustration of overfitting and underfitting in machine learning when fitting a line to a set of data samples. From left: 1) Underfitting with a too simple model, 2) good fit with a model of appropriate complexity, 3) overfitting with a model of too high complexity. Image courtesy: Anup Bhande¹

values from a discrete set of K classes, like dog breed or whether the crop row is to the left, right or straight ahead. Here, the goal is to separate the classes in a way that minimises the number of erroneous classifications. A typical choice of loss function for classification problems is the *categorical cross-entropy* loss

$$L(\theta) = \sum_j -\log(\sigma(h_\theta(\mathbf{x}_j))), \quad (3.3)$$

where σ is the softmax function:

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, \quad j = 1, \dots, K. \quad (3.4)$$

which force the output to be close to 0 or 1, which gives a better class separation than linear or quadratic loss.

We want to find the model that gives the best fit or the best class separation for the problem at hand. However, a model with a high degree of freedom or *capacity* may produce a very small error on the training samples, but fail to describe the overall relationship. This phenomenon is called *overfitting* and is illustrated with an example in Figure 3.1. The real goal in machine learning is to *generalise*, i.e. find a model that also fits well with samples that are not in the training data.

Therefore, it is the expected prediction error on the *test data*, also known as *test error* or *generalisation error*, that should be minimised. This error can be decomposed into three terms [Hastie et al., 2009, ch. 7] for a test sample x_0 :

$$\mathbb{E}[y - \hat{h}_\theta(x_0)^2] = \text{Bias}^2(\hat{h}_\theta(x_0)) + \text{Var}(\hat{h}_\theta(x_0)) + \sigma^2. \quad (3.5)$$

where \hat{h} is the estimated model, which varies based on the choice of the training set. The variance term indicates how much the prediction varies with the choice of training samples, and the squared bias term is how much it deviates from the true mean. σ^2 is the variance of the noise in the test data, which cannot be reduced even with the perfect model. Typically, a model with a low degree of complexity

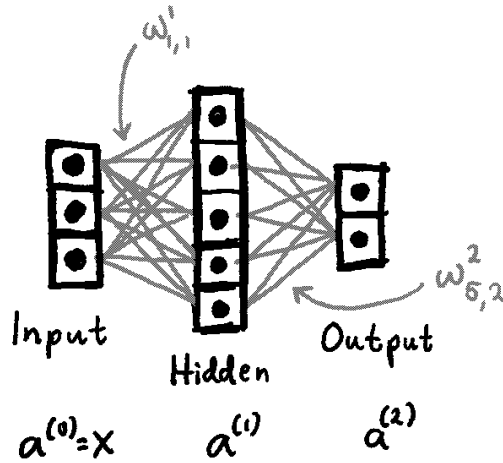


Figure 3.2: A two-layer neural network (it is common not to count the input layer) with one hidden layer of 5 units and one output layer of 2 units.

gives a low variance and high bias, *underfitting*, while a model with a high degree of complexity results in low bias and high variance, *overfitting*. There is always a trade-off between bias and variance to minimise the test error when choosing the model complexity (number of parameters) and fitting the model parameters.

There is a huge variety in model types that has been proposed within machine learning over the years, with different properties when it comes to model complexity and bias-variance trade-offs. For the remainder of this work, we will focus on one particular family of models, namely *neural networks*.

3.2 Neural networks

Neural networks are a family of models that have existed for decades and exploded in popularity with the breakthrough of deep learning. The building blocks of neural networks are simple functions that create models of very high learning capacity when stacked together.

In its most general form, the neural network model is a cascade of *layers* transforming the input:

$$\mathbf{h}_\theta(\mathbf{x}) = f_{\theta_L}^L(\dots f_{\theta_2}^2(f_{\theta_1}^1(\mathbf{x}))), \quad (3.6)$$

3. Machine learning fundamentals

where $f_{\theta_l}^l$ is the function representing layer $l = 1, \dots, L$ with parameters θ_l . The simplest neural network is a *feed-forward neural network* [Goodfellow et al., 2016, ch. 6], [Andrew Ng et al., 2000, Karpathy, 2017] with *fully-connected* layers as shown in the schematics of Figure 3.2. Each *unit* in a layer, or *neuron*, is a cascade of two functions: a linear combination of the outputs from the previous layers, followed by a (non-linear) activation function σ . For the first hidden layer, the expression for the output of unit j is

$$a_j = \sigma(\mathbf{w}_j^\top \mathbf{x} + b_j), \quad (3.7)$$

with parameters $\theta = (\mathbf{w}, b)$, *weights* $\mathbf{w} \in \mathbb{R}^d$ and *bias* $b \in \mathbb{R}$. The output of a fully-connected layer l is a vector with activations of all the n_l individual units, which acts as the input to the next layer. This can be written as a recursive expression:

$$\mathbf{a}^{(l)} = \sigma(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, \dots, L \quad (3.8)$$

where $\mathbf{W}^{(l)}$ is a matrix of size $n_l \times n_{l-1}$, the input is $\mathbf{a}^{(0)} = \mathbf{x}$, and the final output of a network with L layers is $\mathbf{h}_\theta(\mathbf{x}) = \mathbf{a}^{(L)}$. The number of parameters in a fully-connected network (neglecting the bias) is $\sum_{l=1}^L n_l n_{l-1}$, i.e. it depends on the input size $n_0 = d$, the number of units in each layer, and the number of layers. In practice, going deeper than 3 layers for regular fully-connected networks does not increase performance [Karpathy, 2017].

There are several options for the activation function $\sigma(z)$ [Goodfellow et al., 2016, ch. 6], [Karpathy, 2017]. For the hidden layers, this must be a non-linear function, or the network would just produce a linear combination of the inputs. Common choices are the sigmoid function, forcing the output to be close to 0 or 1,

$$\sigma(z) = 1/(1 + e^{-z}). \quad (3.9)$$

More common for deep neural networks is the simpler Rectified Linear Unit (ReLU),

$$\sigma(z) = \max(0, z), \quad (3.10)$$

which sets all negative values to zero. For classification, the output layers consist of K units giving the probability of class K , usually computed by a softmax activation

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, \quad j = 1, \dots, K \quad (3.11)$$

which acts in a similar way as the sigmoid does for the binary case. For regression, the final output is usually computed with a linear activation.

3.3 Convolutional neural networks

A regular fully connected network is not a very good choice for learning image representations. The all-to-all connectivity causes two main issues: 1) The number of parameters does not scale well with input size. For instance, an RGB image with

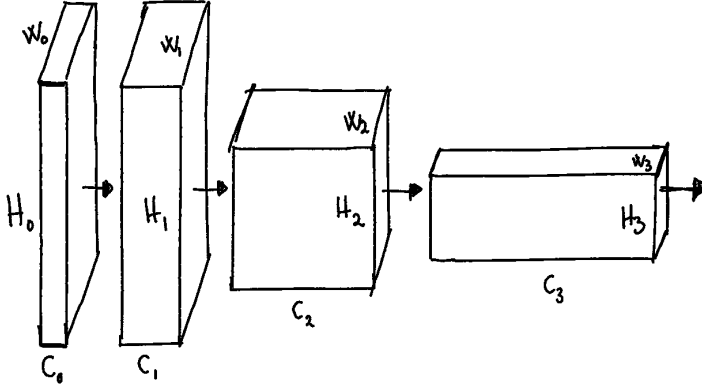


Figure 3.3: Illustration of a three-layer CNN (input layer not counted), with units arranged in 3D blocks of size $C_l \times W_l \times H_l$.

100×100 pixels gives 30000 weights for only *one* unit in the first hidden layer. This is computationally infeasible, and also leads to overfitting. 2) Connections between neighbouring pixels are not explicitly prioritised, which makes it hard to learn even simple image features. To fix these issues, we use *convolutional neural networks* (CNNs) [Goodfellow et al., 2016, ch. 9], [Karpathy, 2017], that implement weight sharing and two-dimensional neighbourhood connectivity.

The units in a convolutional layer are stacked in a three-dimensional structure, with size $C \times W \times H$, where W and H are the width and height of the spatial dimension, and C is the number of channels, or layer depth, as illustrated for a three-layer CNN in Figure 3.3.

As illustrated in Figure 3.4, the pre-activation output of one unit at position (i, j) in a convolutional layer can be computed as a convolution of the output of the previous layer with a filter kernel. The expression per input channel c is

$$\mathbf{z}'^{(l)}(i, j) = (\mathbf{W}^{(l)} * \mathbf{a}_c^{(l-1)})(i, j) + b^{(l)}, \quad (3.12)$$

where the 2D convolution is defined by

$$(\mathbf{W}^{(l)} * \mathbf{a}^{(l-1)})(i, j) = \sum_{m=-k'}^{k'} \sum_{n=-k'}^{k'} \mathbf{W}^{(l)}(m, n) \mathbf{a}^{(l-1)}(i-m, j-n), \quad (3.13)$$

and the weight matrix $\mathbf{W}^{(l)}$ has a limited spatial extent of $k \times k$, where the odd-numbered k is called the filter size and $k' = \frac{k-1}{2}$. In the channel dimension, the kernel spans over the whole depth C_{l-1} of the input, and is combined through summation, as for a fully-connected layer:

3. Machine learning fundamentals

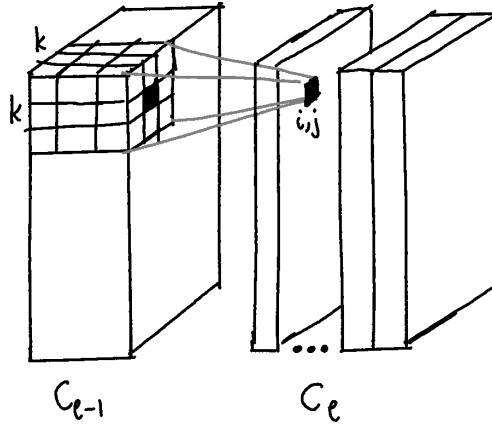


Figure 3.4: The convolution operation with a sliding kernel on a 2D input with three channels

$$\mathbf{z}_d^{(l)} = \sum_{c=1}^{C_{l-1}} \mathbf{W}_{c,d}^{(l)} * \mathbf{a}_c^{(l-1)} + b_d^{(l)}, \quad l = 1, \dots, L \quad (3.14)$$

This operation is repeated for a stack of filters $W_{c,d}$, $d = 1, \dots, C_l$ and combined with the activation function, giving an output with a depth of C_l channels,

$$\mathbf{a}_d^{(l)} = \sigma(\mathbf{z}_d^l), \quad d = 1, \dots, C_l. \quad (3.15)$$

Each channel in \mathbf{a} is called an activation map, and the whole depth is used as input to the next layer.

The number of parameters in a CNN is determined by the spatial extent of the filters, the number of channels of the layers, and the total number of layers, but is independent of the spatial size of the input, in contrast to fully connected layers. This is because the weights of the kernel are shared across the spatial dimensions. For one layer, the number of parameters is $k \times k \times C_{l-1} \times C_l$.

To limit the number of weights and enable deeper networks, it is common to reduce the spatial dimension of the layers at regular intervals. This can be done with a *max pooling layer* [Goodfellow et al., 2016, ch. 9], which performs a maximum operation over units in a small neighbourhood, typically 2×2 , separately for each channel. Although the number of parameters in a CNN is independent of the size of the input, larger inputs require deeper networks with successive downsampling to capture features of a larger spatial scale. As more layers are added, there is a gradual increase in the *receptive field*, i.e. the area of the input that is influencing each unit.

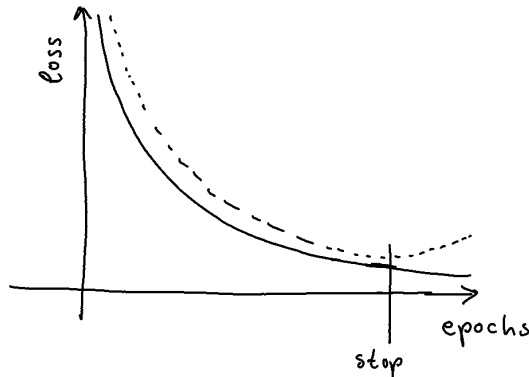


Figure 3.5: Loss as a function of epochs, for a training set (solid line) and validation set (dashed line).

3.4 Training neural networks

The optimal parameters, θ , for a neural network are computed by minimising the error compared to the ground truth, as described by the loss function L . The minimisation problem is typically solved numerically through optimisation with *gradient descent* [Goodfellow et al., 2016, ch.5]. First, a forward pass is executed with the current weights to compute the loss and the gradients for one iteration $\mathbf{g} = \nabla_{\theta} L(\theta)$. To compute the gradient with respect to each weight, the gradients are distributed during a backward pass of the network with a technique called *back-propagation* [Rumelhart et al., 1986], using the chain rule and partial derivation. To perform back-propagation, all operations in the network has to be differentiable. The parameters are updated according to the gradients $\theta \leftarrow \theta - \epsilon \mathbf{g}$, where ϵ is the *learning rate*.

Computing gradients for all samples at once is not feasible for training sets with millions of samples. In practice, *stochastic gradient descent* [Goodfellow et al., 2016, ch. 5] is often used, which estimates the gradient-based on one *minibatch* of samples at a time. When this has been repeated for all mini-batches in the training dataset, an *epoch* has been completed. Several variations have been proposed to improve convergence [Goodfellow et al., 2016, ch. 8], e.g. gradual adjustment of learning rate like RMSProp [Hinton et al.,] or AdaDelta [Zeiler, 2012], or adding momentum as in Adam [Kingma and Ba, 2015].

With a good ground truth, a network of the correct size and a suitable learning rate, the training loss should typically decay as illustrated with the solid line in Figure 3.5.

3. Machine learning fundamentals

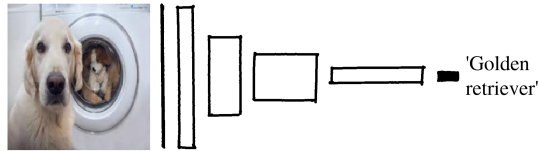


Figure 3.6: Bottleneck architecture for image classification, exemplified with VGG16 [Simonyan and Zisserman, 2014]. Each white block consists of one max pooling layer, followed by three convolutional layers of the same size and depth, with ReLU activations after each. The size of the last convolutional layer in the feature extractor is $7 \times 7 \times 512$. The decision head (black) consists of a series of two fully connected layers with ReLU activations and an output layer with softmax activation. The output is one label for the entire image.

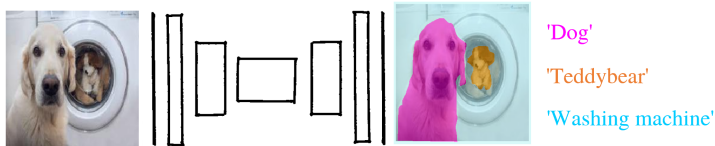


Figure 3.7: Encoder-decoder architecture for segmentation, exemplified with SegNet [Badrinarayanan et al., 2015]. The encoder is similar to a bottleneck network, while each block of the decoder is a series of three convolutional layers followed by an upsampling layer. The output is a per-pixel mask with class labels.

3.5 Deep network architectures

In image recognition, the task is to assign a class value to the content of the whole input image. For this task one would typically use a so-called *bottleneck* architecture, as illustrated in Figure 3.6 with the well-known VGG16 [Simonyan and Zisserman, 2014]. It starts with a high-resolution input with few channels, and through a stack of convolutional layers and pooling gradually reduces the resolution and increases the number of channels. The early layers typically represent simple large-scale features like colour and edges, while the final layers can represent more complex features like leaves, grass, fur, and eyes. The output of this *feature extractor* is fed into a *decision head*, typically a series of fully connected layers followed by softmax activation, which outputs probabilities of all the classes. Note that for this task, there is only one output per image in the input batch.

In *segmentation*, on the other hand, the task is to assign a class value *per pixel* in the input image, which gives a much richer output. The output could be a mask separating different object classes in the image, as shown in Figure 3.7, or pixels with plants vs. pixels with soil. This problem is typically solved with a so-called *encoder-decoder* architecture, which uses a similar feature extractor as for image recognition, but replaces the decision head with upsampling layers that increase the resolution to produce the per-pixel output. This is illustrated in Figure 3.7 with the

well-known SegNet [Badrinarayanan et al., 2015] architecture.

One of the main shortcomings with the basic CNN architectures described above is the loss of resolution for the deeper layers. This is why more recent architectures like [He et al., 2015] and [Huang et al., 2017] implement different routing strategies to combine information from earlier layers in the final decision, which has improved the classification accuracy on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015]. Although these architectures have more intricate connections, the overall principle is still the same. Similarly, segmentation architectures route the information from high-resolution early layers to increase the precision on the borders of the segmentation. SegNet [Badrinarayanan et al., 2015] transfers pooling indices to the decoder to improve the resolution. Later, more intricate routing and multi-resolution pooling have been proposed in for instance RefineNet [Lin et al., 2016] and Deeplab [Chen et al., 2017].

Another shortcoming of CNNs is the lack of (global) spatial information. The convolutional filters save a lot of parameters by sharing weights, which makes them spatial invariant. This is great for learning general visual features but makes it hard for the network to make decisions based on location in the image, as discussed in [Liu and Frank,]. The fully connected layers do not have this weight sharing, but the 2D structure is scrambled when the input to the fully connected layers is serialised. The proposed solution in [Liu and Frank,] is to explicitly encode 2D coordinates as two extra channels for the input image and subsequent convolutional layers, giving the network the possibility to use this information when learning the features.

3.6 Regularisation

Regularisation is an important concept in machine learning that is applied to prevent overfitting. There are many different techniques, but the essence is making the problem harder to learn.

A well-established technique in machine learning is to limit the magnitude of the parameter by adding a penalty in the loss function [Hastie et al., 2009][Andrew Ng et al., 2000]. To apply this to the weights W in a layer of a neural network, we add the term $\lambda\|W\|$ to the loss. The regularisation strength λ is a hyperparameter, and $\|\cdot\|$ can be the L2 norm, L1 norm or a weighted combination. Within deep learning, it is most common to use the L2 norm which puts more penalty on large weights, on one or several of the fully connected layers. This is often called *L2 regularisation*.

A more recent technique introduced specifically for neural networks in [Srivastava et al., 2014] is to restrict the number of active neurons with *dropout*. During training, neurons and their connections are skipped with probability p . At test time, all neurons are active. This gives the effect of ensemble learning with multiple smaller networks. Dropout can be applied to one or more layers, and the dropout probability p is a hyperparameter.

One of the most important hyperparameters in training neural networks, is the epoch at which to stop the training to avoid overfitting. As mentioned earlier, the

3. Machine learning fundamentals

main goal in machine learning is to minimise the generalisation error, not the training error. As a proxy for the generalisation error, we compute the loss on a *validation set*, which is not a part of the backpropagation, and stop the training when the validation error starts rising again (i.e. the network is overfitting), as illustrated in Figure 3.5.

CHAPTER 4

MACHINE LEARNING IN PRACTICE

The potential of learning-based vision goes beyond simply replacing a traditional feature extractor with a neural network, but also comes with new challenges in terms of data collection and explainability. Now that we have established the problem of agri-robot guidance and covered the basics in machine learning, we will go a bit more in depth on topics related to machine learning in practice; How to train with limited data, how deep learning can be applied in robot guidance, and how deep neural networks can be made more explainable.

4.1 Data-efficient learning

One of the main challenges with deep learning for real-world scenarios is the amount of labelled data needed to make the training converge and generalise well for the problem at hand. We use term *data-efficient learning* [Adadi, 2021] can involve efficiency in number of samples, the amount and quality of labels, and label acquisition. This includes for instance artificial creation of more data and/or labels augmentation, transferring knowledge from data-rich domains like simulation or large-scale datasets and using semi- or unsupervised learning.

4.1.1 Augmentation

A well-established technique for increasing the number of samples is by artificially adding variation through augmentation. For the augmentation to be successful, it must produce a different response in the convolutional filters. Typical augmentation strategies for images are affine transformations (stretching, skewing and rotation), added noise and colour jitter. Successful augmentation can prevent overfitting to biases in the dataset.

4.1.2 Label quality

The amount of gradients per input image depends on the type of prediction task. During back-propagation of the neural network, there will be K gradients per input sample in a classification problem with K classes, while there is only one gradient per input sample in regression. For image segmentation, the number of gradients per image is $w \times h \times K$, where $w \times h$ is the size of the input image. The optimisation will converge faster when more gradients are available, and thus segmentation requires fewer training images than classification, and regression will require more. Therefore,

4. Machine learning in practice

inaccurate or noisy segmentation labels can contain more information per image than classification labels of higher quality.

4.1.3 Transfer learning

When training a problem with very limited data, it can be useful reuse weights trained on larger datasets in the same domain through *transfer learning* or *pre-training*. For image recognition with many object categories, it is common practice to start with a network pre-trained on ImageNet [Russakovsky et al., 2015], which contains images from common object categories, for instance dog and cat breeds. A smaller application-specific dataset could be dog vs. cat classification¹. The idea is to first train a model on the big established dataset, and then *finetune* on the small application-specific dataset by training only the last layers while keeping the earlier layers frozen. In this way, we can utilise general features trained on a big dataset, and generalise better than by training only on the small dataset. If the problems are very similar, but with different target labels, it can be sufficient to fine-tune the the last layers that perform the final prediction. For more different problems, the last convolutional layers may also need fine-tuning. For the dog and cat example, the network trained on ImageNet has probably learned all the necessary features for dog vs. cat classification by learning to separate dog and cat breeds in the big dataset, and it should be sufficient to re-train the final fully-connected layers. The drawback of this approach is that the network probably has a lot more capacity and features than necessary for this simple problem. Additionally, the original network may have overfitted to some specific features or biases in the original dataset to solve the tricky problem with many dog breeds, which is probably not present in the new dataset.

4.1.4 Alternative supervision approaches

There are some alternative approaches to fully-supervised learning with hand-labelled data.

Self-supervised learning is very broad class of methods that utilise relevant information or underlying structures to learn from unlabelled data. One example is learning visual feature extractors from various pretext tasks, often reconstructing some kind of perturbation, like predicting colour from grey scale [Zhang et al., 2016], or estimating the location of patches randomly sampled from an image [Doersch et al., 2015], and many more. The feature extractors trained on such pretext tasks can be used as pre-training for downstream tasks like object detection.

Semi-supervised learning is a class of methods that learn from a mix of labelled and unlabelled data. On such approach is called Pseudo-Label [Lee et al., 2013], where the model is first trained on labelled data, then used to predict on unlabelled data, and the most confident predictions is added to the dataset. [Beyer et al., 2019]

¹<https://www.kaggle.com/c/dogs-vs-cats/data>

explore the combination self- and semi-supervised learning in combining pretext tasks with a few labelled samples.

Data-efficient supervision is a currently a very active research topic in computer vision, which we have only touched briefly upon here. Other relevant strategies include reinforcement learning and unsupervised learning, which are both very different from methods in the supervised paradigm.

4.2 Supervision strategies for agri-robot guidance

For practical applications, it is important to consider what supervision strategy is best suited for solving the problem at hand. This is a somewhat underappreciated topic in the literature, but a few related works that are relevant for the guidance of (agri) robots are presented here.

4.2.1 *Semantic segmentation*

The most straight-forward way to apply deep learning in guidance for agri-robots is to replace the segmentation step in the traditional pipeline for vision-based crop row following outlined Section 2.2.3 with a neural network. Semantic segmentation with CNNs has been successfully applied for detection of crops and weed in monitoring and spraying applications [Potena et al., 2016, Milioto et al., 2018, Ma et al., 2019], and has also been used for crop row following in tea plantations [Lin and Chen, 2019] and strawberry fields [Ponnambalam et al., 2020]. However, such approaches require large amounts of manually labelled segmentation masks, which is expensive and impractical for agricultural applications and the wide range of variation and appearance change discussed in Section 2.2. In other domains, like autonomous driving, large datasets for semantic segmentation of urban environments has been established, which makes this approach more feasible. To our knowledge, there is no established dataset relevant for agri-robot guidance that is large enough to train and test CNNs.

4.2.2 *Self-supervised learning*

With a self-supervised learning strategy, labels are automatically generated from the input data. For semantic segmentation, there are several different approaches for automatic label generation, for instance using knowledge of the scene and camera viewpoint [Zeng et al., 2017], other sensor modalities [Wendel and Underwood, 2016, Zhou et al., 2012, Reina and Milella, 2012, Blas and Blanke, 2011], or correspondences [Larsson et al., 2019]. [Zeng et al., 2017] automatically generate a big dataset with segmentation labels for robot grasping, using knowledge of the setup and camera viewpoint. In mobile robotics, it is more common to use other sensor modalities to guide the training. [Wendel and Underwood, 2016] use a hyperspectral scanner to automatically extract training data for weed classification with RGB camera. For autonomous offroad driving applications, 3D sensors (e.g. stereo cameras or scanning

4. Machine learning in practice

lidars) have been used to initially identify and label ground and non-ground regions in matching imagery [Zhou et al., 2012]. Similar approaches have also been applied to the guidance of tractors in agricultural settings for the classification of driving surfaces [Reina and Milella, 2012] and localisation of cut plant material for automatic baling [Blas and Blanke, 2011]. These approaches all require an initial classification of 3D sensor data in order to generate training labels for the visual classifier. In Paper III, we propose a new approach for automatic label generation specifically for crop row following, that generate approximate segmentation masks based on robot position and camera projection.

4.2.3 End-to-end learning

Another approach is to train guidance *end-to-end*, i.e. learn steering commands directly from input images. This eliminates the need for hand-labelled masks, but as all the intermediate outputs are removed, it is less explainable than the segmentation approach. There are several ways to obtain the ground truth steering commands in a (semi) automatic manner, to avoid manual labelling entirely. Reinforcement learning has seen great success in learning robot control in game settings, but the domain gap to real-world scenarios can be challenging to bridge. [Sadeghi and Levine, 2016] managed to transfer image-based drone flight from simulation to indoor environments by use of heavy augmentations. Other approaches, like [Giusti et al., 2015, Loquercio et al., 2018, Dumoulin et al., 2017], phrase guidance as an image recognition problem, and learn steering angles in a fully supervised manner with an automated data collection procedure. [Giusti et al., 2015] collected a trail dataset with a rig of three cameras pointing straight, left and right. This setup automatically gives the ground truth for the yaw angle of the camera. A bottleneck CNN is trained to predict one of the three heading directions, and used for autonomous flight with a UAV along a forest trail. In a similar manner, [Loquercio et al., 2018] collect images in an urban environment with corresponding steering labels by recording the steering commands of a car. To our knowledge, such end-to-end learning approaches has not been applied to row following in agriculture in any other work. In Paper I we apply the approach from [Giusti et al., 2015] to crop row following in agriculture, and extend the labelling procedure to continuous angles through with virtual field-of-view extraction during post-processing.

4.3 Explaining deep neural networks

The black-box nature of deep neural networks is another practical problem with learning-based methods. When humans cannot understand the reason for the predictions, it is hard to debug or improve the system, and decisions may not be trustworthy. Here are a few common issues that motivate research in explainability of neural networks:

“Clever Hans” predictions. Data collection can often introduce unintended correlations that the neural network may use as a shortcut during training. [Ribeiro et al., 2016a] established a classical example, where they intentionally trained a neural network to use snow as the main feature for wolf vs. husky classification. Such errors can be hard to identify in large dataset, so how do we know if the network has made a too “clever” prediction?

Misclassification. When inspecting specific failure cases, it can be hard to tell the reason for the erroneous classification. Is there a specific part or property of the image that cause the error, and why does the model interpret it the wrong way?

Transfer learning is very common in practical applications with little data, but it can be hard to tell whether the features of the existing model fits the new data, especially when validation data is limited. Additionally, there might be artefacts or biases brought over to the new dataset that are only present in the source dataset.

4.3.1 Direct inspection of CNNs


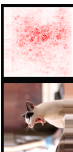




Although bottleneck CNNs are not as transparent as traditional methods, there are several properties that can be used for debugging. Listing the softmax values instead of the final classification decision gives some information about the certainty of the prediction, but the values tend to be close to either 1 or 0 for the over-confident behaviour of cross-entropy loss. Activation maps of different channels in for instance the last convolutional layer, can be inspected as low-resolution heatmaps, giving an indication of what triggers the different channels. However, with many channels (512 in the case of VGG16), it can be hard to get any overall information from this. Plotting the learned filter kernels is also an option, but is not very interpretable beyond the first layers. The distribution of gradients during back-propagation can also give some clues about the health of the training process, for instance to check for exploding gradients or dead units.

Encoder-decoder networks with their per-pixel outputs are inherently more explainable than bottleneck networks. The softmax values or final prediction can be inspected as an image directly, and can be used to inspect whether the network performs worse in specific regions of the input image.

4.3.2 Visualisation methods

There is clearly a need for methods that can extract information about the inner workings of the model and display it in a way that is more interpretable for humans. One strategy is to explain the network’s decision by relating the attention of the network back to the input image in form of a heatmap for visualisation. Many different approaches have been proposed for generating such heatmaps, which all have different properties. A summary of the main approaches with example visualisations is shown in Table 4.1, and the methods are briefly described below.

Table 4.1: Comparison of visualisation methods for explanation of deep neural networks. Example visualisations are retrieved from the original papers or the LRP online demo.² Best viewed in colour.

Method category	Example method	Example visualisation	Backward/forward pass	Properties
Perturbation of input	Occlusion maps [Simonyan et al., 2014]		Forward	Low resolution Slow Easy to interpret
Gradient-based	Saliency maps [Simonyan et al., 2014]		Backward	High resolution High-frequency Hard to interpret Conditioned on a label
Backpropagation-based	LRP [Bach et al., 2015]		Backward	High resolution High-frequency Focused Conditioned on a label
Class activation maps	CAM [Bolei Zhou et al., 2016]		Forward	Mid-resolution Focused Special architecture
Class activation maps	Grad-CAM [Selvaraju et al., 2020]		Backward	Mid-resolution Focused Conditioned on a label
Principal feature visualisation (Ours)	PFV [Bakken et al., 2020], Paper IV		Forward	Mid-resolution Contrasting features Not conditioned on a label

One strategy for relating a classifier's decision to specific regions in the image is to perform simple perturbations (e.g. occlusion) to the input [Zeiler and Fergus, 2014, Simonyan et al., 2014] and aggregate a heatmap per class based on the change in the output. Similarly, more advanced methods for perturbation of the input image has been proposed [Ribeiro et al., 2016b, Agarwal et al., 2019]. The drawback of these methods is that the number of required forward passes is proportional to the number of classes and resulting heatmap resolution.

Another category is gradient-based visualisation, which in its simplest form use the partial derivatives of the class score with respect to every input pixel. Then it visualises the absolute value as a *saliency map*, as proposed in [Simonyan et al., 2014]. This can be interpreted as a mapping of what pixels in the input image that need to change the least to change the class score the most. The visualisation is usually very high-frequent and noisy as seen in Table 4.1, and can therefore be hard to interpret.

Other visualisation methods use back-propagation to distribute the importance from the network output back to the input. Guided Backpropagation [Springenberg et al., 2015] conserve only positive gradients, and can therefore not show negative evidence for a class in the input image. Layer-wise Relevance Propagation [Bach et al., 2015] use a similar principle, but use a different rule for distributing the importance, and can display both positive and negative evidence in the input image. The visualisation can still be a bit noisy, but is more focused than gradient-based methods. In [Lapuschkin et al., 2019], LRP is used to reveal “Clever Hans” predictions and dataset bias in the commonly used Pascal VOC dataset.

The class activation mapping technique (CAM) was introduced in [Bolei Zhou et al., 2016]. It requires a special architecture that feed globally average pooled convolutional layers directly into softmax. In this way, more of the spatial information was kept as late in the neural network as possible, such that the activations can be visualised in the form of a heatmap. The CAM visualisations are well focused on objects and easy to interpret, but comes at the expense of lower accuracy, and can not be applied to existing networks without retraining.

Grad-CAM [Selvaraju et al., 2020] computes a set of neuron importance weights from the class scores to the activation map of the last convolutional layer, which is used to perform a weighted combination of forward activation maps. They show that the weights computed in the backpropagation is the same as the learned feature weights in CAM. Grad-CAM is therefore a generalisation of the CAM method and has similar properties but a more convenient implementation.

All the methods mentioned above explain the whole network including the decision head, and produce visualisations conditioned on one class label at a time. One disadvantage with this, is that running the visualisation on multiple classes, for instance both the correct class and the predicted one, requires multiple passes of the method. You also need access to labels to know what class to propagate from. An alternative approach is to visualise the features directly; Either to target multiple classes at the same time, which can be useful for debugging misclassification, or to reason about the properties of the features extractor alone, which can be useful if

4. Machine learning in practice

considering whether the features are transferable to another problem. This is one of the main motivations for the new visualisation method we present in Paper IV, called Principal Feature Visualisation (PFV). As seen in Table 4.1, our method can highlight both the dog and the cat in one pass. In Paper V, we see that this enable us to discover a bias in a dataset that would not have been detected with a visualisation conditioned on a class label. On the other hand, the PFV method can only explain features from the convolutional layers, and not the final decision. All-in-all, the different methods complement each other and can be used for different purposes.

4.3.3 *Evaluation of explanation methods*

What is a good explanation? Visually pleasing explanations are not necessarily the ones that contain the most information about the model. A good visualisation method needs to explain properties of the model or dataset and relate it to the input image for interpretation. There is a trade-off between interpretability and explanation quality, and it is important to remember that the most visually pleasing explanations not necessarily contain the most useful information.

Some evaluation procedures for visualisation methods have been proposed, for instance in [Samek et al., 2017]. [Adebayo et al., 2018] propose a methodology with series of randomisation tests, that can be used as simple sanity checks to verify sensitivity to the model or dataset. The first test is to successively randomise the weights of the model, and see if the visualisation changes. They find that many of the gradient-based methods, for instance Guided Backpropagation [Springenberg et al., 2015], rely too much on information in the input image, and are actually insensitive to changes in the model. Another test is to randomise the labels of the training data. If the visualisation does not respond to this perturbation, it cannot detect overfitting. The results are repeated in Figure 4.1, were we see that the gradient-based method in the first column shows a random response for the model trained on random labels, which is the correct behaviour. Several other methods, for instance Guided Backprop produce a seemingly reasonable explanation for a model trained on unreasonable data. This is probably because the neural network has developed some kind of edge detection features despite the random labels.

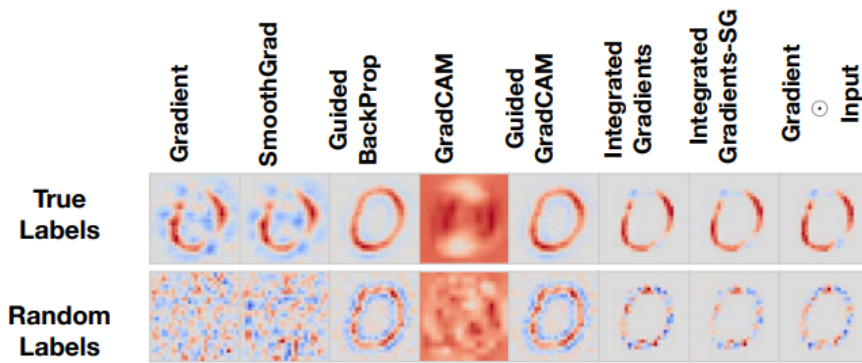


Figure 4.1: From [Adebayo et al., 2018]: Randomisation test for explanations of a model trained on the CIFAR digit dataset. The visualisations highlight the pixels contributing to the "0" label. The grid shows the visualisations for different explanation methods of a model trained on correct labels (top row) vs. a model trained on random labels (bottom row). Note that some methods, for instance Guided BackProp and Guided GradCAM produce an apparently reasonable response also for the model trained on random labels. Best viewed in colour.

CHAPTER 5

SUMMARY OF PAPERS

Learning-based methods have a great potential to enable low-cost vision-based guidance for mobile robots, but also come with challenges in terms of transparency and robustness. The work in this thesis seeks to answer the following research questions, as introduced in Chapter 1: 1) How can modern learning-based methods best be *applied* to crop-row following with agri-robots? 2) How can such methods be made more *explainable*? 3) How can such methods be made more *data-efficient*? The first two papers were related to question 1) and 3), and focused on investigating the properties of an established data-efficient learning strategy for application to crop row following. The lack of explainability of this approach led to a different strategy in Paper III, where a more explainable network architecture with a new supervision strategy for this task is proposed. Paper IV focuses on explainability for learning-based computer vision in general, and a new method for feature visualisation of neural networks is proposed. Finally, Paper V applies the visualisation method from Paper IV to the agri-robot application and proposes a new network architecture for row following that is both data-efficient and more explainable.

5.1 Paper I: End-to-end Learning for Autonomous Navigation for Agricultural Robots

In this paper, we proposed using an end-to-end learning strategy to predict steering angles for autonomous crop row following using only RGB image input, and presented the preliminary results for this method implemented onboard an agri-robot operating in a strawberry polytunnel.

The motivation behind this paper was to apply recent work in deep learning to learn a general approach for row following directly from visual input. The chosen end-to-end supervision strategy – i.e. learning steering commands directly from input images – is very data-efficient as the labels are generated as a part of the data collection process. The disadvantages are that there is no other output than the final steering command, which makes it hard to debug, and it also requires a large amount of training data. We applied an approach that was demonstrated on forest trails to an agricultural setting and trained a standard bottleneck CNN to predict steering angles based on a dataset with images labelled left/right/straight.

Initial tests in strawberry polytunnels showed promising results for networks that were pre-trained on trail data and fine-tuned on polytunnel data. This indicated that this could be a good approach, and the work was continued in Paper II. How-

5. Summary of papers

ever, experiments with existing visualisation methods gave ambiguous results, which motivated the development of the new visualisation technique in Paper IV.

5.2 Paper II: End-to-end Learning for Autonomous Crop Row-following

In this paper, we extended the work from Paper I, by implementing continuous angle prediction and using a larger polytunnel dataset collected with our robot setup. The motivation behind this paper was to further explore the role of pre-training and improve the precision of the steering commands. We hypothesised that pre-training on the established trails dataset would reduce the amount of application-specific training data needed.

The two main findings in this paper were that 1) A bit surprisingly, pre-training reduced the accuracy for the small dataset, and 2) Despite good results on the pure classification tasks, the results on angular precision were not good enough for robot guidance. This confirmed the disadvantage of training steering commands end-to-end; it is hard to tell whether this result is due to a lack of generalisation of the visual features, or if the issue lies in the decision head predicting the angle.

This motivated us to take a step back and train the visual features separately, which led to the work on crop row segmentation in [Ponnambalam et al., 2020] and Paper III, before returning to the end-to-end approach in Paper V.

5.3 Paper III: Robot-supervised Learning of Crop Row Segmentation

In this paper, we propose a self-supervised learning strategy for vision-based row following. We call it *robot supervised*, as we are using a supervision robot fully equipped with RTK-GNSS to automatically generate segmentation masks to train a neural network.

The main finding in this work was that the accuracy of the predicted segmentation masks was higher than the automatically generated labels the neural network was trained on. This meant that the network has learnt to ignore the noise introduced by the automatic supervision approach. The result was successful crop row detection through a more data-efficient approach than conventional segmentation labels, and a network that is easier to interpret and debug than the end-to-end approach in Paper II. However, it does require an additional processing step to estimate the row following commands, which is addressed with a hybrid approach in Paper V.

5.4 Paper IV: Principal Feature Visualisation in Convolutional Neural Networks

In this paper, we proposed a new visualisation method for convolutional neural networks, called *Principal Feature Visualisation* (PFV). This method maps learned features of a CNN back to the original image space, where principal directions in feature space are represented as different colours in an RGB image. The motivation was to develop a method that is simple to set up and execute during the forward pass, produces a visualisation that is clearly localised in the input image, and shows features that are easy to interpret.

In contrast to other methods, our visualisation shows the learned features of a bottleneck network directly, independently of the decision head. We demonstrated our method on two explanation use-cases, where we successfully identified missing features in misclassified examples, and predict which classes will work after fine-tuning on a new dataset.

While this paper focused on the explanation of image recognition tasks from the computer vision domain, the method is directly applicable to the end-to-end row following network in Paper I and Paper II, which we will come back to in Paper V.

5.5 Paper V: Applied learning for row-following with agri-robots

In this paper, we suggest two techniques to improve the performance of learning-based methods in practical applications like row-following. First, we apply our visualisation method to make end-to-end learning from crop row following more transparent, and then we unify the supervision approaches from the previous papers to propose a new *hybrid* network architecture and supervision approach that learns segmentation in parallel to steering commands. Addressing the issues from Paper I and Paper II, the motivation was to apply deep learning in a way that is both more transparent and more robust to biases and artefacts in the training data. By adding an additional segmentation output to the end-to-end network, we enforce semantically meaningful features, which makes it easier to control the training process and get robust performance in practical applications with limited data.

This paper has three main contributions. First, we demonstrate the application of the Principal Feature Visualisation (PFV) method “in-the-wild” for the first time, leading to the discovery of a bias in a public dataset. Second, we propose a novel hybrid network architecture and supervision approach for row following that is both explainable, robust and data-efficient. We get good results on a strawberry field dataset, with an average heading error of around 1° . The network even performs well with extremely little data and avoids the pitfall of a biased training set. Finally, we validate our approach through open-loop trials with an agri-robot and demonstrate

5. Summary of papers

good performance in a realistic setting. We believe our solution can perform row-following and enable autonomous driving.

5.6 Relevant papers not included in the dissertation

The candidate has co-authored two relevant publications that have not been included in the thesis: In [Ponnambalam et al., 2020], the focus was on robot control based on a CNN-based crop row segmentation, with the same platform and setting as Paper III, but with manually annotated data. In [Dyrstad et al., 2019] we presented an approach for learning robot bin-picking in simulation, which was successfully tested by picking shiny parts from a bin with a real robot and 3D sensor. This is another direction of data-efficient learning, which is more transferable for 3D data than the RGB images used in the works of this thesis.

5.7 Scientific contribution

The scientific contributions of this thesis can be summarised in the three categories related to the research questions:

Increased explainability has been achieved with the development of the novel feature visualisation method in Paper IV, which provides explanations that are complementary to existing methods, and has been successfully demonstrated both on general image recognition tasks and more specifically for crop row following in agriculture. Additionally, the hybrid approach for crop row following presented in Paper V gave a network architecture that is inherently more explainable than the end-to-end approach, and also demonstrated increased robustness.

Increased data-efficiency for learning-based crop row following has been achieved with the new supervision approach presented in Paper III, which eliminated the need for hand-drawn labels in segmentation-based crop row following. Additionally, the end-to-end learning approach was extended to provide labels with continuous angles, which gives more variation in the training data. The ability to efficiently label data with both segmentation masks and heading angles was an important enabler for the hybrid learning approach in Paper V.

New insight in applications of learning-based methods in the field. This has been achieved by considering the whole pipeline from field data collection to the final prediction result. Supervision strategies have been tested on data from real robots in the field, and successively improved in terms of data efficiency, robustness and explainability in the first three papers, before arriving at the ultimate approach in Paper V that is practical in terms of data collection, easier to debug, and more robust than the original end-to-end approach.

BIBLIOGRAPHY

- [Adadi, 2021] Adadi, A. (2021). A survey on data-efficient algorithms in big data era. *Journal of Big Data* 2021 8:1, 8(1):1–54.
- [Adebayo et al., 2018] Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 9505–9515. Neural information processing systems foundation.
- [Agarwal et al., 2019] Agarwal, C., Schonfeld, D., and Nguyen, A. (2019). Removing input features via a generative model to explain their attributions to an image classifier’s decisions.
- [Ahmadi et al., 2019] Ahmadi, A., Nardi, L., Chebrolu, N., and Stachniss, C. (2019). Visual servoing-based navigation for monitoring row-crop fields. *arXiv*, pages 4920–4926.
- [Andrew Ng et al., 2000] Andrew Ng, Tengyu Ma, Anand Avati, and Kian Katanforoosh (2000). Lecture notes, University of Stanford, CS229 Machine learning. *CS229 Lecture notes*.
- [Åstrand and Baerveldt, 2005] Åstrand, B. and Baerveldt, A. J. (2005). A vision based row-following system for agricultural field machinery. *Mechatronics*, 15(2):251–269.
- [Bach et al., 2015] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):1–46.
- [Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. pages 1–14.
- [Bakken et al., 2020] Bakken, M., Kvam, J., Stepanov, A. A., and Berge, A. (2020). Principal Feature Visualisation in Convolutional Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12368 LNCS, pages 18–31. Springer Science and Business Media Deutschland GmbH.
- [Barfoot, 2017] Barfoot, T. D. (2017). *State estimation for robotics*. Cambridge University Press.

Bibliography

- [Beyer et al., 2019] Beyer, L., Zhai, X., Oliver, A., and Kolesnikov, A. (2019). S4L: Self-supervised semi-supervised learning. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October.
- [Biber et al., 2012] Biber, P., Weiss, U., Dorna, M., and Albert, A. (2012). Navigation system of the autonomous agricultural robot bonirob. In *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal.
- [Blas and Blanke, 2011] Blas, M. R. and Blanke, M. (2011). Stereo vision with texture learning for fault-tolerant automatic baling. *Computers and electronics in agriculture*, 75(1):159–168.
- [Bolei Zhou et al., 2016] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba (2016). Learning Deep Features for Discriminative Localization. *CVPR*, 2016(1):M1–M6.
- [Burud et al., 2017] Burud, I., Lange, G., Lillemo, M., Bleken, E., Grimstad, L., and Johan From, P. (2017). Exploring Robots and UAVs as Phenotyping Tools in Plant Breeding. *IFAC-PapersOnLine*, 50(1):11479–11484.
- [Chen et al., 2017] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv*.
- [Doersch et al., 2015] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430.
- [Dumoulin et al., 2017] Dumoulin, V., Visin, F., Gualtieri, M., Saenko, K., Platt, R., Science, I., Zhu, W., Miao, J., Qing, L., Huang, G. B., García-Santillán, I. D., Montalvo, M., Guerrero, J. M., Pajares, G., English, A., Ross, P., Ball, D., Upcroft, B., Corke, P., Smolyanskiy, N., Kamenev, A., Smith, J., Birchfield, S., Ren, S., He, K., Girshick, R., Sun, J., Mai, X., Zhang, H., Meng, M. Q., Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2017). Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 156(6):61–79.
- [Dyrstad et al., 2019] Dyrstad, J., Bakken, M., Grøtli, E., Schulerud, H., and Mathiassen, J. (2019). Bin Picking of Reflective Steel Parts Using a Dual-Resolution Convolutional Neural Network Trained in a Simulated Environment. In *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*.
- [García-Santillán et al., 2018a] García-Santillán, I., Miguel Guerrero, J., Montalvo, M., Pajares, G., and Pajares pajares, G. (2018a). Curved and straight crop row detection by accumulation of green pixels from images in maize fields. *Precision Agriculture*, 19:18–41.

- [García-Santillán et al., 2018b] García-Santillán, I., Miguel Guerrero, J., Montalvo, M., Pajares, G., and Pajares pajares, G. (2018b). Curved and straight crop row detection by accumulation of green pixels from images in maize fields. *Precision Agriculture*, 19:18–41.
- [Giusti et al., 2015] Giusti, A., Guzzi, J., Cire, D. C., He, F.-l., Rodríguez, J. P., Fontana, F., Fässler, M., Forster, C., Schmidhuber, J., Caro, G. D., Scaramuzza, D., and Gambardella, L. M. (2015). A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters*, PP(99):1–1.
- [Gonzalez and Woods, 2007] Gonzalez, R. C. and Woods, R. E. (2007). *Digital Image Processing (3rd Edition)*.
- [Goodfellow lan, 2016] Goodfellow lan, Bengio Yoshua, C. A. (2016). *Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books*.
- [Grimstad and From, 2017] Grimstad, L. and From, P. J. (2017). Thorvald II - a Modular and Re-configurable Agricultural Robot. *IFAC-PapersOnLine*, 50(1):4588–4593.
- [Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition.
- [Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*, volume Springer.
- [He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385v1*, 7(3):171–180.
- [Hinton et al.,] Hinton, G., Srivastava, N., and Swersky, K. Neural Networks for Machine Learning Lecture 6a Overview of mini-batch gradient descent. Technical report.
- [Hough, 1962] Hough, P. V. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.
- [Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. Technical report.
- [Karpathy, 2017] Karpathy, A. (2017). Course notes from Stanford University CS231n: Convolutional neural networks for visual recognition.
- [Kingma and Ba, 2015] Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Bibliography

- [Kuhn, 2016] Kuhn, T. (2016). The Controlled Natural Language of Randall Munroe’s Thing Explainer. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9767:102–110.
- [Lapuschkin et al., 2019] Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K. R. (2019). Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1–8.
- [Larsson et al., 2019] Larsson, M., Stenborg, E., Toft, C., Hammarstrand, L., Sattler, T., and Kahl, F. (2019). Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 31–41.
- [Le et al., 2019] Le, T., Glenn, J., Gjevestad, O., and From, J. (2019). Online 3D Mapping and Localization System for Agricultural Robots. (2015):2015–2020.
- [Lee et al., 2013] Lee, D.-H. et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3.
- [Lin et al., 2016] Lin, G., Milan, A., Shen, C., and Reid, I. (2016). RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January*:5168–5177.
- [Lin and Chen, 2019] Lin, Y.-k. and Chen, S.-f. (2019). Development of Navigation System for Tea Field Machine Using Semantic Segmentation. (2018).
- [Liu and Frank,] Liu, R. and Frank, E. An intriguing failing of convolutional neural networks and the CoordConv solution. pages 1–24.
- [Loquercio et al., 2018] Loquercio, A., Maqueda, A. I., Carlos, R., and Scaramuzza, D. (2018). DroNet : Learning to Fly by Driving.
- [Ma et al., 2019] Ma, X., Deng, X., Qi, L., Jiang, Y., Li, H., Wang, Y., and Xing, X. (2019). Fully convolutional network for rice seedling and weed image segmentation at the seedling stage in paddy fields. *PloS one*, 14(4):e0215676.
- [Marchant and Brivot, 1995] Marchant, J. A. and Brivot, R. (1995). Real-Time Tracking of Plant Rows Using a Hough Transform. *Real-Time Imaging*, 1(5):363–371.
- [Milioto et al., 2018] Milioto, A., Lottes, P., and Stachniss, C. (2018). Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2229–2235.
- [Munroe, 2015] Munroe, R. (2015). *Thing Explainer — Complicated Stuff in Simple Words*. Houghton Mifflin Harcourt.

- [Otsu, 1979] Otsu, N. (1979). THRESHOLD SELECTION METHOD FROM GRAY-LEVEL HISTOGRAMS. *IEEE Trans Syst Man Cybern*, SMC-9(1):62–66.
- [Ponnambalam et al., 2020] Ponnambalam, V. R., Bakken, M., Moore, R. J., Gjevestad, J. G. O., and From, P. J. (2020). Autonomous crop row guidance using adaptive multi-roi in strawberry fields. *Sensors (Switzerland)*, 20(18):1–17.
- [Potena et al., 2016] Potena, C., Nardi, D., and Pretto, A. (2016). Fast and accurate crop and weed identification with summarized train sets for precision agriculture. In *International Conference on Intelligent Autonomous Systems*, pages 105–121. Springer.
- [Reina and Milella, 2012] Reina, G. and Milella, A. (2012). Towards autonomous agriculture: Automatic ground detection using trinocular stereovision. *Sensors*, 12(9):12405–12423.
- [Ribeiro et al., 2016a] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016a). "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016.
- [Ribeiro et al., 2016b] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016b). "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August, pages 1135–1144.
- [Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In: Rumelhart D E, McClelland J L et al. (eds.) *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1(V).
- [Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.
- [Sadeghi and Levine, 2016] Sadeghi, F. and Levine, S. (2016). CAD2RL: Real Single-Image Flight without a Single Real Image.
- [Samek et al., 2017] Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K. R. (2017). Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673.
- [Scaramuzza et al., 2006] Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006). A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701.

Bibliography

- [Selvaraju et al., 2020] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2):336–359.
- [Simonyan et al., 2014] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings*, pages 1–8.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1.
- [Springenberg et al., 2015] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*.
- [Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Technical report.
- [Stefas et al., 2016] Stefas, N., Bayram, H., and Isler, V. (2016). Vision-Based UAV Navigation in Orchards. *IFAC-PapersOnLine*, 49(16):10–15.
- [Wendel and Underwood, 2016] Wendel, A. and Underwood, J. (2016). Self-supervised weed detection in vegetable crops using ground based hyperspectral imaging. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:5128–5135.
- [Woebbecke et al., 1995] Woebbecke, D. M., Meyer, G. E., Bargaen, K. V., and Mortensen, D. A. (1995). Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the American Society of Agricultural Engineers*, 38(1):259–269.
- [Zeiler, 2012] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method.
- [Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS(PART 1):818–833.
- [Zeng et al., 2017] Zeng, A., Yu, K.-T., Song, S., Suo, D., Walker, E., Rodriguez, A., and Xiao, J. (2017). Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1386–1383. IEEE.

- [Zhang et al., 2016] Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9907 LNCS.
- [Zhang et al., 2018] Zhang, X., Li, X., Zhang, B., Zhou, J., Tian, G., Xiong, Y., and Gu, B. (2018). Automated robust crop-row detection in maize fields based on position clustering algorithm and shortest path method. *Computers and Electronics in Agriculture*, 154:165–175.
- [Zhou et al., 2012] Zhou, S., Xi, J., McDaniel, M. W., Nishihata, T., Salesses, P., and Iagnemma, K. (2012). Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain. *Journal of Field Robotics*, 29(2):277–297.

Papers

PAPER I

END-TO-END LEARNING FOR AUTONOMOUS NAVIGATION FOR AGRICULTURAL ROBOTS

Marianne Bakken, Richard J. D. Moore, Pål From

Workshop on Robotic Vision and Action in Agriculture at ICRA 2018. <https://research.qut.edu.au/future-farming/projects/icra-2018-workshop-on-robotic-vision-and-action-in-agriculture/>

End-to-end Learning for Autonomous Navigation for Agricultural Robots

Marianne Bakken^{1,2,*}, Richard Moore¹ and Pål From²

Abstract—For robotic technology to be adopted within the agricultural domain, there is a need for low-cost systems that can be deployed autonomously across a wide variety of crop types, environmental conditions, and planting methods, without extensive re-engineering. We present an end-to-end learning approach for row following in agriculture, that can be used for navigation on lightweight robotic platforms. Building on recent work on deep convolutional neural networks (DCNNs) and end-to-end learning approaches, we propose to train our DCNN to output control commands directly from RGB image input data, using a large-scale forest trail dataset and then fine-tune on small datasets from agricultural settings. For this purpose, we recorded data for row-following from a strawberry polytunnel and a sugar cane field. Preliminary evaluation on independent test datasets show promising results on a domain not seen during training. This indicates that our approach generalises well across agricultural domains, and that the low-level features obtained from the trail dataset are relevant for agricultural applications. Future work includes data capture from different applications and seasons to train and test on more data, and verify the control approach on a real robot or drone.

I. INTRODUCTION

Automating agricultural practices through the use of robots (i.e. agri-robots) is a key strategy for raising farm productivity and achieving sustainable food production for future generations. To maximise efficiency and to avoid damaging crops, agri-robots must be able to navigate precisely and reliably through crop plantations. However, modern food production techniques have resulted in diverse growing environments – from greenhouses and polytunnels to open fields – presenting a significant technological challenge for the development of generally useful agri-robots.

External localisation systems such as DGPS/RTK-GPS [1] can provide precise position information for robots, but such systems are expensive and require a network of base stations to provide real-time correction data as well as a precise map of crop locations. Visual-inertial navigation (V-INS) or VI-SLAM systems do not rely on external hardware and enable impressively accurate state estimation for lightweight autonomous vehicles [2], but they also require a precise map of crop locations and suffer from coordinate frame drift in agricultural settings due to the typically long non-overlapping trajectories and self-similar environments.

There is therefore great interest in developing local navigation solutions that enable the robot to extract some degree of understanding from the current scene in order to make *intelligent* guidance decisions. Such approaches could be

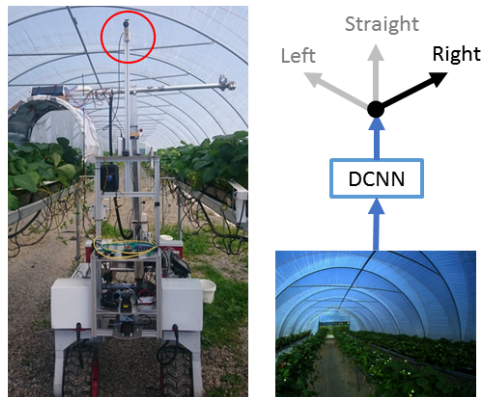


Fig. 1: The mobile robot recording setup used to capture the strawberry polytunnel dataset. A forward-facing, wide-angle video camera (red circle) mounted above the robot was used to capture images for offline training and testing. A DCNN was trained to predict view orientation for autonomous row following in a diverse set of agricultural scenes.

low-cost as they do not rely on external infrastructure or precise maps, and the same technology could be flexibly applied to diverse environments or other robotic platforms. Our proposed method should be applicable to ground based robots as well as aerial robots.

Existing local sensing approaches typically aim to segment the scene into vegetation and non-vegetation classes based on either 2D image data from RGB [3], [4] or NIR [5] cameras, or 3D data from stereo systems [6] or scanning LIDAR [7]. The vehicle’s lateral offset from the preferred trajectory is then computed by leveraging the typically linear layout of crop plantations [6], [3], [4], [8]. However, 3D methods do not perform well when crops are too sparse or too dense, and 2D methods traditionally employ hand-crafted features, or features that are specific for a particular environment, and thus do not generalise well to other agricultural settings.

Here we build on recent work with deep convolutional neural networks (DCNNs) to train optimised image features for classification. In order to train environment-independent features, the training dataset should comprise images from very many different agricultural settings. To the best of our knowledge, no such openly available general agricultural dataset exists and collating and annotating such a dataset would be costly, so instead we propose to leverage open trail-

*Corresponding author, marianne.bakken@sintef.no

¹SINTEF Digital, Oslo, Norway

²Norwegian University of Life Sciences (NMBU), Ås, Norway

following datasets that contain mixed vegetation and trail scenes [9]. Our hypothesis is that a network trained to detect features present in the trail-following dataset could easily be adapted for crop row following in agricultural settings and should perform well over a wider variety of environments than networks trained on a small number of agricultural scenes. Furthermore, we propose to use an *end-to-end* learning strategy, i.e. using one single neural network to predict high-level control policies directly from sensor data. This follows on from the work of [9] and [10], who showed that end-to-end learning can be successfully employed to overcome the problem of designing control policies for aerial platforms in scenes with widely varying appearances. We will train our DCNN to output control commands directly from RGB image input data. This should enable reliable crop row following over a wider variety of agricultural settings and with more flexibility than traditional local navigation approaches.

Specifically, we propose to 1) use existing large-scale datasets for trail following to train DCNNs for crop row following; 2) use fine-tuning on smaller domain-specific datasets to adapt the trained networks for use in agricultural environments; and 3) investigate how the networks should be trained to generalise as well as possible to diverse agricultural settings without additional data capture. We present a preliminary evaluation of our approach in two different agricultural domains: strawberry polytunnels and sugar cane fields. We also investigate visualisation techniques to analyse the features and decision structures learned by the networks.

II. RELATED WORK

In this section, we will focus on literature regarding deep neural networks for robotic control based on RGB cameras. This literature is often focused on aerial robots, but is applicable to other mobile robot platforms as well.

Recently, there has been an increasing amount of work on learning control policies directly from RGB images using deep neural networks. Reinforcement learning in particular has seen great success in game settings, but require an extremely large number of training examples, which are usually not possible to collect in real environments. Transfer of networks trained purely on simulated data to real-world scenarios has been successfully demonstrated by [11], showing image-based autonomous indoor flight with a drone without using any real images for training. AirSim [12] provides a simulation environment specifically targeted for outdoor aerial platforms, but there are very few simulation environments available. With such diversity in environments and tasks, building a full simulation environment for all agricultural applications is infeasible.

Supervised learning approaches require fewer samples compared to reinforcement learning, but the samples must be labelled. The success of DCNNs in object recognition and detection is due mainly to the massive amount of manually labelled data in datasets like ImageNet [13] and Pascal VOC [14]. Networks pre-trained on such datasets can often be used directly as general feature extractors for domains that

are represented in the dataset (e.g. pedestrian detection for Pascal VOC), but training DCNNs for other domains, such as agriculture, requires large amounts of new data to be collected and annotated, which is both time-consuming and expensive.

Recent work has shown promising results in supervised end-to-end learning of high-level control for aerial robots [10]. However, for autonomous control of aerial robots it is often not practical or possible to acquire large scale labelled datasets in flight, thus [10] use a car driving dataset from Udacity¹ to train a drone to follow roadways. This approach gives a continuous output and can be taught a wide range of control policies, but since it needs ground truth steering commands, it requires an expert driver for data capture. A simpler data collection approach was employed by [9], who collected an extensive dataset for prediction of view orientation on forest trails with a head-mounted three-camera rig, which gave a built-in labelling of orientation (left/straight/right). Based on this, they trained a view orientation classifier, which was used to compute yaw control of a drone from RGB images only. [15] developed this approach further by experimenting with different network architectures and adding lateral control, which showed improved performance and indicated good generalisation capabilities within the trail domain. This approach allows ground truth commands to be easily generated during data capture, but is limited somewhat by the type of steering commands that can be learned.

We expect that the IDSIA trail dataset² from [9] better captures typical agricultural features like vegetation and soil than for instance the Udacity city driving dataset. Thus, we select this dataset for pre-training our DCNN.

III. METHODOLOGY

Our approach uses a deep convolutional neural network (DCNN) to learn steering angles from images labelled with different viewpoints. We pre-train our DCNN using the large-scale IDSIA trail dataset before fine-tuning the top (classification) layers with our own smaller dataset recorded in a strawberry polytunnel. We investigate several regularisation techniques to improve the generalisation capability of the trained network as much as possible prior to fine-tuning.

A. Network architecture

Our approach is based on the principles from the trail following method in [9]. As a starting point, we use the VGG16 [16] network architecture, a popular and well-tested architecture that was also used in [15] for trail following. The last fully-connected layer was modified to work with three output classes and the input size was changed to 150x150. An overview of the network architecture is shown in Fig. 2.

B. Datasets

1) *Trail dataset*: The IDSIA Swiss Alps trail dataset from [9] consists of several kilometres of trail recordings. The images are recorded with a rig of three cameras looking left, straight and right, that makes up the ground truth labels for the

¹Available at <https://github.com/udacity/self-driving-car>

²Available at <http://people.idsia.ch/~giusti1/forest/web/>

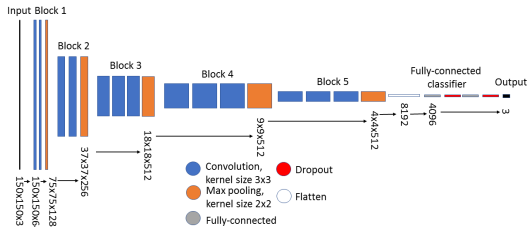


Fig. 2: Network architecture for VGG16 with dropout on fully-connected layers and 3 output classes. Dropout layers was added to enable experiments with regularisation, but not used in the "original" version.

three viewpoint classes. The dataset contains different kinds of trails, and also has some road sections. Most recordings are from the same season, probably late autumn, and have very little green vegetation. See Fig. 3 for example images. As in [15], we used the folders 003, 008 and 010 for testing, and the remainder for training.

2) *Agricultural datasets*: We recorded two new datasets for row following in agriculture, with a recording approach similar to that described for the IDSIA dataset. The first dataset comprises a strawberry polytunnel, and was captured using a mobile robot equipped with a Basler Ace camera and Sunex 190 degree fisheye lens mounted on a pole to mimic a drone’s perspective (Fig. 1). Video sequences were recorded while driving straight along the strawberry rows in the tunnel, by means of an autonomous control system not part of this work. By using a wide-field-of-view camera instead of a fixed rig of three cameras, we could extract virtual camera views from any angle after the recordings were made. To make it compatible with the IDSIA dataset and the 3-class viewpoint classification, virtual camera views were extracted with a field of view of 140 degrees and offsets of $-27/0/27$ degrees (similar to the GoPro rig described in [9]). Example images from the dataset are shown in Fig. 3. The images in the strawberry polytunnel dataset are quite self-similar, despite the robot driving along two separate rows from both directions. To make the test data as different as possible, the recordings from the second row was reserved for the test dataset.

The second agricultural dataset was recorded with a mobile phone in a sugar cane field in Brazil. The videos were recorded by walking straight along a row and holding the camera in one of three different directions. Example images are shown in Fig. 3.

C. Training procedures

1) *Framework and setup*: Our network was first trained on the IDSIA dataset as described in [15] using the Keras framework [17] with a Tensorflow backend. Hyperparameters not stated in [15] were determined empirically. To avoid many similar training images, every second image was sampled, resulting in 7571 different images in the training set, prior to augmentation. The folders in the IDSIA training set were split

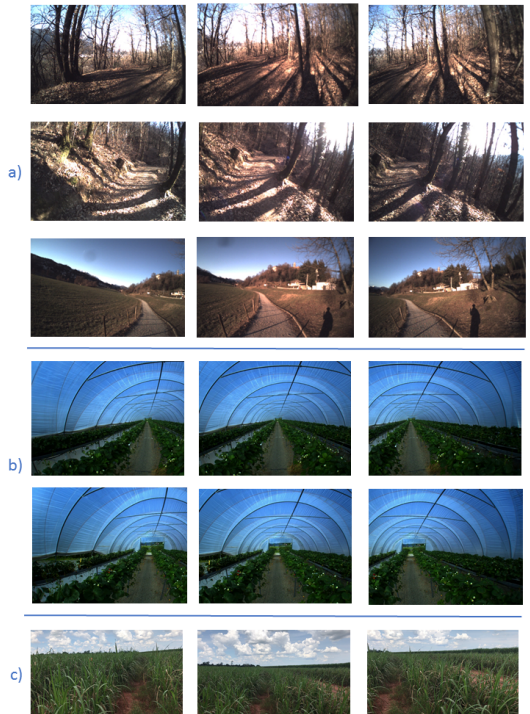


Fig. 3: Example left, straight and right class images from the different datasets: a) IDSIA (forest trails), b) Sylling (strawberry polytunnels), c) Brazil (sugarcane fields). The two agricultural datasets are very different; the strawberry tunnels are very structured and self-similar, while the rows in the sugar cane field are almost hidden in the tall grass.

2/7 for validation/training. We used built-in data augmentation in Keras: random shear (0.2), zoom (0.2), horizontal flips, rotation (10 degrees) and height shift (0.2). Transformations that would change the class of the image, such as random crops was omitted, and for the horizontal flip the left/right labels was swapped. The network was trained with the Adam optimiser [18] with categorical cross-entropy loss, a learning rate of $7 \cdot 10^{-5}$ and a batch size of 32 for 50 epochs. This takes 3.5 hours on a TitanX GPU. Model checkpoints were saved every 5th epoch, and the model with the best validation accuracy was picked for further testing.

2) *Regularisation*: We experimented with three common regularisation techniques during training: Early stopping, weight decay, and dropout. Early stopping was implemented by picking the model with the lowest loss after 50 training epochs. The L^2 parameter norm penalty, also known as weight decay, was implemented through the kernel regulariser for each layer in Keras. Dropout was implemented via the Keras dropout layer. We experimented with different values for learning rate and the hyperparameters of weight decay and dropout during training. The most promising combination was

weight decay with regularisation strength 0.001 and dropout with probability 0.5 on the fully-connected layers only, trained with a learning rate of $1 \cdot 10^{-5}$.

3) *Fine-tuning*: fine-tuning was done by freezing all the layers except the fully-connected layers on a model pre-trained on the IDSIA dataset, and then running the training procedure on the Sylling training set with a very low learning rate ($7 \cdot 10^{-8}$). The training was run for 50 epochs, and model checkpoints were saved such that we could pick the final model from any time in the training process. Since the dataset is small and we only train the top layers, this training procedure is quite fast (50 epochs takes 22 minutes on a TitanX GPU).

D. Visualisation of deep neural nets

The Picasso visualisation framework [19] was used to obtain saliency maps for selected example images. Activation maps are extracted directly from the layers in Keras.³

IV. EXPERIMENTAL RESULTS

In this section, we show preliminary quantitative results for the view orientation classification for different datasets. As the amount of agriculture-specific data is very limited, we first investigate the performance and generalisation capabilities of different regularisation strategies using a network trained on the large-scale IDSIA trail dataset. We then compare our network’s performance in agricultural settings before and after fine-tuning with domain-specific data.

A. Regularisation

Training experiments with different regularisation approaches were performed with the IDSIA training set as described in III-C.2. The two most promising models from the training experiments was then evaluated on the IDSIA, Sylling, and Brazil test datasets to investigate their generalisation capabilities.

The results in the upper part of Table I show that the original network implementation with no explicit fine-tuning scores well on the IDSIA test set (about the same accuracy as the one reported in [15]), but performs worse than random guessing on the agriculture datasets. With any of the two regularisation strategies, performance is reduced somewhat on the IDSIA dataset but shows improved accuracy for the agriculture datasets, although the error rates are still unacceptably high. The confusion matrices in Fig. 4a–c reveal that the classifier fails to recognise the straight class in the two agriculture datasets. The IDSIA classification is biased equally towards left and right, while the Brazil classification is biased towards the right class.

As further discussed in Section V, visualisation indicates that the network with weight decay and dropout has better generalisation properties, and this was therefore chosen as a basis for further experiments with fine-tuning.

³Visualised with a modified version of <https://github.com/philipperemy/keras-visualize-activations>

TABLE I: Accuracy on different datasets. Upper part: Trained on IDSIA dataset only, different regularisation strategies. Lower part: Pre-trained on IDSIA, fine-tuned on Sylling.

	Test accuracy [%]		
	IDSIA	Sylling	Brazil
Original (no explicit regularisation)	89.1	29.6	28.1
Early stopping	87.9	77.0	55.5
Weight decay and dropout on top layers	88.9	65.7	64.0
Fine-tuned on Sylling data	85.9	96.9	98.4

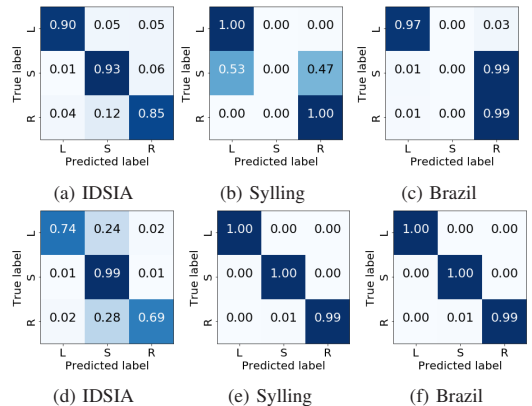


Fig. 4: Confusion matrices for prediction on different datasets before (a–c) and after (d–f) fine-tuning, for the network with weight decay and dropout on top layers. This shows that the Sylling and Brazil classification fails on the straight class before fine-tuning.

B. Fine-tuning

The accuracy on the test datasets before and after fine-tuning is shown in the lower part of Table I, and the confusion matrices are shown in Fig. 4d–f. As expected, we see a decrease in performance for the IDSIA data after fine-tuning with Sylling data, and increased performance for the Sylling data. The Sylling test dataset is however quite similar to the training dataset. It is therefore more interesting to look at the result for the Brazil dataset, which was not seen during training and comprises scenes very different to those of the Sylling dataset. We observed that the test accuracy of the Brazil dataset increased from 64% to 98.4% after fine-tuning on Sylling data – a significant performance improvement on an unseen domain.

V. DISCUSSION

Our approach shows promising classification results in our initial tests with diverse agricultural data. In this section we analyse our results using several visualisation techniques to try and better understand the benefits and limitations of this approach as well as the differences between regularisation techniques and the effect of fine-tuning.

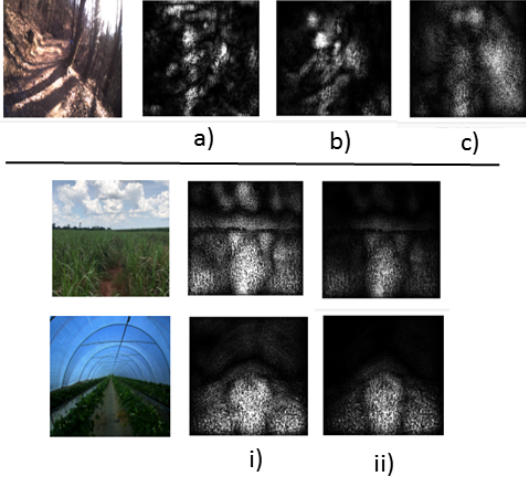


Fig. 5: Saliency maps for example images and different network versions. Upper part: regularisation a) original (no explicit regularisation); b) early stopping; c) weight decay and dropout on top layers. Lower part: i) before fine-tuning; ii) after fine-tuning. Saliency maps indicate which parts of an input image trigger a particular class the most – here we illustrate the response for an example of the straight ahead class.

A. Regularisation

In Table I, the original network (without explicit regularisation) showed a clear tendency for overfitting on the IDSIA training data, and while the two regularisation techniques showed improved performance on the diverse test datasets, it is difficult to judge generalisation ability based solely on accuracy metrics. This is highlighted by the saliency (Fig. 5) and activation (Fig. 6) maps for the three different networks. As seen in the upper part of Fig. 5, the saliency map for the original network shows a very specific and localised response, which is still present for the early stopping network. However, the network regularised with weight decay and dropout shows broader and smoother responses with distinguishable responses for trail and vegetation regions, indicating features that might more robustly be transferred to diverse environments. There is also a clear difference in the activation maps (Fig. 6): the regularised networks, especially that with weight decay and dropout, have more activated layers, meaning that a larger portion of the network is actually in use.

Both the saliency maps and the activation maps indicate that the early stopping network has features that are less developed and will be more prone to overfitting than the weight decay and dropout network. It was therefore concluded that the weight decay and dropout regularisation approach was a better starting point for fine-tuning the network for agricultural settings.

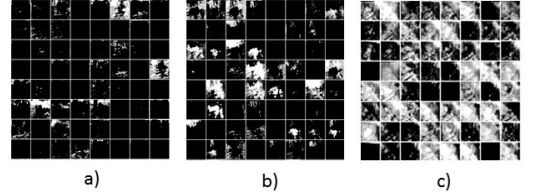


Fig. 6: Activations of 64 of the filters in Layer 2 of Block 3 in 2, during forward pass of one example trail image, for different regularisation approaches: a) Original (no explicit regularisation), b) Early stopping, c) weight decay and dropout on top layers. (Contrast has been enhanced to help visibility.)

With proper regularisation, the performance of our network on the Sylling and Brazil datasets improves to an accuracy of around 66% (Table I). However, it is clear from studying the confusion matrices (Fig. 4) that, although the left and right classes perform well, the straight ahead class performs very poorly. This is an interesting result and highlights the fact that each class is learned independently by the network – i.e. there is no logical relationship between the learned classes. The particular features learned by the network to represent the straight ahead class in the IDSIA trail dataset are not well represented in the agricultural datasets. This could be as a result of evident differences between the forest trail and agricultural domains – namely that in the agricultural datasets, the sky or tunnel ceiling is clearly visible in all cases and provides little information on the location of the crop row, whereas in the trail dataset the sky is often occluded and the upper image regions contain background vegetation. Features in the upper image region that may have been strong indicators for the straight ahead class in the trail dataset no longer provide useful input for the agricultural data and hence, the network performs poorly for this class. This explanation is supported by the saliency maps for the regularised network when tested with agricultural data (Fig. 5), which show significant activations in the sky region prior to fine-tuning.

Representing the guidance problem as a regression with continuous steering angle should permit a more logical relationship between outputs and avoid unexpected network behaviour like that described above. Our data collection setup, with wide-field-of-view camera and virtual viewpoints, supports this approach, but the data collected in the IDSIA trail dataset does not. We plan to collect additional datasets to investigate this approach.

B. Fine-tuning

The network accuracy on agricultural test sets following fine-tuning indicates that the spectrum of features learned from the trail data is broad enough to encompass those features present in the agricultural domain, verifying our initial hypothesis. This also indicates that the feature detection/classification layers of our DCNN could be efficiently

adapted to the agricultural domain with little new data, and generalise well to diverse agricultural settings.

The changes made to the classification layers during fine-tuning have altered the relative weights of the various features used to distinguish between orientation classes. As discussed above, there are some evident differences between the trail data and agricultural data, and fine-tuning has made the fully-connected layers more focused on detecting the rows in the lower part of the image, since there are no relevant features above the horizon. This observation applies also for many other agricultural applications, and it will be interesting to perform additional testing with different environmental conditions and settings to investigate this further.

VI. CONCLUSIONS

We have presented an approach for learning row following in agriculture, that can predict steering angles from RGB images only, enabling lightweight autonomous navigation for e.g. low-flying drones and lightweight agriculture robots. We trained a deep neural network based on a large-scale trail dataset, and fine-tuned on our own agriculture dataset, showing promising results for an agricultural domain not seen in training. Our preliminary results indicate that our approach has the advantage of enabling a proficient, pre-trained base network to be easily adapted to new domains with little application-specific data. The adaptation is done by fine-tuning a small part of the network, which gives a more rapid test cycle than full training of a deep network, and opens up for the possibility to adapt the system for new applications or seasons on-site in the field.

VII. FUTURE WORK

It should be noted that the agricultural datasets used in this paper are very small and can only give a preliminary indication of the performance of this approach. More data is needed for further training and testing of varying seasons and applications.

To date, we have only evaluated classification performance for our network, and not control policies. The control policy used here is very simple – discrete yaw commands – which results in jerky control patterns and can have difficulties recovering from lateral offsets. This can be improved by adding lateral control as in [15], which requires one more camera in the recording setup. Alternatively, one could learn steering commands directly as in [10], but this increases the complexity of the recording step.

Our network was trained on settings with a single row, and our agricultural test sets contained a single predominantly visible row. For many agricultural settings, multiple rows would be visible in each input image, which may confuse our network. In future work we will investigate how the presence of multiple crop rows impacts network performance and whether this problem could be overcome by augmenting the training datasets, or by utilising alternative network designs, e.g. recurrent networks that might enable the robot to “lock on” to the current row.

We also plan to perform tests on a real robot or drone in an agricultural setting to verify the control performance.

ACKNOWLEDGEMENT

This work was funded by The Norwegian Research Council, grant number 259869. The authors would like to thank Eirik Solberg and Simen Myhre for help with recording data from strawberry polytunnels, as well as Asbjørn Berge for fruitful discussions about visualisation of deep neural networks.

REFERENCES

- [1] M. Perez-Ruiz and S. K. Upadhyaya, “GNSS in precision agricultural operations,” in *New Approach of Indoor and Outdoor Localization Systems*, InTech, 2012.
- [2] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.
- [3] T. Bakker, H. Wouters, K. Van Asselt, J. Bontsema, L. Tang, J. Müller, and G. van Straten, “A vision based row detection system for sugar beet,” *Computers and electronics in agriculture*, vol. 60, no. 1, pp. 87–95, 2008.
- [4] G.-Q. Jiang, C.-J. Zhao, and Y.-S. Si, “A machine vision based crop rows detection for agricultural robots,” in *Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on*, pp. 114–118, IEEE, 2010.
- [5] G. Halmetschlagler, J. Prankl, and M. Vincze, “Probabilistic near infrared and depth based crop line identification,” in *Workshop Proceedings of IAS-13 Conference on*, pp. 474–482, 2014.
- [6] M. Kise, Q. Zhang, and F. R. Más, “A stereovision-based crop row detection method for tractor-automated guidance,” *Biosystems engineering*, vol. 90, no. 4, pp. 357–367, 2005.
- [7] P. Biber, U. Weiss, M. Dorna, and A. Albert, “Navigation system of the autonomous agricultural robot bonirob,” in *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal, 2012.
- [8] A. English, P. Ross, D. Ball, B. Upcroft, and P. Corke, “Learning crop models for vision-based guidance of agricultural robots,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 1158–1163, IEEE, 2015.
- [9] A. Giusti, J. Guzzi, D. C. Cireşan, H. Fang-Lin, J. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, and L. M. Gambardella, “A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots,” *Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.
- [10] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, “DroNet: Learning to Fly by Driving,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.
- [11] F. Sadeghi and S. Levine, “CAD2RL: Real Single-Image Flight without a Single Real Image,” 2016.
- [12] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” *Field and Service Robotics*, 2017.
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- [14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.
- [15] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, “Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness,” *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. pp. 4241–4247, 2017.
- [16] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *CoRR*, vol. abs/1409.1, 2014.
- [17] F. Chollet et al., “Keras,” <https://keras.io>, 2015.
- [18] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014.
- [19] R. Henderson and R. Rothe, “Picasso: A modular framework for visualizing the learning process of neural network image classifiers,” *Journal of Open Research Software*, vol. 5, no. 1, 2017.

PAPER II

END-TO-END LEARNING FOR AUTONOMOUS CROP ROW-FOLLOWING

Marianne Bakken, Richard J. D. Moore, Pål From

IFAC-PapersOnLine 52.30, Special Issue for 6th IFAC Conference on Sensing,
Control and Automation Technologies for Agriculture AGRICONTROL 2019.
<https://doi.org/10.1016/j.ifacol.2019.12.505>

II

End-to-end Learning for Autonomous Crop Row-following^{*}

Marianne Bakken^{*,**} Richard J. D. Moore^{*} Pål From^{**}

^{*} SINTEF Digital, Oslo, Norway (email: marianne.bakken@sintef.no)

^{**} Norwegian University of Life Sciences (NMBU), Ås, Norway

Abstract: For robotic technology to be adopted within the agricultural domain, there is a need for low-cost systems that can be flexibly deployed across a wide variety of crop types, environmental conditions, and planting methods, without extensive re-engineering. Here we present an approach for predicting steering angles for an autonomous, crop row-following, agri-robot using only RGB image input. Our approach employs a deep convolutional neural network (DCNN) and an end-to-end learning strategy. We pre-train our network using existing open datasets containing natural features and show that this approach can help to preserve performance across diverse agricultural settings. We also present preliminary results from open-loop field tests that demonstrate the feasibility and some of the limitations of this approach for agri-robot guidance.

Keywords: Robot vision; Robot navigation; Machine learning; Agricultural robotics; Mobile robots

1. INTRODUCTION

Automating agricultural practices through the use of robots (i.e. agri-robots, Fig. 1) is a key strategy for improving farm productivity and achieving sustainable food production to meet the needs of future generations. However, modern food production techniques have resulted in diverse growing environments – from greenhouses and polytunnels to open fields (Fig. 2) – presenting a significant technological challenge for the development of generally-useful agri-robots.

In order for autonomous agri-robots to be a realistic and cost-effective alternative for the end-user (i.e. farmers), they must overcome the following challenges:

- (1) Accurate navigation to maximise efficiency and avoid damaging crops.
- (2) Flexibility to support various environments, crop types, and environmental conditions.
- (3) Minimal setup and installation cost.
- (4) Safe and reliable operation, including intelligent response to unexpected conditions or events.

Accurate and flexible navigation can be achieved with external localisation systems such as D-/RTK-GNSS (e.g. Perez-Ruiz and Upadhyaya (2012)), but such systems require a network of base stations to provide real-time correction data as well as a precise map of crop locations and are therefore expensive to install. Visual-inertial navigation (V-INS) or visual- or lidar-based SLAM systems do not rely on external hardware and have been demonstrated on board agri-robots (e.g. Le et al. (2019)), but they also require a precise map of crop locations and can suffer from coordinate frame drift in agricultural settings due to the

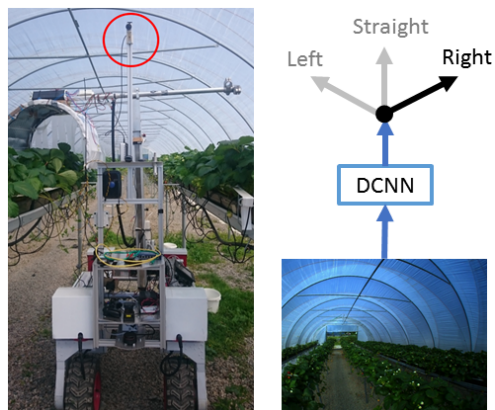


Fig. 1. The mobile robot recording setup: A forward-facing, wide-angle video camera (red circle) mounted above the robot was used to capture images for offline training as well as online field tests. A DCNN was trained to predict view orientation for autonomous row following in agricultural environments.

typically long non-overlapping trajectories and self-similar environments. To meet the above challenges, it is therefore interesting to investigate local (i.e. onboard) navigation solutions that provide direct guidance relative to crop locations and have no external hardware dependencies.

Existing local sensing approaches typically aim to segment the scene into vegetation and non-vegetation classes based on either 2D image data from RGB (Bakker et al. (2008); Jiang et al. (2010)) or NIR (Halmetschlager et al. (2014)) cameras, or 3D data from stereo systems (Kise et al.

^{*} This work was funded by The Norwegian Research Council, grant number 259869.

(2005)) or scanning LIDAR (Biber et al. (2012)). The vehicle’s heading or lateral offset from the preferred trajectory can then be computed by leveraging the typically linear layout of crop plantations (Kise et al. (2005); Bakker et al. (2008); Jiang et al. (2010); English et al. (2015)). However, 3D methods do not perform well when crops are too sparse or too dense, and 2D methods traditionally employ hand-crafted features, or features that are specific to a particular crop type and/or cultivation method, and thus do not generalise well to other agricultural settings.

Much recent work has shown that deep convolutional networks (DCNNs) are able to learn optimised image features for a wide range of classification and segmentation tasks even in poorly structured scenes, so it is therefore interesting to investigate whether DCNNs can enable more general solutions for crop row following than traditional approaches. However, DCNN-based approaches require feature appearance variation to be captured at training time, which means data capture and annotation is a time-consuming and costly process.

Our hypothesis is that by pre-training a DCNN on available data sets with non-specific vegetation features, e.g. forest trails (Giusti et al. (2015)), we can reduce the amount and diversity of agricultural-specific training data required and additionally improve the ability of the network to generalise to other agricultural use cases. Our preliminary work (Bakken et al. (2018)) has shown that a network trained on forest trail data can be relatively simply adapted for crop row following in various settings by fine-tuning with a small amount of agricultural-specific training data. Here we expand on this work to compare the performance of our pre-trained and fine-tuned network with a network trained fully on agricultural data to show that our approach retains better generalisation capability.

Additionally, since our ultimate aim is to establish whether this approach can be used successfully to provide guidance for an autonomous agri-robot in diverse settings, we implement our approach on board a test platform for field testing. We propose to use an *end-to-end* learning strategy to train our DCNN to output control commands for our autonomous vehicle directly from RGB image input data, following on from the work of Giusti et al. (2015) and Loquercio et al. (2018), who showed that end-to-end learning can be successfully employed to overcome the problem of designing control policies for autonomous platforms in scenes with widely varying appearances. Here we present preliminary results from initial open-loop field trials that demonstrate the feasibility and some of the limitations of our approach.

2. RELATED WORK

Recently, there has been an increasing amount of work on learning control policies directly from RGB images using deep neural networks. Reinforcement learning in particular has seen great success in game settings, but requires an extremely large number of training examples, which are usually not possible to collect in real environments. Transfer of networks trained purely on simulated data to real-world scenarios has been successfully demonstrated by Sadeghi and Levine (2016), but with such diversity



Fig. 2. Diverse agricultural scenes with crop rows (t-b, l-r): sugarcane, apple, strawberry, broccoli. Rows are often not easily identifiable and can change rapidly in appearance – presenting a difficult challenge for autonomous robots. We use the strawberry polytunnel case as a controlled environment for initial testing, but have designed our approach to generalise to other agricultural scenes.

in environments and tasks, building a full simulation environment for all agricultural applications is not feasible.

Supervised learning approaches require fewer samples compared to reinforcement learning, but the samples must be labelled and/or the networks pre-trained on datasets such as ImageNet (Deng et al. (2009)) or Pascal VOC (Everingham et al. (2010)). However, features from the agricultural domain are not well represented by such datasets and so large amounts of new data would need to be collected and annotated, which is both time-consuming and expensive.

Supervised end-to-end learning of high-level control policies directly from RGB input has shown great promise in alleviating the difficulty of annotating training data (e.g. guidance of aerial robots, Loquercio et al. (2018)). However, for autonomous control of aerial robots it is often not practical to acquire accurate ground truth labels for data captured in flight, thus Loquercio et al. (2018) use a car driving dataset from Udacity¹ to train a drone to follow roadways. Their approach uses a network with regression output layer that gives a continuous output and can be taught a wide range of control policies, but requires an expert driver for data capture. In a recent publication, Kaufmann et al. (2018) trained a similar network for drone racing by carrying a drone around a race course to collect training data, which did not require expert steering, but depended on additional sensors and an offline state estimation for data labelling. A simpler data collection approach was employed by Giusti et al. (2015), who collected an extensive dataset for prediction of view orientation on forest trails with a head-mounted three-camera rig, which gave a built-in labelling of orientation (left/straight/right). Based on this, they trained a view orientation classifier, which was used to compute yaw control of a drone from RGB images only. Smolyanskiy et al. (2017) developed this approach further by experimenting with different network architectures and adding lateral control, which showed improved performance and indicated good generalisation

¹ Available at <https://github.com/udacity/self-driving-car>

capabilities within the trail domain. This approach allows ground truth commands to be easily generated during data capture, but is limited somewhat by the type of steering commands that can be learned.

We expect that the IDSIA trail dataset² from Giusti et al. (2015) better captures natural features such as vegetation and soil than for instance the Udacity city driving dataset. Thus, we select this dataset for pre-training our DCNN.

3. METHODOLOGY

We propose to apply the the principles from the trail following method of Giusti et al. (2015) to crop row-following in agriculture, and use trail data for pre-training to reduce the amount application-specific agricultural training data needed. In our preliminary study (Bakken et al. (2018)), this showed promising results on a very limited polytunnel dataset. In this paper, we expand our dataset substantially to polytunnels from several different locations and seasons. We have also extended the label generation procedure and the network architecture to work for regression, in order to provide a continuous output angle. To assess our network’s ability to generalise beyond one single location and setting, controlled experiments are performed with networks trained on differing amounts of polytunnel training data and tested on locations not seen during training. We also assess the performance of our approach for guidance of an autonomous agri-robot with open-loop field tests.

3.1 Network architecture

Our approach is based on Giusti et al. (2015), using the VGG network architecture (Simonyan and Zisserman (2014)) with three output classes and dropout on fully-connected layers. We also implemented a regression network based on this architecture, with one continuous output value from the last layer instead of a three-class output. An overview of our network architecture is shown in Fig. 3.

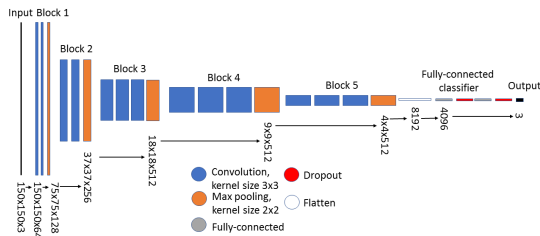


Fig. 3. Network architecture for VGG16 with dropout on fully-connected layers and 3 output classes. For the regression version of the network, the output size is one instead of three.

3.2 Datasets and label generation

Trail data set The IDSIA Swiss Alps trail data set from Giusti et al. (2015) consists of several kilometres of trail recordings. The images are recorded with a rig of three cameras looking left, straight, and right that

² Available at <http://people.idsia.ch/~giusti/forest/web/>

provides the ground truth labels for the three viewpoint classes. The data contains different kinds of trails, and also has some road sections. Most recordings are from the same season, probably late autumn, and have very little green vegetation. See Fig. 4 for example images. As in Smolyanskiy et al. (2017), we used the folders 003, 008, and 010 for testing (*trails test*), and the remainder for training (*trails training*).

Agricultural datasets We recorded a new data set for row following in strawberry polytunnels, with a recording approach similar to that described for the trails data set. The data was captured from five different polytunnels, totalling 3 km of recordings at 5 fps. Our recording setup was a Basler Ace camera with Sunex 190 degree field of view (FoV) fisheye lens. For most recordings, this was mounted approximately 2 m above ground level with a downward tilt of 25 degrees. Some recordings were performed with lower height and less tilt for more variation. Video sequences were recorded travelling straight along the centre of each row, either on board a mobile agricultural robot (Fig. 1) or by hand.

The strawberry polytunnel dataset is divided into three subsets for training and testing: *single polytunnel* consists of data from one row within a single polytunnel at a single point in the growth cycle; *diverse polytunnels* includes data from other rows within the same tunnel as *single polytunnel* as well as three additional tunnels; and *polytunnel test* consists of data from a separate location and season and is used only for testing. For each tunnel, data has been recorded in both directions. Example three-class images from two of five different polytunnels are shown in Fig. 4.

Label generation By employing a wide-FoV camera instead of a fixed rig of three cameras as in Giusti et al. (2015), we are able to extract virtual camera views from arbitrary angles after the recordings were made. This gives much more flexibility than a fixed rig, and makes it possible to train a continuous regression output. Our procedure for extracting virtual camera views was integrated directly into the Keras image augmentation pipeline, such that roll, pitch, and yaw angle offsets could be specified and corresponding virtual views extracted directly at training and test time. In our regression training setup, we specify a fixed roll and pitch and three random yaw angles between -27 and 27 degrees with a 140 degree FoV, per full-FoV image. For classification, three fixed yaw angles of -27, 0 and 27 was used per full-FoV image.

3.3 Training procedures

Our network was pre-trained on the *trails* dataset with additional regularisation, followed by fine-tuning the fully-connected layers of the DCNN (Fig. 3) on our own polytunnel data. A full description of the training setup as well as experiments with hyperparameters and regularisation is given in Bakken et al. (2018). For comparison, we also performed training from scratch on polytunnel data only, using the same setup as with the *trails* dataset, but with a slightly smaller learning rate $1 \cdot 10^{-6}$, until a loss plateau was reached.

For our regression network, the loss function was changed to mean-squared error. The weights in all other layers from

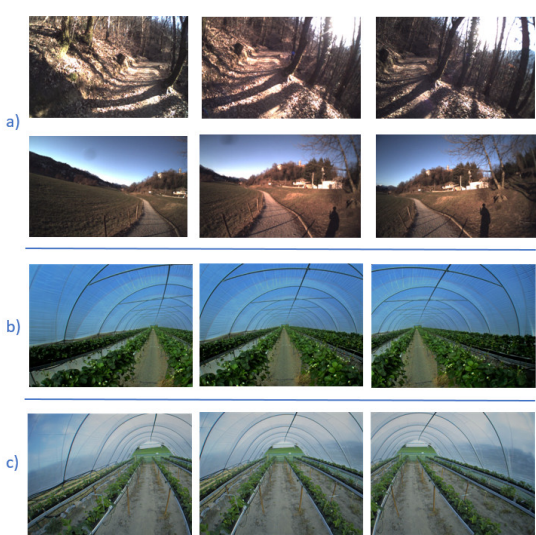


Fig. 4. Example left, straight and right class images from the different datasets: a) *trails*, and two different tunnels from our strawberry polytunnel datasets: b) *single polytunnel* and c) *polytunnel test*

the trails pre-training were kept (since the trails data has only discrete labels), but the fine-tuning was run on our own data with regression labels. The learning rate was slightly lower than for the classification network, $4 \cdot 10^{-7}$, and the training was run until a loss plateau was reached.

3.4 Classification experiments

We compared the classification performance of our network under two different training regimes:

- (1) trained using only data from the *single polytunnel* set, and
- (2) pre-trained on the *trails training* dataset and then fine-tuned using the *single polytunnel* dataset.

Both trained networks were then tested for classification accuracy (against left/right/straight ground truth steering angles) on two data sets (see section 3.2):

- (1) the *polytunnel test* set, containing an unseen polytunnel at a different phase in the growing cycle (significantly different vegetation density) to that of the training set, and
- (2) the *trails test* set containing unseen forest trails.

The same experiments were repeated with training data from the *diverse polytunnels* training set.

3.5 Preliminary field trials

Preliminary field trials were performed in a strawberry tunnel similar to the *polytunnel test* set (not seen during training) with the same robot and camera setup as for data collection. We integrated our DCNN into a ROS node that received a live image stream from the camera

and predicted steering angles in real-time. Our ROS node executed on a laptop CPU on board the mobile robot platform, with a rate of 9 Hz. During this preliminary testing, we operated our system open loop and the robot was steered manually at a speed of 0.4 m/s along a slalom path between crop rows for qualitative performance analysis on live data.

4. EXPERIMENTAL RESULTS

Here we present both quantitative results from offline analysis as well as qualitative results from online field testing. We first compare the steering angle classification accuracy of our network pre-trained on forest trail data and fine-tuned on agricultural data with a network trained fully on agricultural data. We then investigate the usefulness of this approach for steering an autonomous robotic platform performing crop row following, and compare discrete classification network output with continuous regression output.

4.1 Classification accuracy in diverse settings

The results from the classification experiments (described in section 3.4) are summarised in Table 1.

Table 1. Classification accuracy for different training regimes and test cases.

Training data set	Classification accuracy (%)	
	Polytunnel test	Trails test
Single polytunnel	84.0	31.5
Trails + single polytunnel	78.5	85.7
Diverse polytunnels	99.5	48.1
Trails + diverse polytunnels	97.9	78.6

When trained using only data from the *single polytunnel* set, our DCNN performed well in other polytunnel environments, despite the unseen variation in tunnel appearance, camera angle, and vegetation density. However, the same trained network was not able to transfer at all to the general vegetation scenes present in the *trails test* dataset. On the other hand, by pre-training on the *trails training* set and fine-tuning on *single polytunnel* data, we were able to also achieve reasonable performance in all polytunnel environments whilst preserving good performance for general vegetation scenes. Increasing the diversity and amount of polytunnel data using for training (*diverse polytunnels*) enabled our DCNN to improve its classification performance on the general vegetation scenes in the *trails test* set, but still fell well short of the performance of the network pre-trained on *trails* data. These results suggest that the features learned by the network pre-trained on *trails* data are more general than those learned by the network trained only on a specific agricultural setting (*single polytunnel*), and that this approach should therefore generalise more readily to diverse agricultural settings.

4.2 Preliminary field trials

To test the robustness of our fine-tuned network to real world conditions, we implemented our DCNN on board a mobile robot (section 3.5) and drove it through the tunnels found in the *polytunnel test* set, which were not presented

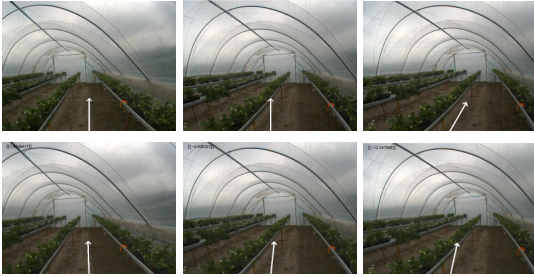


Fig. 5. Snapshots from ROS visualisation video showing predicted steering angle from a classification network (top) and a regression network (bottom) as the robot is turning left. The classification output does not respond to the moderate angle in the middle image. The regression output gives some response to moderate angles, but underestimates both the moderate and the large angle.

during training and included significant seasonal and other differences. Quantitatively, the network fine-tuned on *diverse polytunnels* showed very good performance when tested on the *polytunnel test set*, which contains only three possible steering angles ($[-27, 0, 27]$ degrees, as for the trail data in Giusti et al. (2015)) (Table 1 lower). However, qualitative analysis (Fig. 5) from field tests showed that the sensitivity of the network to smaller steering angle disturbances was not sufficient for autonomous crop row following.

4.3 Regression for continuous steering

To address the poor sensitivity of our network, we re-implemented the top-most layer in our network to give single continuous output (section 3.1). When fine-tuned on the *diverse polytunnels* training set and tested on the unseen *polytunnel test set*, we achieved an RMSE of 5.8 degrees compared to ground truth viewing angles. Qualitative assessment of the field test data shows a much improved sensitivity to steering angle disturbances compared to the classification network, but the regression network has a tendency to underestimate the viewing/steering angle.

5. DISCUSSION

It is important for our use case that the agri-robot is capable of adapting to a new environment with minimal setup effort and cost. It is therefore not feasible to collect training data across all locations, seasons, and conditions in order to fine-tune the network for each new setting. Our hypothesis for this work was that pre-training on a general dataset containing a mix of appropriate features would reduce the amount of training data needed from the specific use case, and furthermore that generic agricultural features could be obtained by pre-training on available data sets containing general vegetation scenes, e.g. forest trails.

5.1 Diverse agricultural settings

The overall classification accuracies for both the specific agricultural setting (polytunnels) and more general setting (forest trails), presented in Table 1, support our hypothesis that features extracted from general vegetation scenes are applicable for agricultural use cases and appear to be more readily generalisable to diverse settings than those obtained from a specific agricultural setting. However, further research is required to prove or disprove our hypothesis that pre-training on a general dataset reduces the amount and/or diversity of training data required from the particular use case. In our results presented here (Table 1), the same quantity of polytunnel training data (*single polytunnel* or *diverse polytunnels*) was used for both training of the standalone network and fine-tuning of the pre-trained network, and in fact the standalone network performed better on the *polytunnel test set*. This is not surprising, as training and testing on the same setting (although with differences, see section 3.2) can lead to overfitting and inflated performance measures, and perhaps indicates that our *polytunnel test* could have contained more diversity or perhaps that polytunnel environments contain enough visual cues beyond those of the crops themselves that seasonal variations are not as important.

In future work we plan on expanding our research to more diverse agricultural settings with less structure (e.g. Fig. 2). The benefit (or not) of pre-training on a general setting should be more evident from these test cases.

5.2 Autonomous control

Our open-loop field trials reaffirm our conclusions from the offline polytunnel tests: that viewing/steering angle prediction performs well even for seasons and locations not experienced during training. However, we also identified some important limitations to our initial classification approach, which to some extent has been alleviated by changing to a regression output with continuous angle. The precision of the steering angle is not yet satisfactory, and some adjustments of the training setup are required to improve this. Further field testing will be performed to evaluate this simple yaw-angle based control policy. A natural next step could be to consider adding lateral control as in Smolyanskiy et al. (2017), which requires at least an additional camera for the recording setup. Alternatively, one could learn steering commands directly as in Loquercio et al. (2018), but this increases the complexity of recording training data. To ensure safe and reliable operation for such an end-to-end training approach, we will also investigate methods to recognise if the current environment is outside its scope of operation, and present a confidence measure along with the predicted steering commands. For our test case, the robot is driving at a very slow speed (0.4 m/s), and a processing rate of 9 Hz is more than sufficient for closed-loop execution. However, a GPU could be used for DCNN inference to reduce processing time and accommodate faster driving speeds in future applications.

6. CONCLUSIONS

We have presented an approach for predicting steering angles for an autonomous, crop row-following, agri-robot

using only RGB image input. Our approach employs a deep convolutional neural network (DCNN) and an end-to-end learning strategy to learn steering angles from images labelled with different viewpoints. We leveraged existing open datasets to pre-train our DCNN with naturalistic features, which improved generalisation capabilities compared to training from scratch on data from a specific agricultural setting. Experiments on existing forest trail datasets and our own datasets from an agricultural setting have demonstrated the accuracy of our approach and its ability to generalise to environments and seasonal conditions not experienced during training. Our online field testing on board an agri-robot operating in a strawberry polytunnel demonstrated the feasibility of this approach for autonomous robot guidance, but also revealed some limitations for steering sensitivity, which will be addressed in future work. Our approach promises a flexible alternative to traditional 2D- and 3D-based onboard guidance schemes and with lower setup costs than external-localisation solutions.

7. FUTURE WORK

Our continuing work will focus on investigating the factors affecting the response of our network to seasonal and environmental variations; investigating the performance of our network on diverse agricultural settings; and implementation on board our autonomous agri-robot for closed-loop field testing.

REFERENCES

- Bakken, M., Moore, R., and From, P. (2018). End-to-end learning for autonomous navigation for agricultural robots. *ICRA 2018 Workshop on Robotic Vision and Action in Agriculture*. URL <https://research.qut.edu.au/future-farming/wp-content/uploads/sites/3/2018/06/End-to-end-Learning-for-Autonomous-Navigation-for-Agricultural-Robots.pdf>.
- Bakker, T., Wouters, H., Van Asselt, K., Bontsema, J., Tang, L., Müller, J., and van Straten, G. (2008). A vision based row detection system for sugar beet. *Computers and electronics in agriculture*, 60(1), 87–95.
- Biber, P., Weiss, U., Dorna, M., and Albert, A. (2012). Navigation system of the autonomous agricultural robot bonirob. In *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal.
- Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.
- English, A., Ross, P., Ball, D., Upcroft, B., and Corke, P. (2015). Learning crop models for vision-based guidance of agricultural robots. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 1158–1163. IEEE.
- Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2), 303–338.
- Giusti, A., Guzzi, J., Cire, D.C., He, F.I., Rodríguez, J.P., Fontana, F., Fässler, M., Forster, C., Schmidhuber, J., Caro, G.D., Scaramuzza, D., and Gambardella, L.M. (2015). A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters*, PP(99), 1–1. doi:10.1109/LRA.2015.2509024. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7358076>.
- Halmetschlager, G., Prankl, J., and Vincze, M. (2014). Probabilistic near infrared and depth based crop line identification. In *Workshop Proceedings of IAS-13 Conference on*, 474–482.
- Jiang, G.Q., Zhao, C.J., and Si, Y.S. (2010). A machine vision based crop rows detection for agricultural robots. In *Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on*, 114–118. IEEE.
- Kaufmann, E., Loquercio, A., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. (2018). Deep drone racing: Learning agile flight in dynamic environments. *2nd Conference on Robot Learning (CoRL 2018)*, Zurich, Switzerland.
- Kise, M., Zhang, Q., and Más, F.R. (2005). A stereovision-based crop row detection method for tractor-automated guidance. *Biosystems engineering*, 90(4), 357–367.
- Le, T.D., Ponnambalam, V.R., Gjevestad, J.G.O., and From, P.J. (2019). A low-cost and efficient autonomous row-following robot for food production in polytunnels. *Journal of Field Robotics*, (April), 1–13. doi:10.1002/rob.21878. URL <http://doi.wiley.com/10.1002/rob.21878>.
- Loquercio, A., Maqueda, A.I., Del-Blanco, C.R., and Scaramuzza, D. (2018). DroNet: Learning to Fly by Driving. *IEEE Robotics and Automation Letters*, 3(2), 1088–1095. doi:10.1109/LRA.2018.2795643. URL <http://ieeexplore.ieee.org/document/8264734/>.
- Perez-Ruiz, M. and Upadhyaya, S.K. (2012). GNSS in precision agricultural operations. In *New Approach of Indoor and Outdoor Localization Systems*. InTech.
- Sadeghi, F. and Levine, S. (2016). CAD2RL: Real Single-Image Flight without a Single Real Image. doi:10.15607/RSS.2017.XIII.034. URL <http://arxiv.org/abs/1611.04201>.
- Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1. doi:10.1016/j.infsof.2008.09.005. URL <http://arxiv.org/abs/1409.1556>.
- Smolyanskiy, N., Kamenev, A., Smith, J., and Birchfield, S. (2017). Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4241–4247. doi:10.1109/IROS.2017.8206285. URL <http://arxiv.org/abs/1705.02550>.

PAPER III

ROBOT-SUPERVISED LEARNING OF CROP ROW SEGMENTATION

Marianne Bakken, Vignesh Raja Ponnambalam, Richard J. D.
Moore, Jon Glenn Omholt Gjevestad, Pål Johan From

IEEE International Conference on Robotics and Automation (ICRA) 2021

III

Robot-supervised Learning of Crop Row Segmentation*

Marianne Bakken^{1,2}, Vignesh Raja Ponnambalam¹, Richard J. D. Moore²,
Jon Glenn Omholt Gjevestad¹ and Pål Johan From¹

Abstract—We propose an approach for robot-supervised learning that automates label generation for semantic segmentation with Convolutional Neural Networks (CNNs) for crop row detection in a field. Using a training robot equipped with RTK GNSS and RGB camera, we train a neural network that can later be used for pure vision-based navigation. We test our approach on an agri-robot in a strawberry field and successfully train crop row segmentation without any hand-drawn image labels. Our main finding is that the resulting segmentation output of the CNN shows better performance than the noisy labels it was trained on. Finally, we conduct open-loop field trials with our agri-robot and show that row-following based on the segmentation result is likely accurate enough for closed-loop guidance. We conclude that automatically generating noisy segmentation labels is a promising approach for vision-based row following that can be quickly and easily adapted to new scenes.

I. INTRODUCTION

Automating agricultural practices through the use of robots (i.e. agri-robots, Fig. 1) is a key strategy for improving farm productivity and achieving sustainable food production to meet the needs of future generations. One of the basic requirements for such robots is to be able to navigate autonomously to and from their base station and along the crop rows. Finding robust, fast, and cost-efficient navigation solutions that can generalise across different field types is an active research topic that can facilitate more wide-spread use of agri-robots.

There is a wide range of sensing options for agri-robot navigation, all with different strengths for different types of fields. In open fields, real-time kinematic (RTK) GNSS provides an accurate position for the robot but does not inherently describe the location nor extent of the crops, thus requiring additional setup effort and cost. Onboard sensors such as scanning lidar or machine vision cameras enable direct sensing of the crops and structures surrounding the robot. Lidar-based navigation has been shown to work well in structured environments such as strawberry polytunnels [1]. Vision-based crop row following using RGB images is a well-established strategy, typically employing colour (e.g. greenness) to segment crops from soil, followed by line extraction to locate crop rows [2]. This has been demonstrated to work well for several crop types, particularly where the crop can be imaged from overhead and/or crop rows are well delineated.

*This work was partly funded by The Norwegian Research Council, grant no. 259869

¹ Norwegian University of Life Sciences, 1430 Ås, Norway

² SINTEF Digital, Forskningsveien 1, 0373 Oslo, Norway, marianne.bakken@sintef.no



Fig. 1. The Thorvald II agri-robot platform operating in a strawberry field during data collection. We perform CNN-based segmentation to detect crop rows for visual guidance and propose an automatic labelling strategy for generating training data. During training, a mask representing crop locations is projected onto the camera image using the pose of the robot, measured with a dual-antenna RTK GNSS system. After training, CNN-based image segmentation can be used to guide the robot along the crop rows, without hand-drawn training labels.



Fig. 2. Example appearance variation in strawberry fields on a Norwegian farm. From top left: Thin plants, lanes partly covered with hay, strong shadows, crops with red leaves in autumn, clean lanes without offshoots and lanes covered completely with green offshoots.

With large seasonal variations, as illustrated in Fig. 2 for strawberry crops, greenness index is not always sufficient to get a good separation of plants and lanes. By utilising the recent advances in deep learning, it should be possible to learn a wider variety of features from labelled data. Recent work [3], [4] has shown promising results using Convolutional Neural Networks (CNNs) for semantic segmentation in agricultural scenes. However, hand-labelled training data covering all possible seasonal variations and crop types does not scale very well, and a neural network trained with insufficient data does not necessarily produce features that are more generalisable than traditional methods.

To overcome this limitation we have developed a robot-supervised learning approach that enables us to successfully

train a CNN for semantic segmentation of crop rows without hand-labelled training images, as illustrated in Fig. 1. To achieve this, we utilise knowledge of the sensor setup, structure of the field, and robot pose during the training phase to learn robust features with a CNN that can later be reused on cheaper robot platforms without RTK GNSS, or in sections of the field that do not have GNSS labelled rows. We develop and test our method on data from a strawberry field, but the approach can be applied to any type of field with row-based geometry.

Our hypothesis is that the CNN will be able to learn good features for crop row segmentation despite the reduced accuracy at the borders of the automatically generated labels. We test this hypothesis against hand-labelled real world data and with open-loop field trials. Our test field had relatively limited variation and distinct crop and lane appearances (Fig. 1), which allowed us to isolate and analyse the effect of noisy labels. The ability to generate and label training data on-the-fly will be critical for adapting our system to more complex scenes (Fig. 2).

The main contributions of this paper are: 1) We present an approach for automated generation of training labels for crop row segmentation with a robot platform. 2) Evaluation on real field data show that automatic labelling gives comparable (or better) network performance to manual labelling. 3) Field trials indicate that the segmentation result should be accurate enough for autonomous robot guidance.

II. RELATED WORK

1) *Vision-based crop row following*: Vision-based crop row following in agriculture has been a research topic for decades, and several works have shown accurate and robust row detection for various crop types. To get a good segmentation separating plants from soil, these methods typically involve some variation of greenness identification (e.g. Excess Green Index (ExG) [5]), combined with thresholding and morphological operations. Then, lines are typically estimated in the segmented image with e.g. Hough Transform as in [2], [6] or least squares fit as in [7], [8], [9] to extract paths that can be used for guidance of autonomous robots.

While methods using greenness index as the main feature can do a great job in many types of fields, there are several situations where this approach may fail. The plants can be covered by dirt after a rainfall, seasons may change the spectral signature of the leaves, or the ground can be covered in vegetation due to weed or offshoots (as in the strawberry field in Fig. 2), to name just a few. There are a few examples of classical methods that use other features, like [10] who propose a learning-based method with Support Vector Machines (SVM) to tackle plants covered in dirt after a rainfall, or [11], who uses stereo cameras to create an elevated crop row map. In any case, tailoring new features for each new field/crop type or appearance does not scale well.

2) *Supervised learning*: Supervised deep learning approaches, particularly CNNs for semantic segmentation, have been successfully applied for vision-based guidance

of autonomous vehicles, and more recently also for off-road and agricultural environments. Maturana et al. [12] build their own off-road dataset with semantic labels and elevation maps, and demonstrate autonomous driving on off-road paths. Valada et al. [13] collect data with RGB, NIR and depth from forest roads, and fuse these modalities in a CNN for segmentation that shows good results in challenging light conditions and appearance variations. Recently, learning-based semantic segmentation has also been applied for row following in agri-cultural environments, like tea plantations [3], and our earlier work in strawberry fields [14]. These works all relied on large quantities of manually-labelled training images to learn the different semantic classes.

3) *Self-supervised learning*: One way to overcome the need for manually labelled data is using a self-supervised learning strategy, where labels are automatically generated from the input data. There have been many different approaches to label generation for semantic segmentation, for instance using knowledge of the scene and camera viewpoint [15], other sensor modalities [16], [17], [18], [19], or correspondences [20]. Zeng et al. [15] automatically generate a big dataset with segmentation labels for robot grasping, using knowledge of the setup and camera viewpoint. They showed that features learned in a such a simplified setup perform well in cluttered scenes as well. In mobile robotics, it is more common to use other sensor modalities to guide the training. [16] use a hyperspectral scanner to automatically extract training data for weed classification with RGB camera. For autonomous offroad driving applications, 3D sensors (e.g. stereo cameras or scanning lidars) have been used to initially identify and label ground and non-ground regions in matching imagery [17]. Similar approaches have also been applied to the guidance of tractors in agricultural settings for the classification of driving surfaces [18] and localisation of cut plant material for automatic baling [19]. These approaches all require an initial classification of 3D sensor data in order to generate training labels for the visual classifier.

4) *End-to-end learning*: Another option for avoiding manual labelling is to perform training in an end-to-end manner, i.e. learn some form of control policy directly from input images. Recently, both reinforcement learning [21] and CNN-based approaches [22], [23], have been used for vision-based guidance of mobile platforms. This eliminates the need for detailed per-pixel image labels for training the underlying networks, and simplifies the labelling process. In our previous work, we have shown that this approach can be applied to the guidance of agri-robots [24]. However, end-to-end learning approaches do not separate the process of learning visual features from classification or policy-learning, and their black-box nature can make it hard to adapt the system to new settings or perform troubleshooting. They also require orders of magnitude more training images than supervised semantic segmentation.

III. METHOD

The pipeline for visual crop row following in this paper consists of three steps: 1) automatic generation of training

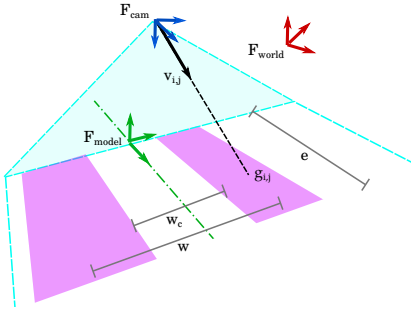


Fig. 3. Illustration of the label projection principle. The (local) virtual field model represent the crop rows as adjacent rectangles, specified by the lane spacing w and the crop width w_c with extent e . The model is projected to the camera image to create semantic labels.

data through label projection, 2) training a CNN for crop row segmentation, and 3) using the result for visual guidance by estimating robot pose from the segmentation. In this section, we focus on label projection, and briefly report the setup of the CNN for segmentation and the procedure for row following for completeness.

A. Automatic label generation

We use camera projection and position of the robot relative to the crop rows to project approximate segmentation labels to camera images, as illustrated in Fig. 3. To perform this projection, we make a few assumptions about the geometry of the field: 1) the crop rows are locally straight and parallel with a fixed width, and 2) the ground is flat and parallel with the robot coordinate frame.

We create a virtual field model using a set of rectangles that are locally aligned with the crop rows. The position and alignment of the virtual model is done in the following way: We measure consecutive points on the crop row centreline with a GNSS receiver, and position reference frame of the virtual model $F_{\text{model},i}$ on each point. The orientation of the rectangles is then aligned with the crop row using a local linear fit of the centreline points.

Using position and heading information from GNSS on the robot, we compute the lateral offset and the yaw angle deviation compared to the local crop row centreline, which is used to transform F_{cam} to the nearest model frame $F_{\text{model},i}$. Then the projection from pixel to ground point is computed using the camera intrinsics and extrinsics.

The geometry of the virtual field model can be adjusted as indicated in Fig. 3. Typically, the lane spacing is fixed, as it corresponds to the wheel spacing of the tractor. The crop width will vary and has to be measured separately for different fields and seasons.

B. CNN for crop row segmentation

Based on the automatically generated labels, we train a CNN for semantic segmentation that gives a per-pixel

classification of the input image with the labels of interest, i.e., crops, lanes, and background in this case. The training procedure and network architecture are straightforward and well-tested, but are listed here for completeness. We use the SegNet [25] implementation from the Keras Image Segmentation Library [26] with ResNet50 [27] as the base model, input size 360×640 , output size 320×176 and 3 output classes. Training is performed with the following setup: categorical cross-entropy loss ignoring the zero class, adadelata as an optimiser, and regularisation through early stopping (choosing the epoch with the lowest validation loss.) The setup is identical for training with automatic and manual labels, but the epoch for early stopping will vary.

For the experiments in this paper, we trained the models on our field dataset as described in IV-A.3.

C. Crop row following

In order to compute steering commands for crop row following, we must estimate the instantaneous heading angle deviation and lateral offset of the robot from the centreline of the crop row. To compute these parameters during open-loop field trials, we used the following approach: 1) The image region corresponding to the active crop row was isolated from the predicted segmentation mask image (CNN output) using a heuristic algorithm. 2) The set of pixels corresponding to the midline of the extracted crop row blob was computed. 3) The set of midline pixels was projected onto the ground plane using the intrinsic and extrinsic calibration parameters for the camera. 4) A robust linear fit was applied to the projected midline points to compute the relative heading deviation and lateral offset of the robot.

IV. EXPERIMENTAL EVALUATION

Our approach is tested with a robot in a real field to evaluate how our approximations and possible inaccuracies in positioning affect the label quality. We then train a CNN based on automatically generated labels and compare the segmentation results to a CNN trained on manual labels. Finally, we evaluate whether our training and segmentation approach is sufficiently accurate for row following with our agri-robot.

A. Experimental setup

1) *Robot and positioning system:* We use the *Thorvald* [28] agri-robot platform from Saga Robotics to collect images and robot pose data in the field. The sensor setup with dual GNSS antennas and camera is shown in Fig. 4.

The dual-antenna GNSS receiver AsteRx4 from Septentrio is used to record accurate robot pose whilst driving in the field. In the current setup, the GNSS receiver is equipped with two AntCom G8Ant-3A4TB1-M1 antennas with approximately 0.5 m separation, which provides high accuracy positions and attitude information (i.e. true heading and roll) at 10 Hz. With RTK GNSS, the position accuracy is estimated to be 1.5 cm horizontally and 3 cm vertically, and the heading accuracy 0.3° with this setup.



Fig. 4. Our data collection setup showing RealSense camera (for this study we use only RGB images) and dual GNSS antennas mounted on Saga Robotics’ Thorvald platform.

Measurements of the static GNSS position of the crop row centreline were obtained manually using a Topcon HIPER SR geodetic GNSS RTK receiver.

Both GNSS receivers utilise corrections from the virtual reference network CPOS from the Norwegian Mapping Authority (NMA) to obtain integer fixed carrier phase RTK GNSS solutions.

The robot has an Intel Realsense D345 camera mounted at centre front, with a tilt of 22.5° downwards. The colour images from the camera have a resolution of 640×480 (we do not use the depth data in this paper), and the framerate was set to 6 fps. Images and position data are synchronised through ROS [29] and data is recorded using rosbags.

2) *Field data collection*: Data collection was performed in a strawberry field with an uneven and hilly terrain with slightly curved rows. The lane spacing is at a fixed 1.25 m, but the width of the crops varies and was measured individually for each row of the recordings.

During data capture, the robot was driven manually along the rows in both directions, in two different patterns 1) straight and centred (approximately) and 2) turning from side to side in a slalom pattern. The driving speed was approximately 0.5 m s^{-1} .

3) *Dataset*: In order to assess the quality of the automatically generated labels and validate the final segmentation result, we required some reference manual image annotations. The manual labelling was performed with the open-source annotation tool Labelme [30] where labels are hand-drawn with piecewise linear boundaries. Background, Crops, and Lanes were assigned labels 0, 1, and 2. The pixels that fall outside the 3 middle crop rows or 2 middle lanes in the image were labelled as background.

For training and testing the CNN, we used images recorded in a slalom pattern to get variation in angular and lateral offset. The training set consists of 195 images from one row, recorded in both directions and sampled at a 20 frames interval to avoid too much overlap between frames. After annotation,

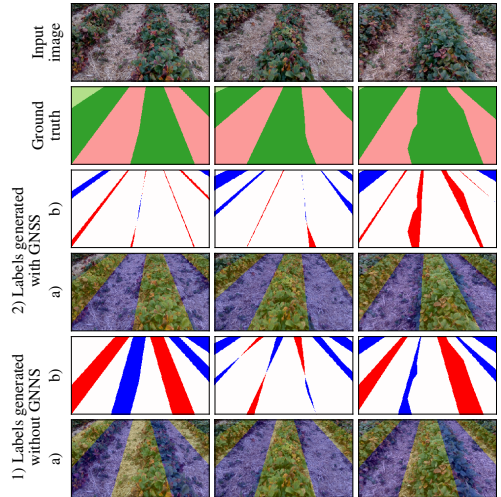


Fig. 5. Examples of automatically generated labels for crop rows in a strawberry field 1) without and 2) with GNSS positions. The label visualisations show a) mask overlaid on image and b) false positives and false negatives (blue) for the lane class.

TABLE I
MASK QUALITY OF AUTOMATIC LABELS, MEASURED IN MEAN IOU COMPARED TO MANUAL LABELS.

Driving pattern	IoU	
	With GNSS	Without GNSS
Straight and centred	0.78	0.80
Slalom	0.79	0.51

20% of the data was reserved for validation during training of the CNN, to choose hyper-parameters. The test set was recorded in a different row from the same field that was not seen during training, by driving in a similar pattern as for the training set. There are 46 images in the test set. This dataset does not cover all the variation shown in Fig. 2, but since we focus on the performance of the labelling approach and not the overall generalisation of the segmentation, we believe it is sufficient for this purpose.

4) *Evaluation metrics*: For quantitative comparisons of label masks and segmentation results, we use frequency-weighted Intersection over Union (IoU) ignoring the background class.

B. Automatic labels

We compare our automatically generated labels with manual hand-drawn labels for the two different driving patterns described above. In addition to the standard setup with GNSS data, we also generated a set of labels without accounting for robot motion with GNSS data, i.e. assuming perfect alignment with the crop row. We report the mean IoU between manual and automatic masks in Table I, and visualisation of a few examples is shown in Fig. 5. When using GNSS data to project the labels, the same performance

TABLE II
SEGMENTATION RESULTS ON TEST SET, MEASURED IN MEAN IOU
COMPARED TO MANUAL LABELS.

Labelling strategy	IoU
Manual	0.93
Automatic with GNSS	0.88
Automatic without GNSS	0.53

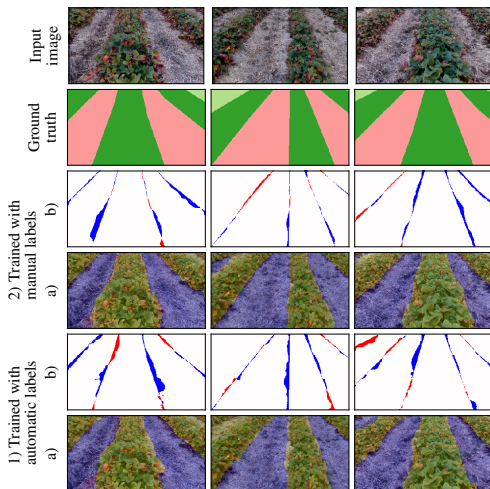


Fig. 6. Example segmentation results for models trained on 1) automatically generated masks with GNSS positions and 2) manual (hand-drawn) labels and. Visualisations show a) segmentation overlaid on image and b) false positives (red) and false negatives (blue) for the lane class.

is achieved for the slalom driving pattern as the straight and centred one.

C. Segmentation

The model for crop row segmentation was trained as described in Section III-B on labels automatically generated with GNSS data, from the same row as shown in Section IV-B. For comparison, we also trained a model on the same images, with manual labels. The model used for testing was picked based on minimum validation error, which was after 9 epochs for the automatic labels and 4 epochs for the manual labels. The models were tested on the separate test set, using manually labelled data as ground truth. The mean IoU of the segmentation masks are reported in Table II, and some example segmentation masks and their pixel-wise errors displayed in Fig. 6.

From the numbers in Table II and the examples in Fig. 6, we see that the model trained on automatic labels performs quite well. The mean IoU of 0.88, is actually slightly higher than the IoU of the masks it was trained on. When larger patches are misplaced, as is the case for labels generated without GNSS in sharp turns, the CNN is not able to learn any general features, as expected.

TABLE III
OPEN-LOOP ROBOT TRIALS. MEAN ABSOLUTE ERROR (MAE) OF
ESTIMATED YAW AND POSITION COMPARED TO GNSS GROUND TRUTH.

	Yaw angle MAE	Lateral offset MAE
Manual labels	0.6°	4.8 cm
Automatic labels	0.1°	0.6 cm
Predicted masks	1.6°	4.6 cm

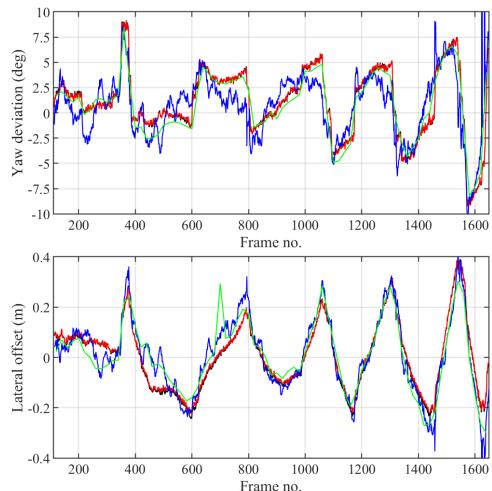


Fig. 7. Yaw angle deviation (top) and lateral offset (bottom) between robot and crop row centreline for a slalom drive. Traces show the yaw/lateral offset estimated from predicted masks (blue) as well as directly from manual labels (green) and automatically generated labels (red), and also the GNSS-based ground truth (black). Best viewed in colour.

D. Open-loop robot trials

We performed a series of open-loop field tests to validate that our CNN trained with automatic labels produced predicted crop masks with accuracy sufficient for closed-loop robot crop following. We compared the yaw angle deviation and lateral offset estimated from the predicted masks (see Section III-C) against the same values estimated from GNSS ground truth.

Fig. 7 shows that both the yaw angle deviation and lateral offset estimated from the segmentation results closely followed the ground truth for the entire dataset. The mean errors between estimated and ground truth values are summarised in Table III. For comparison, yaw deviation and lateral offset were also computed from the training labels, for both manual and automatic labels. Estimating yaw and lateral offset from the automatic labels was predictably closest to GNSS-derived ground truth because the automatic labels were generated from GNSS data. This result at least confirms that our approach for extracting yaw angle and lateral offset from the mask images is valid.

V. DISCUSSION

1) *Automatic labelling*: Our results indicate that when accounting for robot motions with GNSS, the overall alignment of the automatically generated masks is equally good for any driving pattern. However, the assumptions made when generating the mask introduce some errors, and the three most common are summarised in Fig. 8. The first column shows a lateral bias in predicted masks, possibly due to uncorrected roll angle of robot w.r.t. ground plane due to uneven track depths. The second shows a section where the height and width of the crop row is larger than the value estimated at the beginning of the row. In the third, there is a dent in the crop row boundary, that is not captured by straight boundaries of the projected mask. The first issue could be reduced by computing a full 6-DOF pose of the robot, while the other two are expected due to the limitation of the rectangular fixed-size field mask.

2) *Segmentation*: The reported IoU values for the final segmentation actually showed better performance than the automatically generated labels it was trained on, indicating that the neural network was able to learn the general appearance of the classes despite noisy labels along the boundaries. This is probably because of the large amount of good pixel labels per image, which dominate the total loss during training. Closer inspection of the segmentation masks reveal some issues, as shown in Fig. 9. The first two cases show under- and over-estimation of the crop row width, which may arise from the label errors discussed above. As long as this is consistent within each image, it does not introduce any shift for the row following. For the third case, the errors are mostly due to the fact that the segmentation is more fine-grained than the simplified ground truth, which does not capture the detailed curves around the plants.

Finally, it should be noted that the segmentation results reported in this article pertain to a limited test set that does not encompass a large variety of crops or seasonal changes in appearance. However, we believe that this is sufficient for exploring the effect of training with inaccurate segmentation boundaries. The overall performance of the final segmentation and row following system needs to be further evaluated on more data and different crop types, which will be addressed in future work.

Overall, these results indicate that our proposed auto-labelling approach produces guidance information that should be of sufficient accuracy for future closed-loop crop following trials with our agri-robot platform.

VI. CONCLUSIONS

In this paper, we have proposed a new approach for automated labelling for crop row segmentation, using GNSS data from an agri-robot in the field. As expected, the simplified mask introduces some labelling errors near the class boundaries, however the resulting segmentation output of the CNN showed slightly better performance than the noisy labels it was trained on. This indicates that the neural network was able to learn general features despite inaccuracies in the labelled crop regions. Our open-loop field trials indicate that

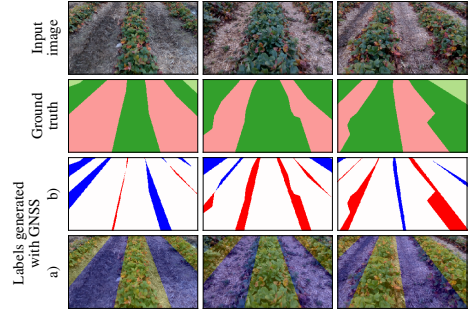


Fig. 8. Three example failure cases for automatically generated masks with GNSS positions. From left: 1) lateral bias in predicted masks; and 2) variation in crop row height and width unaccounted for by the rectangular crop labels; and 3) detail of rough crop boundaries not captured by straight label boundaries. Best viewed in colour.

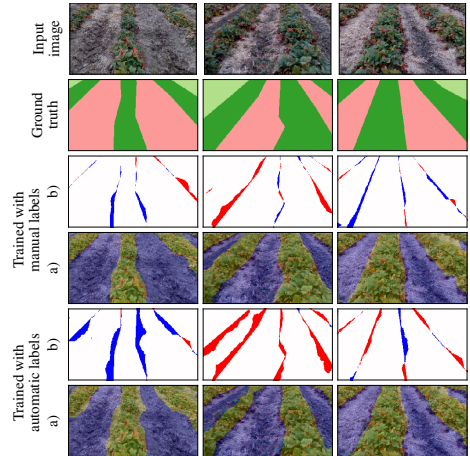


Fig. 9. Three example failure cases for segmentation with model trained on automatically generated masks. From left: 1) and 2) under- and over-estimation of crop row width; and 3) the simplified ground truth does not follow detailed curves around the plants. For reference, segmentation result for model trained with manual labels is also shown. The different visualisations are described in Fig. 6. Best viewed in colour.

the segmentation accuracy is sufficient for row-based guidance of an agri-robot.

We conclude that training with labels that are generated automatically but noisily is a promising approach for quickly and easily adapting a vision-based row following robot to seasonal variations and new crops or fields on-the-go. In future work we will test our approach on a broader dataset to investigate its capabilities for generalisation.

ACKNOWLEDGEMENT

The authors would like to thank Per Fredrik Saxebøl for access to his strawberry farm, as well as Lars Grimstad at NMBU for help with the robot setup, and Sigvald Marholm for proofreading.

REFERENCES

- [1] T. D. Le, V. R. Ponnambalam, J. G. O. Gjevstad, and P. J. From, "A low-cost and efficient autonomous row-following robot for food production in polytunnels," *Journal of Field Robotics*, no. April, pp. 1–13, 2019. [Online]. Available: <http://doi.wiley.com/10.1002/rob.21878>
- [2] J. A. Marchant and R. Brivot, "Real-Time Tracking of Plant Rows Using a Hough Transform," *Real-Time Imaging*, vol. 1, no. 5, pp. 363–371, 11 1995.
- [3] Y.-K. Lin and S.-F. Chen, "Development of navigation system for tea field machine using semantic segmentation," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 108–113, 2019.
- [4] P. Lottes, M. Hörferlin, S. Sander, and C. Stachniss, "Effective vision-based classification for separating sugar beets and weeds for precision farming," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1160–1178, 2017.
- [5] D. M. Woebbecke, G. E. Meyer, K. V. Bargaen, and D. A. Mortensen, "Color indices for weed identification under various soil, residue, and lighting conditions," *Transactions of the American Society of Agricultural Engineers*, vol. 38, no. 1, pp. 259–269, 1 1995. [Online]. Available: <https://pennstate.pure.elsevier.com/en/publications/color-indices-for-weed-identification-under-various-soil-residue->
- [6] B. Åstrand and A. J. Baerveldt, "A vision based row-following system for agricultural field machinery," *Mechatronics*, vol. 15, no. 2, pp. 251–269, 3 2005.
- [7] X. Zhang, X. Li, B. Zhang, J. Zhou, G. Tian, Y. Xiong, and B. Gu, "Automated robust crop-row detection in maize fields based on position clustering algorithm and shortest path method," *Computers and Electronics in Agriculture*, vol. 154, pp. 165–175, 11 2018. [Online]. Available: www.elsevier.com/locate/compag
- [8] I. García-Santillán, J. Miguel Guerrero, M. Montalvo, G. Pajares, and G. Pajares pajares, "Curved and straight crop row detection by accumulation of green pixels from images in maize fields," *Precision Agriculture*, vol. 19, pp. 18–41, 2018. [Online]. Available: <https://doi.org/10.1007/s11119-016-9494-1>
- [9] A. Ahmadi, L. Nardi, N. Chebrou, and C. Stachniss, "Visual Servoing-based Navigation for Monitoring Row-Crop Fields," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2020.
- [10] J. M. Guerrero, G. Pajares, M. Montalvo, J. Romeo, and M. Guijarro, "Support Vector Machines for crop/weeds identification in maize fields," *Expert Systems with Applications*, vol. 39, no. 12, pp. 11 149–11 155, 9 2012.
- [11] M. Kise, Q. Zhang, and F. Rovira Más, "A stereovision-based crop row detection method for tractor-automated guidance," *Biosystems Engineering*, vol. 90, no. 4, pp. 357–367, 4 2005.
- [12] D. Maturana, P.-W. Chou, M. Uenoyama, and S. Scherer, "Real-time semantic mapping for autonomous off-road navigation," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds. Cham: Springer International Publishing, 2018, pp. 335–350.
- [13] A. Valada, G. L. Oliveira, T. Brox, and W. Burgard, "Deep multi-spectral semantic scene understanding of forested environments using multimodal fusion," in *2016 International Symposium on Experimental Robotics*, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds. Cham: Springer International Publishing, 2017, pp. 465–477.
- [14] V. R. Ponnambalam, M. Bakken, R. J. Moore, J. Glenn Omholt Gjevstad, and P. Johan From, "Autonomous crop row guidance using adaptive multi-roi in strawberry fields," *Sensors*, vol. 20, no. 18, p. 5249, 2020.
- [15] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 1386–1383.
- [16] A. Wendel and J. Underwood, "Self-supervised weed detection in vegetable crops using ground based hyperspectral imaging," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 5128–5135, 2016.
- [17] S. Zhou, J. Xi, M. W. McDaniel, T. Nishihata, P. Salesses, and K. Iagnemma, "Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain," *Journal of Field Robotics*, vol. 29, no. 2, pp. 277–297, 2012.
- [18] G. Reina and A. Milella, "Towards autonomous agriculture: Automatic ground detection using trinocular stereovision," *Sensors*, vol. 12, no. 9, pp. 12 405–12 423, 2012.
- [19] M. R. Blas and M. Blanke, "Stereo vision with texture learning for fault-tolerant automatic baling," *Computers and electronics in agriculture*, vol. 75, no. 1, pp. 159–168, 2011.
- [20] M. Larsson, E. Stenborg, C. Toft, L. Hammarstrand, T. Sattler, and F. Kahl, "Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 31–41.
- [21] G. Ryou, Y. Sim, S. H. Yeon, and S. Seok, "Applying asynchronous deep classification networks and gaming reinforcement learning-based motion planners to mobile robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6268–6275.
- [22] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018. [Online]. Available: <http://ieeexplore.ieee.org/document/8264734/>
- [23] A. Giusti, J. Guzzi, D. C. Cireşan, H. Fang-Lin, J. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.
- [24] M. Bakken, R. J. Moore, and P. From, "End-to-end learning for autonomous crop row-following," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 102–107, 2019.
- [25] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [26] D. Gupta, "Implementation of various Deep Image Segmentation models in keras." <https://github.com/divangupta/image-segmentation-keras>, 2017.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [28] L. Grimstad and P. J. From, "The Thorvald II Agricultural Robotic System," *Robotics*, vol. 6, no. 4, 2017.
- [29] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: <https://www.ros.org>
- [30] K. Wada, "labelme: Image Polygonal Annotation with Python," <https://github.com/wkentaro/labelme>, 2016.

PAPER IV

PRINCIPAL FEATURE VISUALISATION IN CONVOLUTIONAL NEURAL NETWORKS

Marianne Bakken, Johannes Kvam, Alexey A. Stepanov,
Asbjørn Berge

Lecture Notes in Computer Science, vol 12368, Proceedings of European
Conference on Computer Vision (ECCV) 2020.

https://doi.org/10.1007/978-3-030-58592-1_2

Principal Feature Visualisation in Convolutional Neural Networks [★]

Marianne Bakken^{1,2}[0000-0003-4958-8194], Johannes Kvam¹,
Alexey A. Stepanov¹, and Asbjørn Berge¹

¹ SINTEF Digital, Forskningsveien 1, 0373 Oslo, Norway
marianne.bakken@sintef.no
<https://www.sintef.no/en/>

² Norwegian University of Life Sciences (NMBU), 1432 Ås, Norway

Abstract. We introduce a new visualisation technique for CNNs called Principal Feature Visualisation (PFV). It uses a single forward pass of the original network to map principal features from the final convolutional layer to the original image space as RGB channels. By working on a batch of images we can extract contrasting features, not just the most dominant ones with respect to the classification. This allows us to differentiate between several features in one image in an unsupervised manner. This enables us to assess the feasibility of transfer learning and to debug a pre-trained classifier by localising misleading or missing features.

Keywords: Visual explanations, deep neural networks, interpretability, principal component analysis, explainable AI

1 Introduction

Deep convolutional neural networks (CNNs) have had a significant impact on performance of computer vision systems. Initially they were used for image classification, but recently these methods have been used for pixel-level image segmentation as well. Segmentation methods are able to capture more information, but require significantly more expensive labelling of training data. Moreover, classification (bottleneck) networks are still used for many applications where the problem can't be formulated as a segmentation task or pixel-wise labelling is too expensive.

One of the main issues with bottleneck networks is that they provide no visual output, that is, it is not possible to know what part of the image contributed to the decision. As a consequence, there is a demand for methods that can help visualise or explain the decision-making process of such networks and make it understandable for humans.

A range of visualisation and explanation methods have been proposed. Class Activation Mapping, e.g. [10], is a computationally efficient way to show the support of a class in the input image, but the resulting heatmap is quite coarse.

[★] Funded by The Norwegian Research Council, grant no. 259869

Gradient-based methods like [3] give a more localised response, but require back-propagation through the whole network, and is very sensitive to edges and noise in the input image.

All these methods operate in a *supervised* manner on one category or feature at a time. In contrast, our method is *unsupervised* and visualise several categories or features in one pass. It can be applied directly to any bottleneck network without any additional instrumentation.

Our approach provides a visualisation that maps the principal contrasting features of a batch of images to the original image space in a single forward pass of the network. We target bottleneck networks, such as image classifiers, and use a singular value decomposition on the feature map of the layer we wish to visualise, e.g., the final convolutional layer, to extract the principal contrasting features for a batch of images. These features are then interpolated back to the original image space, and the activation maps of the earlier layers are used to weight the resulting feature visualisation. An overview of the method is shown in Fig. 1.

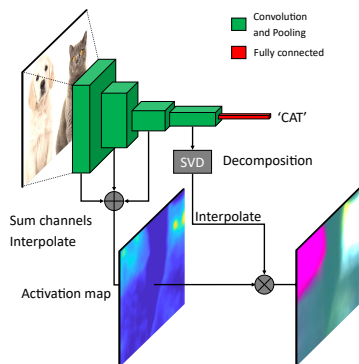


Fig. 1. Overview of our Principal Feature Visualisation (PFV) method.

The main advantages of our method are:

1. Contrast: Per-pixel visualisation of the principal contrasting features.
2. Lightweight: Requires a single forward pass of the original unmodified network, using only intermediate feature maps.
3. Easy to interpret: suppresses non-relevant features.
4. Unsupervised: No additional input or prior knowledge about image classes is required.

We show how the advantages of the method allow it to be used as a tool for debugging misclassification and assessing the feasibility of transfer learning in Section 5.

Our code is publicly available at <https://github.com/SINTEF/PFV>.

2 Related Work

Several categories of methods to interpret CNNs have been proposed. We focus on the methods that provide a visual human-understandable representation, in particular methods that relate the attention of the network back to the original image space in the form of masks or heatmaps.

One way of attributing classifier decision to location in the input is to perform simple perturbations (e.g. occlusion) to the input [16,13] and make a heatmap per class based on change in the output. Similarly, more advanced methods for perturbation of the input image has been proposed [9,2]. The drawback of these methods is that the number of required forward passes is proportional to the number of classes and resulting heatmap resolution.

Other methods focus on localisation of semantically meaningful concepts in the input. For instance by extracting and clustering superpixels, and then compute the saliency as a ranking [7] over these extracted “concepts” [6]. Network dissection is another direction [4], where the response in network hidden units (convolutional layers) are scored according to a predefined set of visual concepts.

Gradient-based visualisation is a group of methods that provide more localised responses and are widely cited in literature. The simplest form of this is to compute the partial derivatives of the output with respect to every input pixel [13]. Several additions to this principle, for instance DeepLIFT [12], Guided Backpropagation [15] and Layer-wise Relevance Propagation (LRP) [3], has improved the localisation and visual appeal. However, as showed through simple sanity checks in [1], many of these methods rely too much on information from the input image, and are actually insensitive to changes in the model. Additionally, they can require a lot of instrumentation, such as special types of layers and separate training of hyperparameters.

Class Activation Mapping provides a direct mapping from the class score to the activations from the forward pass of a CNN. The original work in [5] required a special network architecture, but Grad-CAM [10] provided a more general way to compute the mapping by backpropagation from the class score to the last convolutional layer (not all the way back to the inputs as pure gradient-based methods). Grad-CAM passes the sanity checks in [1], but gives a less localised response than gradient-based methods, and still requires backpropagation from each class to produce responses from multiple classes or objects. Our approach use the activations from the forward pass in a similar manner as Grad-CAM, but rather than computing a mapping through backpropagation, we do a simple unsupervised learning during the forward pass.

Some methods include counter-evidence to give a richer explanation. Grad-CAM and LRP for instance, suggest using negative gradients in addition to the

positive ones to show evidence against a class. In [17], a top-down attention propagation strategy is proposed, that performs backpropagation of both positive and negative activations to create a contrasting visualisation. Our method provides an inherent contrast, and does not need to treat this specifically.

There are also several methods that apply clustering or spectral techniques for model explanation. One such method [8] applies spectral clustering on a set of relevance maps computed with LRP, and performs eigengap analysis and t-SNE visualisation to identify typical prediction strategies. This requires several steps of processing, and is applied on one class at a time. Another work [11] uses Eigenspectrum analysis of the feature maps in neural networks to optimise neural architectures and understand the dynamics of network training. Our approach uses spectral information in a similar manner to these approaches, but to our knowledge is the first one to project this type of information back to image space in one pass.

Compared to existing explanation methods, we aim for an approach that is simple to execute, that depends on activations from the network itself rather than edges in the input image, and can highlight the contrast between several features and classes in one pass.

3 Principal Feature Visualisation

3.1 Method description

Our goal is to obtain a low-dimensional representation of the feature space of feed-forward bottleneck networks which can be mapped to the original image space. Such a visualisation should be achieved in an efficient manner by using a single forward pass of the network, without any additional instrumentation.

Principal component analysis (PCA) projects a signal onto a set of linearly uncorrelated variables (principal components) ranked by the amount of variance explained in the original signal. Conveniently, the projection of features onto these components introduces an implicit measure of contrast, due to the orthogonality of the components.

In brief, our method decomposes a feature map into its principal contrasting features for a batch of images. This is accomplished by extracting principal components through singular value decomposition. The decomposed feature map is then interpolated back to the original image space, where we use the activation maps in the preceding layers as spatial weighting. An overview of the method is shown in Fig. 1, and we describe it in detail below.

Consider a CNN with N convolution and pooling layers. For each layer l a feature map F_l is an $n_B \times n_{c,l} \times n_{x,l} \times n_{y,l}$ matrix, where n_B is the number of images passed through the layer (batch size), $n_{c,l}$ number of channels and $n_{x,l}, n_{y,l}$ is the spatial size of that layer. We denote by $(n_{x,0}, n_{y,0})$ the size of original input images.

Suppose we want to visualise the last convolutional layer N . Our method proceeds as follows. First, for each intermediate F_l we calculate activation maps

for each image in batch

$$A_l^b(i, j) = \sum_{c=1}^{n_{c,l}} F_l(b, c, i, j), \quad b \in \{1, \dots, n_B\} \quad (1)$$

We then compute the total activation map A^b for each batch image as a sum of upsampled activation maps for each layer. That is

$$A^b = \sum_{l=1}^{N-1} P(A_l^b; n_{x,0}, n_{y,0}), \quad (2)$$

where $P(A_l^b; n_{x,0}, n_{y,0})$ denotes upsampling of A_l^b back to original input image size.

Now consider the feature map F_N of the final layer. Our approach is to use PCA to decompose the features for visualisation. First, we reshape F_N to a $n_{c,N} \times (n_B \cdot n_{x,N} \cdot n_{y,N})$ matrix. In this way we treat each per-pixel channel response as a separate observation. We denote this reshaped matrix as F' and centre it by subtracting mean values:

$$F' = F' - \bar{F}' \quad (3)$$

Then we find the principal feature responses by decomposing F' using singular value decomposition as

$$F' = USV^T, \quad (4)$$

where S is a diagonal matrix containing the singular values and U is the decomposition of F' into the space described by the eigenvectors V .

The principal components are then the sorted columns of the following matrix

$$F_{\text{PCA}} = US = [\mathbf{d}_1 \quad \dots \quad \mathbf{d}_r] \quad (5)$$

For visualisation convenience, we choose a subset of F_{PCA} columns $\{\mathbf{d}_1, \dots, \mathbf{d}_{n_d}\}$. For the rest of the paper we assume $n_d = 3$, which allows us to visualise F_N by mapping $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ to red, green and blue channels. We denote by D_N a matrix consisting of these columns

$$D_N = [\mathbf{d}_1 \quad \mathbf{d}_2 \quad \mathbf{d}_3] \quad (6)$$

By reshaping D_N back to $n_B \times 3 \times n_{x,N} \times n_{y,N}$ size and treating each batch image as a separate D_N^b we can upsample D_N^b back to the original size $(n_{x,0}, n_{y,0})$. We use the activation map A^b to weight the upsampled D_N^b and normalise the result as follows

$$V^b = \text{normalise}(A^b \circ P(D_N^b; n_{x,0}, n_{y,0})), \quad (7)$$

where \circ is an element-wise product and P is upsampling operator. Note that the colours in the final images V^b are relative to the processed batch.

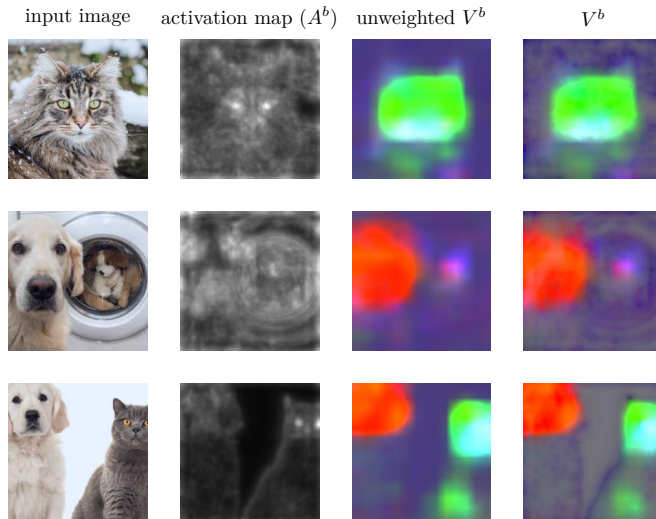


Fig. 2. Variations of our Principal Feature Visualisation method applied on a pre-trained bottleneck CNN (VGG16) and a batch of dog and cat images. The activation map A^b is used to weight the feature map. Colours represent the strongest principal features of the batch and their location in image space. Best viewed in colour.

3.2 VGG Example

We illustrate the properties of our method with a simple example of a few dog and cat images and a VGG16 network [14] pre-trained on ImageNet.

First, we show the final visualisation V^b together with two intermediate steps: the activation maps A^b , and *unweighted* V^b from upsampling directly without weighting. V^b was computed with a forward pass on a batch of six images of dogs and cats. The intermediate activation maps A_i^b were extracted before each max pool layer, and the feature map of the final layer, F_N , was extracted before the last max pool layer. We used bilinear interpolation for upsampling. The results are shown in Fig. 2. For this batch, the principal feature maps assign different colour channels to dogs, cats and background. Studying the intermediate steps, we see that the principal feature map without weighting shows more response from the channel in the background. The weighting with earlier activation maps thus enhances the *strongest* features, while the principal components provides *contrast* between different features.

Second, we illustrate how the visualisation depends on the composition of the input batch. Fig. 3 shows our method applied on different single-image input batches. The colours now represent different features within that image only. For

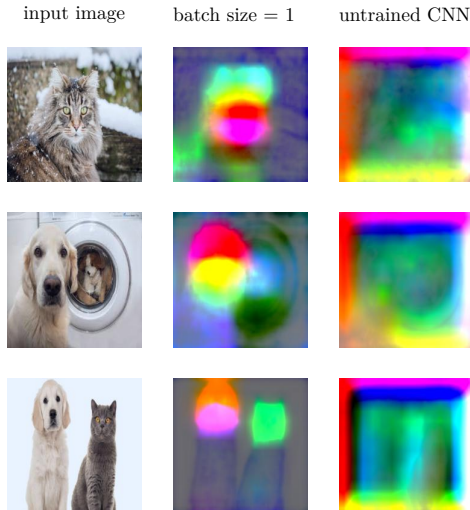


Fig. 3. Batch size illustration and sanity check with untrained network. Second column shows the visualisation of single-image input batches. The colours now represent different high-level features like ears and nose rather than the class-level features in Fig. 2. Third column shows a simple sanity check: the visualisation of an untrained network of randomly initialised weights. The result is completely different, as expected. Best viewed in colour.

the image with two objects, there is still some class-related contrast. This brief example indicates that batch composition can be used deliberately as a tool to control the contrast in the visualisation and tailor it to any application. More examples of this are shown in Section 5.2 and supplementary material.

In order to be useful for model debugging, a visualisation method should be sensitive to the model parameters. We perform a simple parameter randomisation test as suggested in [1], by running our method on a randomly initialised untrained version of the network. As seen in Fig. 3, the resulting visualisation of the random model is visually very different from the pre-trained one. This indicates model sensitivity in our visualisation, which can be used for debugging the training process.

4 Comparison with other methods

We compare our method (PFV) with Grad-CAM [10] and Contrastive Excitation Backprop (c-EBP) [17] on VGG16 pre-trained on ImageNet. We use a batch of images that is not included in ImageNet, but contains objects of ImageNet

categories. A few examples are shown in Fig. 4, where we have used the top-3 predicted classes as targets for Grad-CAM and c-EBP.

Grad-CAM and c-EBP are supervised methods based on backpropagation, that generate a heatmap conditioned on the predicted class. Consequently, these methods highlight evidence for a particular class, and suppress sources that do not contribute to the decision. Contrastive EBP approximates the probability that a given image pixel contribute positively or negatively to the decision. When the target classes are unknown and we simply specify them as the top-k predictions, these methods require a potentially large number of backward passes to describe the feature diversity in the image.

In contrast, our PFV is an unsupervised method calculated based on a single forward pass, that highlights the principal contrasting features in a batch of images. As our method is based on principal components which form an orthogonal basis where one component cannot explain another, it focuses on feature variance instead of evidence for a decision. The colours of PFV represent different features, with no direct connection to the final classification. However, by performing PFV on a batch of images, e.g. the three images in Fig. 4, colours are consistent across the batch and show which objects that have similar features.

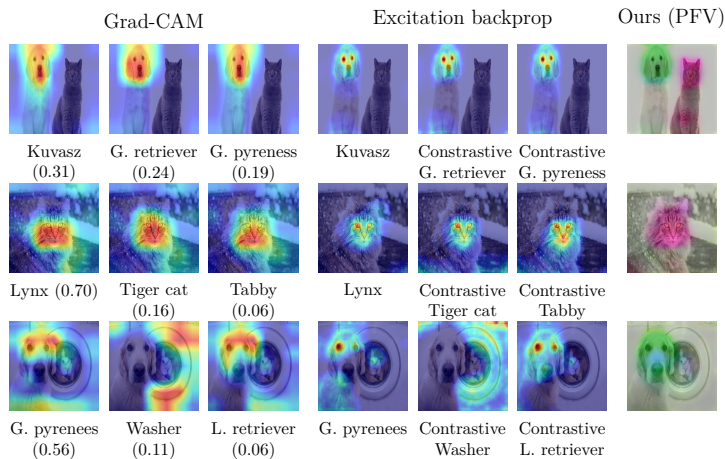


Fig. 4. Comparison of GradCAM, Contrastive Excitation Backprop (c-EBP) and PFV on VGG16 pre-trained on ImageNet. Grad-CAM and c-EBP results are shown for the top-3 predicted classes. PFV results is for a batch of the three images shown. Colours represent heatmaps for Grad-CAM and c-EBP, and principal features for PFV. Best viewed in colour.

5 Applications

In this section we apply our method to two use-cases: debugging misclassified examples by localising misleading and missing features in the input image; and ad hoc prediction of the success of transfer learning with a pre-trained network.

5.1 Debugging classification errors

When a network fails to classify an image correctly, it can be hard to know what part of the image is to blame. We show how our method can be used to identify misleading or missing features and their location in the image by comparing principal feature maps of incorrectly and correctly classified samples.

To do this, we apply PFV on an example task: dog breed classification. There are 120 dog breeds among the 1000 categories of the ImageNet dataset, and the features of the pre-trained VGG16 network should therefore be well suited for this task. We ran prediction on a handful of images of the class “English Springer Spaniel” not present in the original dataset, and identified the failed samples. It turns out that all the failed samples show dogs in water, and we want to examine why they fail. Is it because of the water, occlusion of body parts, or something else?

We applied the following procedure: For each misclassified sample, PFV was applied on a batch of six correctly classified samples; three of the true class and three of the mistaken class. To aid the comparison of the PFV images, we also plot the distribution of red, green and blue in the foreground of the PFV image, i.e., the three strongest principal components.

Figure 5 shows the result of running PFV on two batches of images containing two misclassified images: Batch A (“Springer spaniel” misclassified as “goose”) and Batch B (“Springer spaniel” misclassified as “Sussex Spaniel”). To identify missing or misleading features, we compare the PFV distributions of the other images in the batch with the failed sample, and look for the location of the colours with large deviation. In the left case (Batch A), the misclassified sample has a red component on the head as in the true class “springer”, but is missing the red component on the rest of the body. It also has a strong green component on the body as in “goose”. In the right case (Batch B), the misclassified sample is missing the strong green component located on the white fur in front in the “springer” image, and the PFV distribution is more similar to that of “sussex spaniel”, which has no white fur. For both cases, the location of the missing features reveal that the failed classifications can most likely be blamed on body parts occluded by water.

This example shows that our method can be used to localise missing or misleading features, because it highlights the *contrasting* features within a batch, not just the most dominant features from the classification.

5.2 Transfer learning

Transfer learning is often applied when there is limited training data available to train a deep neural network from scratch. In this section we show that it is

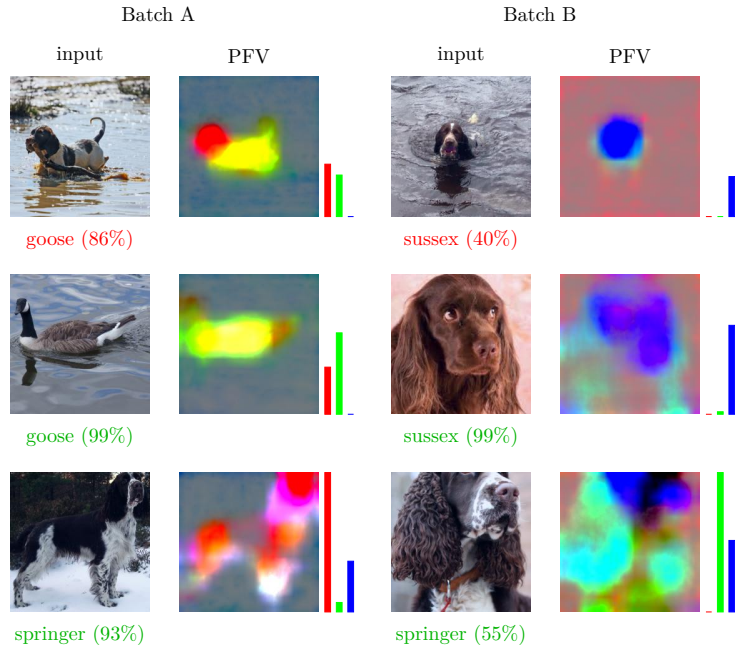


Fig. 5. Principal Feature Visualisation (PFV) on misclassified samples compared to correctly classified samples. In the first row, the two input images are of the category “English Springer Spaniel”, but has been classified as “goose” and “Sussex Spaniel”. In the second and third row, the input images are examples from the two different PFV batches. Bars show the distribution of red, green and blue foreground pixels of the PFV image. The colour encoding is not consistent because the method is applied on two different batches, and hence the principal vectors are different. Best viewed in colour.

possible to predict the success or failure of transfer learning on a new dataset by visualising the principal features of the pre-trained network on images from this dataset.

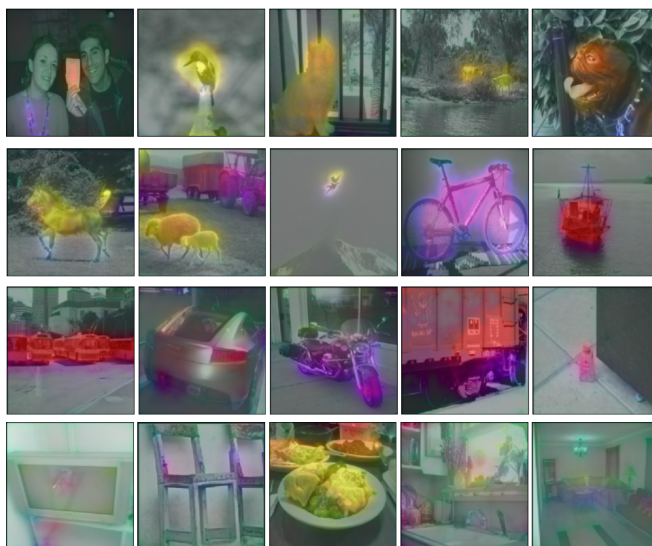


Fig. 6. Initial principal feature visualisation of VGG16 features on the Pascal VOC2012 dataset. The dataset contains 20 classes, features are visualised for a random example from each class. Similar colours indicate similar features. Best viewed in colour.

We analyse the features of VGG16, pre-trained on ImageNet, applied to the Pascal VOC2012 dataset.

Initially, we randomly sample one image from each of Pascal VOC2012's 20 classes and form a batch of these images. We then apply PFV and visualise the principal contrasting features of this batch, shown in Fig. 6. For simplicity, the feature visualisations are shown as an overlay to a grey scale version of the input image. As the images are quite dissimilar, decomposing the features of the images into three principal features, only gives us a coarse indication of which examples contain similar feature sets. Based on this visualisation we observe that the animal classes appear to have similar features, while vehicles and bicycles appear to have a different set of features. Interestingly, we also see observe that there are only weak feature responses for chair, sofa and potted-plant, while for the class dining-table, the main responses are from the objects on the actual table.

To further investigate the difference between the features in the animal categories, that have similar colours in Fig. 6, we randomly sample new batch of images from these categories. This time, we sample 4 random images from each of the categories: “dog”, “cow”, “cat”, “horse” and “sheep”. We then again apply PFV to find the principal contrasting features for this batch of images, shown in Fig. 7. Note again, that the colours in the images are relative to each batch. As the class variation in this batch of images is lower than in the initial experiment, we observe that we obtain a finer decomposition. Here we see that cats and dogs become more clearly separated from the other classes. The other three classes; cows, horses and sheep, does appear to contain similar features. In addition, one example from the “dog” class and one from the “cat” class appear as outliers, which might be due to the images being difficult examples or that ImageNet contains multiple cat and dog breeds.



Fig. 7. Principal feature visualisation of VGG16 features on the Pascal VOC2012 dataset for the classes; “dog”, “cat”, “cow”, “horse” and “sheep”, with a batch of four random examples sampled from each class. Similar colours indicate similar features. Best viewed in colour.

Based on this analysis we hypothesise that in a fine-tuned model using VGG16 ImageNet features, we expect little confusion between the cat and dog class, a more pronounced confusion between the “horse”, “cow” and “sheep” classes. In addition, the weak feature responses for classes “chair”, “diningtable”

and “sofa”, indicate an overall poorer performance in the detection of these classes.

To check this hypothesis we fine-tune VGG16 pre-trained on ImageNet on the Pascal VOC2012 dataset. We retrain only the final fully connected layer (the classifier), the rest of the network (i.e., all convolutional layers) is kept fixed during training. For simplicity we only select images containing one class per image, to be able to use a standard cross-entropy loss in the optimisation. We train until the validation loss stops decreasing and investigate the final performance in terms of a confusion matrix. The confusion matrix is shown in Fig. 8.

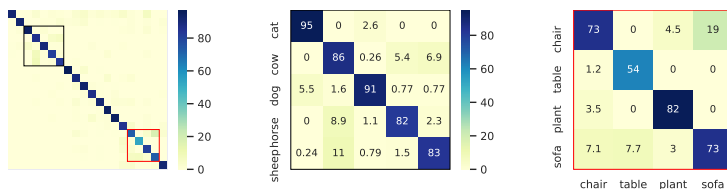


Fig. 8. Confusion matrix for the validation set for a VGG16 network, after fine-tuning on Pascal VOC2012. Left, overall view of confusion matrix. Middle, confusion between classes “cat”, “cow”, “dog”, “horse” and “sheep”. Right, confusion between classes “chair”, “table”, “plant” and “sofa”. Best viewed in colour.

The worst performing categories are of the classes “dining table”, “sofa”, and “chair”. We also observe that “cow” is significantly confused with classes “horse” and “sheep”. These observations suggest that such a feature visualisation strategy can give an intuition about when pre-training will be beneficial and when it might fail.

6 Conclusion

We have presented a method for visualising the principal contrasting features of batch of images during forward pass of a bottleneck CNN. Our approach has several advantages over related methods, namely that it combines low overhead with intuitive visualisation, and doesn’t require any user input or modification of the original CNN. We have shown how these advantages allow us to interpret the performance of CNNs in two common settings: debugging misclassification and predicting the applicability of transfer learning.

Our code is available at <https://github.com/SINTEF/PFV>.

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: *Advances in Neural Information Processing Systems*. vol. 2018-Decem, pp. 9505–9515. Neural information processing systems foundation (10 2018), <http://arxiv.org/abs/1810.03292>
2. Agarwal, C., Schonfeld, D., Nguyen, A.: Removing input features via a generative model to explain their attributions to an image classifier’s decisions (2019), <http://arxiv.org/abs/1910.04256>
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* **10**(7), 1–46 (2015). <https://doi.org/10.1371/journal.pone.0130140>
4. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017)
5. Bolei Zhou and Aditya Khosla and Agata Lapedriza and Aude Oliva and Antonio Torralba: Learning Deep Features for Discriminative Localization. *CVPR* **2016**(1), M1–M6 (8 2016). <https://doi.org/10.5465/ambpp.2004.13862426>, <http://journals.aom.org/doi/10.5465/ambpp.2004.13862426>
6. Ghorbani, A., Wexler, J., Zou, J., Kim, B.: Towards Automatic Concept-based Explanations. *NeurIPS* (2 2019), <https://github.com/amiratag/ACE><http://arxiv.org/abs/1902.03129>
7. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., Sayres, R.: Interpretability beyond feature attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In: *35th International Conference on Machine Learning, ICML 2018*. vol. 6, pp. 4186–4195 (2018)
8. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.R.: Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications* **10**(1), 1–8 (2019). <https://doi.org/10.1038/s41467-019-08987-4>
9. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?" Explaining the predictions of any classifier. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. vol. 13-17-Aug, pp. 1135–1144 (2016). <https://doi.org/10.1145/2939672.2939778>
10. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision* **128**(2), 336–359 (2020). <https://doi.org/10.1007/s11263-019-01228-7>, <http://gradcam.cloudcv.org>
11. Shinya, Y., Simo-Serra, E., Suzuki, T.: Understanding the effects of pre-training for object detectors via eigenspectrum. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops* (Oct 2019)
12. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: Precup, D., Teh, Y.W. (eds.) *Proceedings of the 34th International Conference on Machine Learning*. *Proceedings of Machine Learning Research*, vol. 70, pp. 3145–3153. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017), <http://proceedings.mlr.press/v70/shrikumar17a.html>
13. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *2nd International Con-*

- ference on Learning Representations, ICLR 2014 - Workshop Track Proceedings pp. 1–8 (2014)
14. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR **abs/1409.1** (2014). <https://doi.org/10.1016/j.infsof.2008.09.005>, <http://arxiv.org/abs/1409.1556>
 15. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: 3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings (2015)
 16. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **8689 LNCS**(PART 1), 818–833 (2014). https://doi.org/10.1007/978-3-319-10590-1_53
 17. Zhang, J., Bargal, S.A., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-Down Neural Attention by Excitation Backprop. International Journal of Computer Vision **126**(10), 1084–1102 (10 2018). <https://doi.org/10.1007/s11263-017-1059-x>, <http://arxiv.org/abs/1608.00507>

PAPER V

APPLIED LEARNING FOR ROW-FOLLOWING WITH AGRI-ROBOTS

Marianne Bakken, Johannes Kvam, Richard J. D. Moore, Pål From
Submitted to Computers and Electronics in Agriculture, July 2021.

Submitted version.

Applied learning for row-following with agri-robots*

Marianne Bakken^{a,b,*}, Richard J. D. Moore^a, Johannes Kvam^a and Pål J. From^b

^aSINTEF Digital, Oslo, Norway

^bNorwegian University Of Life Sciences, Ås, Norway

ARTICLE INFO

Keywords:

Agricultural robotics
Deep learning
Autonomous navigation
Computer vision
Explainable learning

ABSTRACT

Deep neural networks have the power to be game changing for cost-effective visual navigation solutions such as for agri-robots, but the gap between controlled data science research and deployment in real applications is still a hindrance for further adoption. In this paper, we address several issues that are commonly faced when training neural networks for practical applications: how to efficiently acquire training labels, how to inspect which features the network has learned, and how to ensure robustness against dataset bias. Our proposed solution is two-fold; we apply a feature visualisation method to debug black-box row-following networks, and we propose a novel network architecture and learning approach for row-following that is inherently more explainable and robust than end-to-end networks. Finally, we demonstrate the performance of our method by performing row-following in open-loop field trials with an agri robot in a strawberry field. Our experiments led us to discover an unexpected bias in a public dataset and demonstrated that our proposed hybrid network architecture is robust to dataset bias and can learn meaningful features, even with small quantities of training data.

1. Introduction

Deep neural networks (DNNs) have revolutionised the field of computer vision since their introduction about a decade ago, producing classification performance better than a human baseline on several challenging benchmark datasets. There is increasing adoption of these techniques within robotics and other practical fields as well, but their real-world performances often do not meet expectations. Due to limited quantity and quality of training data, practical DNN applications are more susceptible to problems such as domain shift and dataset bias than controlled data science experiments. Furthermore, DNNs are harder to debug and adapt than classical methods due to their black-box nature. However, the capability to learn complex and generic representations directly from sensor data is powerful, and issues with applied learning should be addressed to realise the full power of DNNs for practical applications.

Agricultural robotics is a rapidly growing field where deep learning could make a huge difference. One of the key technologies required for large-scale adoption of lightweight agri-robots is robust, fast, and cost-efficient navigation solutions. Such solutions must generalise across seasons and be adaptable to different field types. Navigation and guidance of mobile robots is huge field of research, and there are myriad different sensor options including GNSS positioning and scanning LIDARs that are well suited for many applications. Within agriculture, vision-based row-following using RGB-images is a well-established strategy that requires only a single camera and can be adapted to most applications. Classical camera-based methods usually implement some form of greenness index, which has been demonstrated to work well for some crop types. However, if

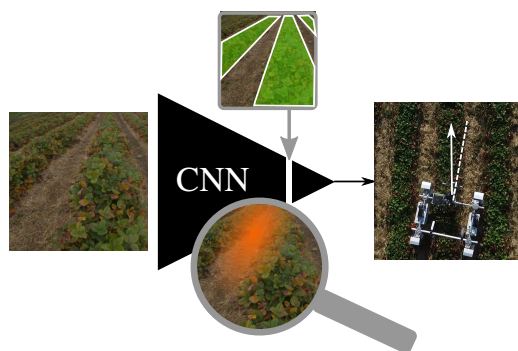


Figure 1: We propose techniques that make learning-based row following less of a black-box: 1) Visualisation of contrasting features with PFV (orange) and 2) enforcing a meaningful representation by adding a segmentation output (green). The proposed system is trained in a data-efficient manner without manual image labels, and is validated with open-loop field trials with an agri-robot.

this approach is to be adapted to applications with very different visual appearance, for instance red crops or overgrown lanes, the features must be manually tailored.

When applying DNNs to the problem of visual row following, it is possible to learn a wide array of visual features directly from images. Instead of tailoring algorithms to every crop type or season, we can train a neural network using examples labelled with the properties it should learn. The challenge is to collect enough labelled data to cover all the variation the robot will ever encounter in the field. Convolutional neural networks (CNNs) for semantic segmentation have been successfully applied for crop row following [1, 2],

* This research was funded by the Norwegian Research Council, grant no. 259869

*Corresponding author

✉ marianne.bakken@sintef.no (M. Bakken)

ORCID(s):

but require large amounts of hand drawn labels, which does not scale well in practical applications. Other strategies enable more efficient capturing and labelling of training data, for instance by training row-following *end-to-end* with different camera view angles that are automatically labelled during data capture [3]. The main drawback of such methods is that they are much less transparent, and have proven difficult to debug and use in practice[4, 5].

For successful development of adaptable and flexible visual guidance solutions in practical applications, we need to close the gap between state-of-the-art neural networks and the classical methods for the application in question. This paper takes the practical problem of row-following in agriculture and highlights solutions to potential problems that will be encountered in this scenario. Specifically, we suggest three techniques to avoid common pitfalls: 1) data-efficient supervision strategies that minimise the labelling effort by trading label quality for data variation, 2) visualisation techniques that help understand what the network has learned, such that adjustments can be made to dataset, network architecture, or training setup, and 3) add additional outputs that can help the network to learn semantically meaningful and robust features, which makes it inherently more explainable and more robust to issues such as dataset bias.

The specific contributions of this paper are: 1) the first demonstration of the Principal Feature Visualisation (PFV) method “in-the-wild”, leading to the discovery of a bias in a public dataset; 2) proposal of a novel hybrid network architecture and supervision approach tailored for the row following problem that is optimised for interpretability, robustness, and data-efficiency, which shows good results even with extremely little data and a biased dataset; and 3) validation of our approach through open-loop trials with an agricultural robot in a strawberry field.

This paper is organised as follows: first, we briefly summarise state-of-the-art on the broad range of topics relevant for this paper, before we go through techniques from our previous work on learning-based row following and introduce the specific problems that motivate the work in this paper; then, the three main contributions will be presented and discussed in three separate sections, *Visualisation of end-to-end learning*, *Hybrid learning*, and *Field trials*; and finally, we summarise our findings and give an overall conclusion.

2. Background

2.1. Related work

Vision-based crop row following Vision-based crop row following in agriculture has been a research topic for decades, and several works have shown accurate and robust row detection for various crop types. To get a good segmentation separating plants from the soil, these methods typically involve some variation of greenness identification (e.g. Excess Green Index (ExG) [6]), combined with thresholding and morphological operations. Then, lines are typically estimated in the segmented image with e.g. Hough Transform as in [7, 8] or least squares fit as in [9, 10, 11] to extract paths

that can be used for guidance of autonomous robots.

While methods using greenness index as the main feature can do a great job in many types of fields, there are several situations where this approach may fail. The plants can be covered by dirt after a rainfall, seasons may change the spectral signature of the leaves, or the ground can be covered in vegetation due to weed or offshoots, to name just a few. There are a few examples of classical methods that use other features, like [12] who propose a learning-based method with Support Vector Machines (SVM) to tackle plants covered in dirt after a rainfall, or [13], who uses stereo cameras to create an elevated crop row map. In any case, tailoring new features for each new field/crop type or appearance does not scale well.

Learning-based crop row segmentation Recently, convolutional neural networks (CNNs) for semantic segmentation have been successfully applied for guidance of autonomous cars, and more recently also for agricultural environments like tea plantations [1] and paddy fields [14], and our previous work in strawberry fields [2]. The main drawback of these methods is they rely on labour-intensive per-pixel labelling of the whole dataset, and do not address how such labels can be adjusted or updated for practical applications with large variation to seasonal appearance.

Another issue with state-of-the-art architectures for semantic segmentation is that they are optimised for benchmarking datasets [15, 16], where the task is to separate many object classes and produce high-resolution object boundaries. This requires many parameters in the network, which makes both training and inference slower, and is more capacity and resolution than necessary for crop row following, which is usually in a simpler environment but with potentially large seasonal variations. For real-time applications, it is desirable to have lightweight network architectures with fewer parameters. This can be achieved, for instance, by reducing the output resolution of the segmentation.

End-to-end learning Performing segmentation as a separate step aligns well with classical approaches for row following, but performing the row and steering estimation in an end-to-end manner can give more flexibility and ease the labelling process. Supervised end-to-end learning of steering commands was first proposed for cars and UAVs. Some works[17, 18] record steering commands and use them as training labels, whilst [19] employ static cameras at different angles to automatically record ground truth viewing angle, eliminating the need for per-pixel image labels and simplifying the labelling process. However, end-to-end learning approaches do not separate the process of learning visual features from classification or policy-learning, and their black-box nature can make it hard to adapt the system to new settings or perform troubleshooting. They also require orders of magnitude more training images than supervised semantic segmentation and are more vulnerable to biases in the dataset.

Automatic labelling Methods for generating training labels automatically can reduce the manual workload required to train CNNs. In [5] we adapted and applied the approach from [3] to crop row following, but we used a single wide-FOV camera and extracted virtual camera views in software, which gave increased flexibility and enabled us to also generate training labels for regression of a continuous heading angle.

In some settings, the geometry of the scene can be employed to simplify and automate the labelling of training data. For example, in [20] we presented a method for automatic projection of crop row labels, based on camera view geometry and GNSS-positions of the robot, assuming locally straight and parallel rows with a fixed width. This produces an approximate mask with straight boundaries. We trained a neural network for crop row segmentation based on these labels, and found that it was able to predict a more accurate segmentation than the approximate labels it was trained on, which indicated that it had learnt to ignore the noise in the labels.

Dataset bias Deep neural networks are more dependent on a good dataset than traditional methods. In particular, bottleneck networks with many parameters – such as those used for end-to-end learning – are susceptible to overfitting on artefacts in the dataset. In machine learning, this phenomenon is known as *dataset bias*. It is usually a by-product of the recording or labelling procedure, where a misleading feature correlates with the label. Several works have demonstrated that neural networks overfit to such biases, e.g. where a poorly balanced training set lead the neural network to learn gender stereotypes and classify male subjects as doctors and female subjects as nurses [21]. [22] show that the trained model focus on a source tag that is present in many horse images in the commonly used Pascal VOC dataset. When the tag is pasted to car images, it is classified as horse. Since the validation data comes from the same dataset, this artefact went unnoticed for a long time. Both examples above applied visualisation techniques to identify these biases.

Visualisation of neural networks To make it easier to interpret and debug convolutional neural networks, several explanatory methods have been proposed. Many of them aim to visualise the attention of the network in form of masks or heatmaps on top of the original image. This is often done by computing the gradient of the output with respect to every pixel in the input image during back-propagation [23, 24, 25]. This gives a localised response for the input pixels which matter most for the classification. Grad-CAM[21] compute the gradients with respect to the last convolutional layer (not all the way back to the input image), and can in this way relate the features from the forward pass to different class responses. Several works [21, 22, 26] use such visualisation techniques to highlight known biases in datasets, which can help identify what changes must be made to the dataset to fix the issue.

In [27], we presented the Principal Feature Visualisation

(PFV) method, which is a lightweight visualisation method that is executed during the forward pass, and shows the features of a CNN as a colour-encoded heatmap. The difference from other visualisation methods is that it shows the *contrasting features* of the CNN relative to the input image, independently of the decision head. This means that our method can be used for debugging the feature extractor itself, and for instance determine if learned features of a model are suitable for transfer learning to a new dataset, as demonstrated with generic image recognition datasets in [27].

2.2. Motivation

For practical applications of deep learning, access to quantities of varied training data is usually limited, and it is not always possible to avoid biases in the recorded data. The capability to detect these biases and avoid overfitting them is crucial for success in practical applications. The work in this paper is specifically motivated by the problem of row-following for agri-robots.

End-to-end learning is a convenient supervision strategy for row-following, which makes it easy to acquire labelled data, but there are some pitfalls that need to be addressed. We believe that the high degree of freedom in our end-to-end network in [5] coupled with minimal training data led to overfitting and poor robustness to artefacts and variations in the dataset. Debugging these issues proved difficult due to the black-box nature of end-to-end learning.

Visualisation techniques can be employed to make end-to-end networks more transparent. In [4], we used activation and saliency maps [24] to debug issues with transfer learning from trails to polytunnels, but this did not give any clear answers. The more recent PFV method [27] can visualise the learned features directly and, as we shall see in section 3, gives new insight into the source of this issue.

Although visualisation can help us understand the issues with end-to-end networks, we need a more transparent training approach that can avoid such problems in the first place. One way of achieving this is to explicitly train a segmentation of the crop rows, as we did in [2]. The promising results with automatic generation of segmentation labels in [4] make this a much more feasible approach. In this work, we build on the network architecture and labelling approach from [20] and combine it with the end-to-end learning approach from [4], to get a transparent architecture that can be trained in an end-to-end fashion.

To summarise, the motivations for our work presented in this article were two-fold:

1. Make the end-to-end approach less ‘black-box’ and be able to detect issues in the training process by use of feature visualisation.
2. Develop a more robust and transparent network architecture that can still be trained in a data-efficient manner.

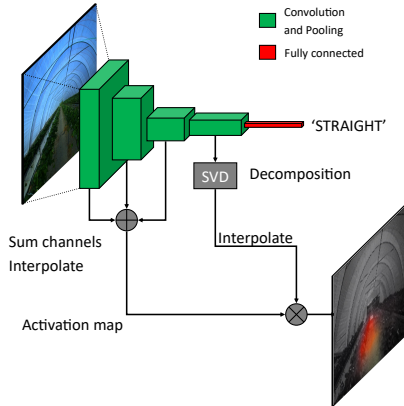


Figure 2: Overview of the PFV visualisation method on a bottleneck CNN.

3. Visualisation of end-to-end learning

We developed the PFV method [27] partly in response to difficulties we experienced transferring learning from trail to agri use cases. In this section we present the results of our investigations with this approach as applied to our agri use case. In particular, our research was centred around two questions: 1) can PFV predict whether transfer learning will work for particular use cases, and 2) can PFV illuminate the underlying issues that cause misclassification after transfer learning?

This is the first demonstration of the PFV method “in-the-wild” in a real application rather than on standard image recognition datasets. For visualisation of neural networks, human perception is an important factor. Therefore, we also present a new colour mapping of the features that is easier to interpret than the original mapping.

3.1. Method

3.1.1. Principal Feature Visualisation (PFV)

In short, the principal feature visualisation method uses singular value decomposition to project the high-dimensional response of a convolutional layer down to a low-dimensional mapping in image space. This operation is performed during the forward pass of the network, on a batch of input images. The resulting mapping can be displayed as a pixel-wise heatmap for each input image, where the contrasting features in the image batch are encoded as different colours. An overview of the method is shown in fig. 2 and described further below. For more details, see [27] or the open-source code ¹.

A feature map F_l is an $n_B \times n_{c,l} \times n_{x,l} \times n_{y,l}$ matrix containing the responses of layer l in a convolutional neural network (CNN), where n_B is the number of images passed through the layer (batch size), $n_{c,l}$ number of channels and $n_{x,l}, n_{y,l}$ is the spatial size of that layer.

¹<https://github.com/SINTEF/PFV>

We then find the principal feature responses by decomposing the feature map using singular value decomposition as

$$F' = USV^T, \quad (1)$$

where F' is a centered version of the feature map in the last layer reshaped into a column vector, S is a diagonal matrix containing the singular values, and U is the decomposition of F' into the space described by the eigenvectors V .

The principal components are then the columns of the following matrix, sorted according to descending singular value

$$F_{\text{PCA}} = US = [\mathbf{d}_1 \quad \dots \quad \mathbf{d}_r]. \quad (2)$$

For visualisation, we keep the N first principal components $\{\mathbf{d}_1, \dots, \mathbf{d}_N\}$ and denote the matrix with these columns D'_N . To increase the resolution of the final visualisation, this matrix is multiplied with an activation map A computed from a sum of feature maps from earlier layers,

$$V = A \circ D'_N, \quad (3)$$

where \circ is an element-wise product and D'_N has been reshaped and interpolated back to image size.

3.1.2. Colour mapping for PFV

To visualise V , the original method uses $N = 3$ principal components and maps them directly to red, green, and blue (RGB) image channels. While this is an intuitive mapping for features that lie close to one of the three axes, the colour mixing between the components in RGB is not as easy to interpret. When plotting the two first components in V , we have seen that they often form long clusters stretching in different directions from the origin. Thus, to more clearly visualise the feature space, we propose using the hue, saturation and value (HSV) colour space, which gives an intuitive representation in polar coordinates. The mapping is done in the following way: Let $\{d_1, d_2\}$ be the two first principal components of a pixel. For each pixel, compute the polar coordinates of the point (d_1, d_2) ,

$$\theta = \text{atan2}(d_2, d_1), \quad r = \sqrt{d_1^2 + d_2^2}. \quad (4)$$

where atan2 gives the angle to the x-axis in the interval $(-\pi, \pi]$. The angle is mapped directly to hue, while the squared amplitude is assigned to both saturation and value, to emphasise strong values:

$$h = \theta', \quad s = r'^2, \quad v = r'^2 \quad (5)$$

where θ' is θ normalised to $[0, 1]$ and r' is r clipped to $[0, 1]$.

When applying the PFV method, it is important to consider the composition of the input batch, as it shows the contrasting features of one batch at a time. For instance, to highlight the feature difference in three classes, the batch must consist of a balanced set of images from these classes. If the input batch consist of a single image, the visualisation would highlight the different features within that image instead.

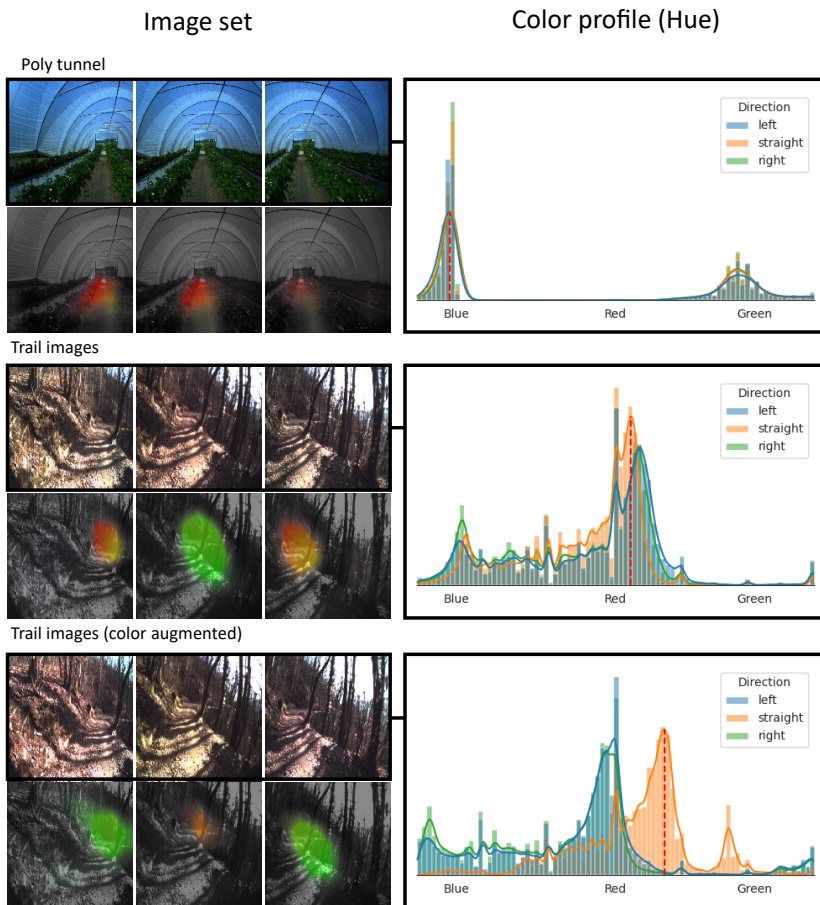


Figure 3: Comparison of features in trail and polytunnel images for a neural network trained on the IDSIA trail dataset. Pictures show example input images for left/straight/right classes, and the principal feature visualisation of the network’s response for strawberry polytunnel images, trail images, and colour-augmented trail images. Different colours indicate different directions in feature space for the trained network. (Note that the green feature only shows up for the ‘straight’ trail image, and not in the polytunnel.) Plots to the right show colour histograms for the different classes for the three image sets. Figure best viewed in colour.

3.2. Experimental evaluation

3.2.1. Debugging transfer learning

In [4], we experience issues with transfer learning. Specifically, when applying the VGG16 network trained only on trail-following data (IDSIA dataset from [3]) to similar data from a strawberry polytunnel, we observed that both ‘left’ and ‘right’ classes were classified correctly, but that there were no predictions for ‘straight’.

Here, we use the PFV visualisation method to detect differences in the feature response for trail and polytunnel images that may explain the failed transfer learning. The model is the VGG16 network trained on trail images from [4], and we compute the principal components based on the last layer before the decision head. The input batch consisted of left,

right, and straight example images from both trail and polytunnel dataset. Example image triplets from each dataset and their PFV visualisations are shown in fig. 3.

The visualisation shows strong localised responses for the trail images, particularly near the vanishing point of the trail. We see that the ‘straight’ image in the trail dataset has a strong green PFV component that is not present in the ‘left’ and ‘right’ images, which are encoded with orange PFV component. The green component does not appear in the ‘straight’ images from the strawberry polytunnel, which indicates that the feature extractor has learnt some feature that is specific to the ‘straight’ trail images. This is probably why the decision head fails to predict the ‘straight’ class for the polytunnel data. The difference in feature response for

areas with similar appearance indicates that there could be an underlying bias in the dataset.

3.2.2. Detecting dataset bias

To further investigate the possible bias in the dataset, the samples for the different classes of the trail dataset were inspected more closely. We found a slight difference in colour balance between the three classes, as shown in the right column of fig. 3. The peak in the histogram of the straight class in the trail dataset is shifted compared to the two other classes, and the image appears more red in tone, which is barely visible to the human eye.

If the red colour shift is the reason for the green PFV component and misclassification, then the visualisation and classification result should change if this shift is inverted. To test this, the hue was manipulated on a set of trail images from the same location. For the 'left' and 'right' images, the hue was reduced towards red, and for the 'straight' image, the hue was increased towards green. The resulting images, PFV responses, and colour profiles after manipulation are shown in the lower part of fig. 3. We see that after the colour manipulation, the colours of the PFV visualisations are inverted: 'left' and 'right' is now green, and 'straight' shows a faint orange. This manipulation also alters the classification result, which indicates that colour profile is one of the main features that the network uses to classify the direction.

3.3. Discussion

Applying PFV on the trail-to-agri transfer learning problem revealed a bias in the original trail dataset to which the neural network was overfitting. We emphasise that the source of the error was not known to the authors a priori, and the feature visualisation was an essential tool for identifying the problem. To detect the difference between the three classes in this case, the visualisation had to highlight the *difference* between the features, not only which areas are important for the classification, as for other visualisation methods [23, 24].

Now that we have detected the bias, how can this be fixed or avoided in a practical application? The origin of the colour bias could be due to the recording setup with three different cameras, which probably had different white balances. This is not an issue for our polytunnel data, as all viewing angles are recorded simultaneously with a single wide-FOV camera. The colour bias discovered by feature visualisation can explain why fine-tuning of the convolutional layers was necessary to learn the straight class. Applying colour augmentation during the training process, which is fairly common technique in deep learning, would also help in this case.

Although colour augmentation or a different recording setup could resolve this specific problem, there could always be other biases or issues with the data that are more tricky to fix or discover. This example illustrates the importance of making the training process more explainable and *controllable*, to ensure that the network learns reasonable features to base its decision on. This insight is the main motivation for our hybrid network architecture that is presented in the next section.

4. Hybrid learning

The insights from the previous section and previous work on the end-to-end learning for row-following motivated the development of a more explainable and robust network architecture and supervision approach addressing these issues. Additionally, the good results with automatic segmentation labels in [20] makes it possible to train a segmentation output in a data-efficient manner.

In this paper we present a novel *hybrid* network architecture and supervision approach for row following. The overall idea is to enforce a low-resolution segmentation representation and train both this and the angle output simultaneously in a supervised manner with automatically generated labels. We evaluate our approach on a dataset from a strawberry field, with particular attention to robustness to limited data and possible dataset biases.

4.1. Method

4.1.1. Hybrid network architecture

The overall network architecture consist of three modules, as illustrated in fig. 4. The input is an image with RGB channels, which is transformed to a low-resolution feature map with the *segmentation encoder*. There are two different outputs that can be trained in tandem: a crop segmentation mask that is produced by the *segmentation decoder*, which is fed to a *decision head* that predicts the direction of the crop row. The design of these modules is illustrated in the lower part of fig. 5 and discussed in more detail below.

Segmentation encoder For the segmentation encoder, we chose the well-known SegNet architecture [28], as implemented in the Keras Image Segmentation Library². It is a simple architecture without skip connections, and consists mainly of four convolution blocks, where each block consists of zero-padding of the input, a convolutional layer with filter size 3×3 , batch normalisation, ReLU activation and a max-pooling layer with pool size 2×2 . The output of the encoder, or the *bottleneck*, has 256 channels and 22×22 resolution (ignoring the batch dimension).

Light-weight segmentation decoder Conventional segmentation architectures have many parameters that are used for learned upsampling. Inspection of the activations of the architecture used in [20], have indicated that the resolution in the bottleneck should be sufficient to estimate the row direction in the strawberry fields. In that case, we can skip the decoder entirely and save both parameters and inference time. Therefore, rather than using a full SegNet decoder with upsampling layers (indicated with a gray shadow in fig. 4), we added one single convolutional layer to reduce the number of channels from 256 to 3, followed by a softmax activation to produce a segmentation output, as shown in fig. 5.

Light-weight decision head To predict the the crop row direction, we need a decision head that can transform a feature representation to the final one-value output. In the VGG

²<https://github.com/divamgupta/image-segmentation-keras>

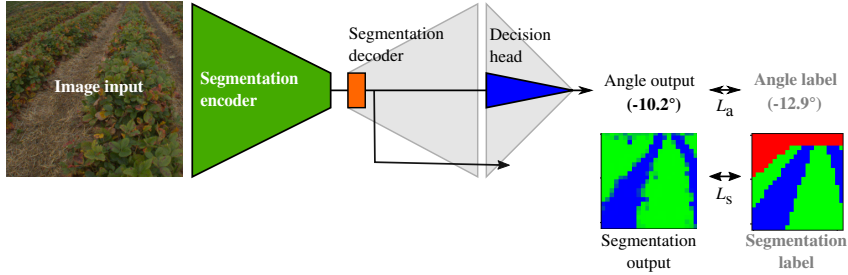


Figure 4: Hybrid network architecture with two output branches: segmentation and angle.

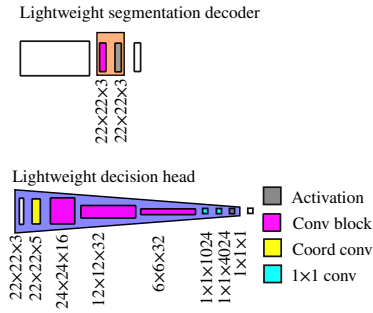


Figure 5: More detailed building blocks of the hybrid network architecture from fig. 4

architecture used for the end-to-end learning in [5], the decision head consists of two fully-connected layers, which takes a low-resolution feature map with a depth of 512 channels as input. The two main problems with this configuration are: 1) the loss of spatial information, and 2) the large capacity and associated risk of overfitting. To help the network understand spatial relationships in the image, we have added coordinate channels as suggested in [29], and use 1×1 convolutions instead of fully connected layers. To prevent overfitting and enforce a semantically meaningful representation, we use the 3-channel softmax segmentation map as input to the decision head. Three shallow convolutional blocks (with the stack of layers as in the encoder) are added before the 1×1 convolutions to reduce the resolution and extract geometrical features from the segmentation mask. The final activation is a softsign function $f(x) = \frac{x}{|x+1|}$, that limits the output to the interval $[-1, 1]$ and converges polynomially.

4.1.2. Supervision approach

Multi-task loss The two outputs of the hybrid network are trained in tandem with a dual loss function

$$L = \alpha L_a + \beta L_s \quad (6)$$

where α and β are weights that determine the contribution of the angular loss L_a and the segmentation loss L_s to the total training loss. For the angular loss, we use the stan-

dard Mean Squared Error (MSE). The segmentation loss is a class-weighted per-pixel cross-entropy which has the effect of ignoring the background class.

With the weighted loss function, the exact same architecture can be trained as a pure segmentation network, a pure end-to-end network, or a combination. In practice, we used two different configurations for our experiments: 1) *Pure end-to-end*, with $\beta = 0$ and $\alpha = 1$, and 2) *Segmentation pre-training*, with $\alpha = 0$ and $\beta = 1$ in the pre-training stage, then with a frozen segmentation part and $\beta = 0$ and $\alpha = 1$ to train the decision head and learn to extract angles from segmentation masks.

Automatic label projection The hybrid network architecture can be trained with hand-drawn segmentation labels, but it would then require more labelling effort than a pure end-to-end approach. Therefore, we use the automated label projection from [20]. In this work, we assume that the robot is driving straight and centred along the crop row, such that there is no need for additional alignment based on GNSS positions. Instead, we combine it with virtual camera view extraction to generate masks with different view angles in software. The full label generation pipeline is illustrated in fig. 6. We refer to [20, 4] and the open-source code³ for more details about this method.

4.1.3. Experimental setup

Data recording The dataset used in this work was recorded in a strawberry field over two years during different seasons, with an RGB camera mounted on an agri-robot platform.

Specifically, we used a Basler Ace camera and a Sunex fisheye lens with 190° FOV, mounted at approximately 1 m height and 22.5° downward tilt (extrinsics were measured individually for different versions of the setup). For intrinsic camera calibration, we used the OcamCalib Toolbox [30].

The camera was mounted on the Thorvald [31] robot platform from Saga Robotics⁴ as shown in fig. 7. The robot platform has an adjustable wheel spacing, that was set to 1.25 m to match the row width. The robot was also equipped with several other sensors used for other studies.

The robot was steered manually as straight and centred as

³<https://github.com/SINTEF-Computer-Vision/autolabel>

⁴www.sagarobotics.com

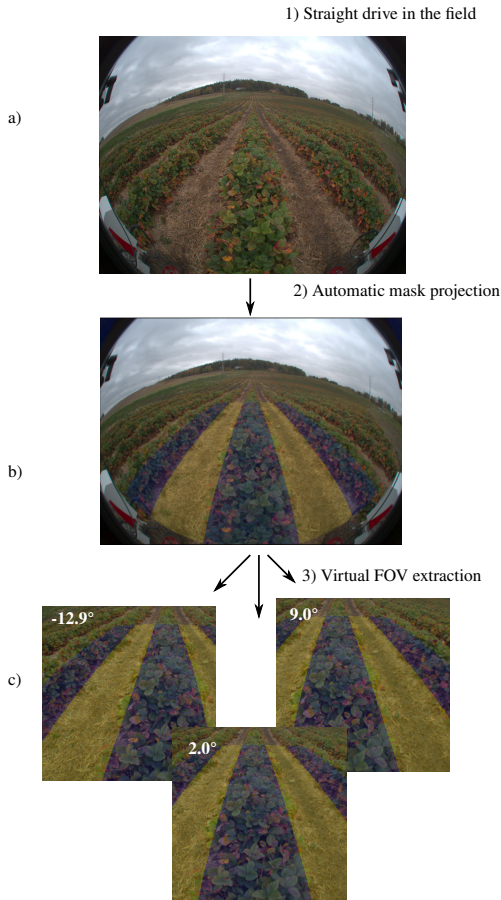


Figure 6: Automatic labelling of crop row data. a) Full-FOV image from fisheye camera. b) Full-FOV segmentation mask generated by projection c) Images and masks from different view angles, extracted virtually in software.

possible along the crop row during data capture, at a speed of approximately 1 m/s to avoid abrupt motions. The onboard computer is running ROS⁵ and image frames were recorded to rosbag at a rate of approximately 5 frames per second along with localisation- and other meta-data at higher rates.

Data collection was performed in a strawberry field in Norway with slightly hilly and curved rows. There are sections of different strawberry types with different plant ages (strawberry plants are perennial), which gives a large variety in appearance. The lane spacing is fixed, but the crop width varies, and was measured manually for each row of the recordings.

Strawberry field dataset For the hybrid learning dataset, we used wide-FOV images from a straight drive, as for the

⁵<https://www.ros.org/>



Figure 7: Data recording setup with robot in the field. The fisheye camera used for the recordings in this work is highlighted with a circle.

procedure from [5], and generated the training labels for segmentation and heading angle offline.

The images in the dataset are from three different rows, recorded in both directions. Some example images are shown in fig. 8. The image stream was temporally downsampled, such that there is little overlap between consecutive frames.

The automatic segmentation labels were projected to the full-FOV image as described in section 4.1.2 based on the camera calibration and extrinsics of the camera-to-robot transformation. We used two lane rectangles and three crop rectangles, which were projected up to 5 m ahead of the robot. All pixels that fell outside these rectangles were labelled as 'background' and assigned a value of 0, such that they would not contribute to the training loss, while 'crops' were assigned a value of 1 and 'lanes' a value of 2.

The angle labels were extracted with the virtual FOV processing explained above. For each image, we extract images and masks with a 60° FOV and three different yaw angles drawn from uniform distributions in the intervals -15 to 5, -5 to 5 and 5 to 15 degrees. All images were extracted with zero roll angle, and a pitch of -10° , corresponding to downward tilt of the camera.

After label generation and view extraction, the full dataset consisted of 2157 images. Of these, 438 (20%) were reserved for testing, by extracting continuous sections from each row, such that there was no visual overlap between training and test sets.

To investigate the effects of training with too little data, we also extracted a very small dataset with 43 images, drawn randomly from the original training set.

Artificial dataset bias To perform controlled experiments on robustness against dataset bias, we generated a dataset where the input images were altered with columns of blank pixels. The widths of the columns were proportional to the angle labels, and they were placed on either the right or left side of the image depending on the sign of the angle, as shown in the examples in the lower part of fig. 8. The labels (both angle and segmentation mask) remained unchanged. One would perhaps expect that the network would overfit by just learning to count the number of blank pixels, but as we shall see, this was not the case with the hybrid network.

The images in the biased dataset are a subset of the full dataset with 894 images for training and 222 for testing.

Training procedure The network was trained with the following hyperparameters: batch size 8, Adam optimiser with initial learning rate of 10^{-4} and learning rate decay, and early stopping when the validation error starts rising. All image input values was scaled to $[0, 1]$, angular label values to $[-1, 1]$.

As mentioned above, the hybrid network architecture was trained with two different configurations: 1) pure end-to-end and 2) segmentation-pretraining. In the first case, the whole network was trained in one go on angle labels only for 110 epochs. In the second case, the network was first trained with segmentation labels only for 200 epochs. Then, the encoder weights were frozen, and the decision head was trained with angle labels. In both cases, the input images were augmented by a series of operations: jitter in hue, saturation, contrast, and lighting, as well as gaussian blur. The number of augmentations active per image was randomly assigned, and the strength also varied. Note that there were no geometric augmentations applied, as this would ruin the angle labels, and not have any effect on the segmentation training.

The frameworks used in this implementation were Keras⁶ for high-level network building blocks, data loading and training/evaluation functionality, with Tensorflow⁷ as back-end. Tensorboard⁸ was used for monitoring the training process. Image augmentation during training was performed with the imgaug library [32].

4.2. Results

4.2.1. Dataset size

To compare the hybrid network with a pure end-to-end approach, we trained both configurations on the full strawberry field dataset, as well as the small subset, and measured the error in the predicted angles on the test set. The results in table 1 show that both achieve a low average error of around 1° , with slightly superior performance for the end-to-end approach. For the small dataset on the other hand, the hybrid network performs better. This is expected, as the segmentation mask has many labels per image (one for each pixel), and is therefore able to learn successfully from very few samples.

⁶<https://github.com/keras-team/keras>

⁷<https://www.tensorflow.org/>

⁸<https://github.com/tensorflow/tensorboard>

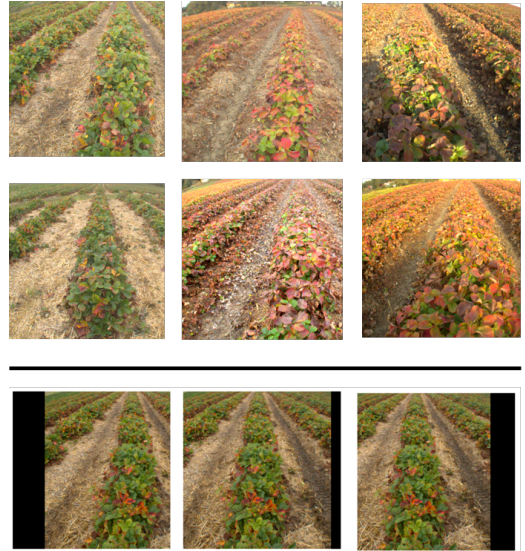


Figure 8: Example images from the strawberry field dataset. Upper two rows: images from different locations and seasons in the same field. Bottom row: Artificial dataset bias created with dead pixel columns of width proportional to the ground truth angle.

Table 1

Performance on full-size and small dataset: Mean absolute error (MAE) and standard deviation (std) of predicted angles for end-to-end and hybrid network.

	End-to-end		Hybrid	
	MAE[°]	Std	MAE[°]	Std
Full dataset	0.9	1.1	1.3	1.6
Small dataset	4.1	5.8	2.2	2.8

4.2.2. Explainability

In order to demonstrate how the segmentation output can be used to better understand failure cases, we show some example outputs for the networks trained on too little data (the small dataset in section 4.2.1), and compare with the same output for a network trained end-to-end without segmentation labels in fig. 9. The left example shows a sample with small angular error, and a good segmentation. For the second example with an angular error of 6.9° , we can see that there are errors in the segmentation, probably due to strong shadows and challenging lighting conditions. This indicates that the feature extractor needs more varied data or augmentation to be able to generalise. The second image however, shows a near perfect segmentation although the angular error is 7° . This indicates a failure in the decision head, probably due to insufficient variation in field geometry and view angle in the dataset. Due to the separation of feature extraction and angle estimation in the hybrid network, it is possible to train the decision head separately on augmented masks to increase the amount of training data. Thus, it is possible to fix

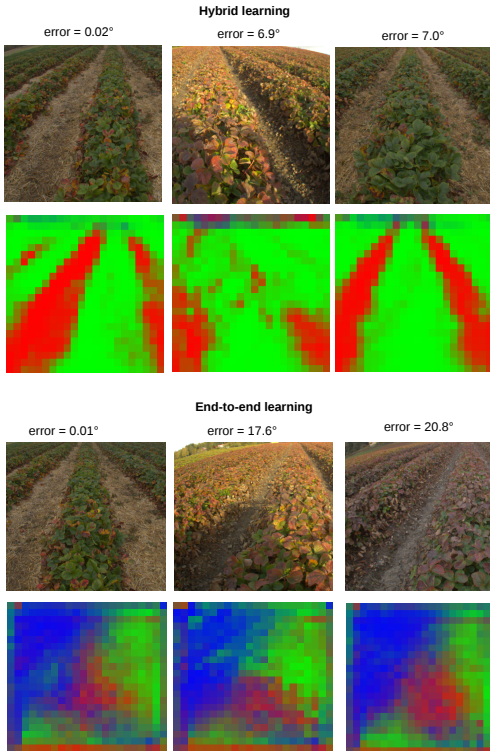


Figure 9: Intermediate outputs for hybrid and end-to-end networks. The models were trained with too little data, and the examples show the output for one good (first from left) and two bad samples in the test set for each of the networks.

the issue in this example without adding any new images to the training set.

The end-to-end network shows a much larger angular error for the worst examples than the hybrid network, but the intermediate output does not show much difference between the good and bad samples at all. Therefore, we can not tell whether these samples fail due to appearance or geometry, and one would just have to get more data and hope for the best.

4.2.3. Dataset bias

To evaluate the different network architectures' robustness to dataset bias, we trained both configurations on the artificially biased dataset with blank pixels, and validated on input images without the artefacts. The resulting test errors for the predicted angle are reported in table 2. We see that the hybrid network has a much lower error than the pure end-to-end network, (for reference, random guessing gives a MAE of 7.5°). To verify that the end-to-end network is actually overfitting the artefact introduced in the image, we compare the training loss (images with artefacts) and vali-

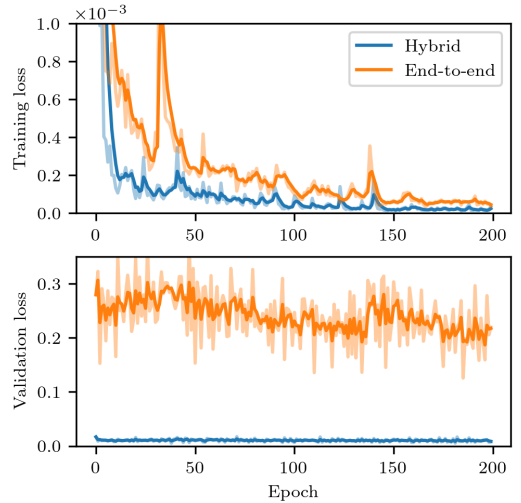


Figure 10: Training and validation (angular) loss as a function of epoch for training on a biased dataset with the end-to-end network (orange) and hybrid network (blue). Note that the end-to-end network has a very high validation loss that is not declining.

Table 2

Performance after training on a biased dataset: Mean absolute error (MAE) and standard deviation (std) of predicted angles, for both networks trained on biased data.

Learning approach	MAE[°]	Std [°]
End-to-end	5.6	6.4
Hybrid	1.2	1.4

dation loss (images without artefacts) for the two different networks in fig. 10. We see that while the training loss is decaying steadily for both approaches, the end-to-end network has a very high validation loss that does not converge at all. Thus, the end-to-end network is strongly overfitting, whilst the hybrid network does not.

4.3. Discussion

The experiments show very good performance on the strawberry field dataset, with small average error in estimated angle (around 1°) for the proposed hybrid architecture. In the ideal situation with sufficient data and little difference between training and test data, the straight-forward end-to-end supervision approach works as well as or slightly better than the hybrid approach. This is probably because the end-to-end learning benefits more from more data and longer training time than the segmentation network. The segmentation network quickly learns a mask that is more accurate than the ground truth due to the inaccurate automatic labels, and therefore cannot reduce the loss any further even though more data is added. Additionally, the limited resolution in the segmentation masks puts a limit to how accurate the predicted angles can be. Thus, this is a trade-off between speed

and accuracy.

We see that the hybrid approach performs much better than the end-to-end approach in less ideal situations, especially with bias in the dataset. Enforcing a segmentation representation in the bottleneck forces the decision head to base its decision on semantically meaningful features, and therefore prevented overfitting to the added blank pixels. In this case, the bias was a very specific synthetic artefact, but this strategy is expected to work just as well on other local artefacts such as buildings in the horizon or structures in the polytunnels.

Separating the feature extraction from the angle prediction makes the network architecture itself more explainable, but it also makes it easier to make small adjustments to the system. For instance, in the case of good segmentation but bad angle prediction, as identified in fig. 9, it would be easy to fix this by generating masks with more varied geometry through virtual view extraction, without recording any new images. In the case with bad segmentation, one could start by adding more image augmentation, but it would perhaps require new recordings as well. The same technique can be used in case of large seasonal changes or when adapting a pre-trained network to a new application, to inspect how well the trained model will transfer to the new task.

5. Robot field trials

To test the performance of the neural network in a real application, we performed a series of open-loop field trials. This was done by driving the robot manually in the same strawberry field as the previous data collection, and running the neural network for angle prediction on each image in the camera stream for the whole length of the crop row.

5.1. Setup

For the first test, we used the same setup and driving pattern (straight along the row) as described in section 4.1.3, but we additionally varied the viewing direction of the virtual camera view that was used to extract image data from the raw images by a sinusoidal pattern as the robot moved along the row – emulating a slalom driving pattern. For the second test, the robot was manually driven in an actual slalom pattern turning from side to side along the row. The main difference from the first test scenario is that the manual slalom drive introduces a varying lateral offset from the crop row centre line in addition to yaw angle, due to the turning motion of the vehicle. The ground truth heading angle was measured in the same manner as in [20] using position data from a dual-antenna GNSS receiver. Instead of extracting virtual FOVs from the fisheye camera, colour images from an Intel Realsense were used. The Realsense camera was mounted above the fisheye camera, as shown in fig. 7, and has slightly different extrinsics and FOV than the images the network was trained on. Both the Realsense camera and the GNSS antennas are shown in the robot setup in fig. 7. For both tests, we used the hybrid network architecture from section 4.1.1 trained on the full dataset with both segmentation

and angle labels. The inference of the neural network ran on a laptop with a GeForce GTX 1070 Mobile GPU.

5.2. Evaluation

The real-world performance of our approach during row following was evaluated qualitatively by plotting the predicted angle for each camera frame against ground truth from GNSS. The results from the two tests are presented in fig. 11 and fig. 12 respectively. Additionally, we have animated sequences of images from each test showing the predicted angle as an arrow overlaid on the camera stream together with the estimated segmentation.

From the first test results (fig. 11), we see that the predicted heading angle follows the ground truth very well throughout the entire length of the row. This result shows both that the segmentation is robust under real-world conditions and that the estimation of heading angle should be sufficiently accurate for steering control.

From the second test results (fig. 12), where the robot was driven in an approximately slalom path, we see that the form of the estimated heading angle matches the ground truth, but that after the initial response to each change in turning direction the estimated heading angle continues to increase in magnitude throughout each turn. We hypothesised that the network was in fact conflating the increasing lateral displacement of the vehicle during each turn with the heading angle, as these two motions produce similar image effects.

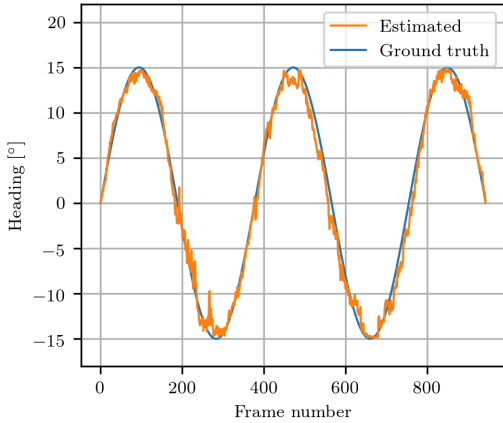
To test this hypothesis we computed the expected apparent angular deviation experienced by the robot by considering a point along the midline of the crop row 2 m in front of the robot’s position. The apparent angular deviation, Δ , of this point due to the robot’s heading angle deviation, θ , and lateral offset, d_L , from the crop row midline is given by

$$\Delta = \theta + \text{atan2}(d_L, 2). \quad (7)$$

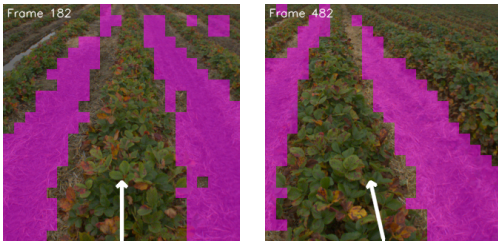
The apparent angular deviation is plotted alongside estimated and ground-truth heading in fig. 12 and it can be seen that it matches well with the heading angle estimated by our network, confirming our hypothesis. Here we have simply determined the distance to the test point (2 m) empirically, but the goodness of the fit indicates that this is in fact what is being estimated by the network.

That our network is estimating apparent angular deviation to a point along the crop row in front of the robot instead of the instantaneous angular deviation is not necessarily detrimental for robot control, as this is in fact analogous to the effect of the integral component in a PID controller – it will correct for any accumulated lateral drift during closed-loop control. Ideally, however, the weighting between the contributions of the angular deviation (P) and lateral offset (I) to the control of the vehicle would be tunable independently. Including estimated lateral offset as an independent network output was not within the scope of this work, but is a topic for continuing research.

The inference time of the neural network with one image at a time was 9 ms on the laptop, which should be sufficient for closed-loop control with this robot. For more real-time demanding applications, faster execution times could



(a)



(b)

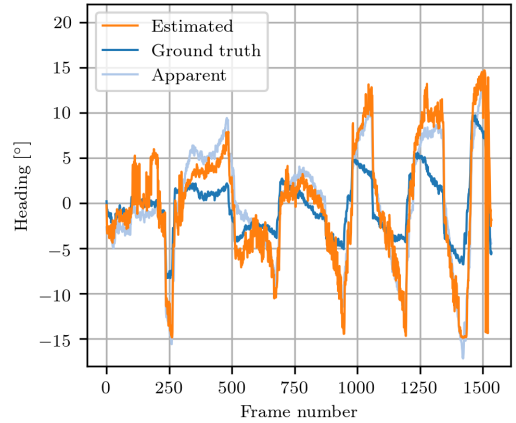
(c)

Figure 11: Open-loop results for a straight path with emulated sinusoidal heading angle variation (see section 5.1 for details). a) Estimated (yellow) vs. ground truth (blue) heading angle deviation. b) and c) Example images frames showing estimated angle and lane segmentation. See video (<https://youtu.be/kKmP4x-j5P8>) for animation.

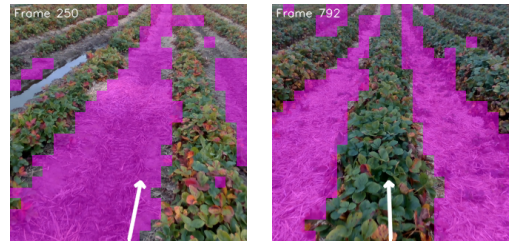
be achieved by reducing the number of parameters in the segmentation encoder.

6. Conclusion

In this work, we have focused on solutions to common pitfalls encountered when applying deep learning approaches in practical settings, with visual row-following for an agri-robot as our specific use case. Our experiments with feature visualisation enabled us to successfully identify and resolve a failure we faced during transfer learning, which turned out to be partly due to a bias in a public dataset. This motivated the development of a hybrid learning approach with an additional segmentation output that enforces semantically meaningful representations. In our experiments, we find that a published end-to-end approach and our new hybrid approach both show similar performance on our strawberry field dataset. But more importantly, we see that our proposed hybrid network outperforms the end-to-end approach in less ideal situations – its performance is almost unaffected when



(a)



(b)

(c)

Figure 12: Open-loop results for a real slalom path. a) Estimated (orange) vs. ground truth (blue) heading angle deviation. Also shown is the apparent angular deviation (light blue) from eq. (7). b) and c) Example images frames showing estimated angle and lane segmentation. See video (<https://youtu.be/XIw4MbgNdoI>) for animation.

training with dataset bias, whilst the performance of the end-to-end is significantly degraded. Additionally, when training with a very small dataset, our hybrid network is less affected than the end-to-end network, and the failed samples can be inspected by plotting the segmentation output. Finally, we have conducted open-loop field trials with an agri-robot that demonstrate good performance of our proposed approach in a realistic setting, and we believe our solution will enable autonomous onboard guidance with our agri-robot platform.

We conclude that our hybrid learning approach is an explainable, robust, and data-efficient solution that can help avoid the pitfalls otherwise encountered when training neural networks in practical applications, such as crop row-following.

Acknowledgement

The authors would like to thank Per Fredrik Saxebøl and Simen Myhre for access to their strawberry farms, as well as Vignesh Raja Ponnambalam and Jon Glenn Gjevestad for help with field-work and data collection.

References

- [1] Y.-K. Lin, S.-F. Chen, Development of navigation system for tea field machine using semantic segmentation, *IFAC-PapersOnLine* 52 (2019) 108–113.
- [2] V. R. Ponnambalam, M. Bakken, R. J. Moore, J. G. O. Gjevestad, P. J. From, Autonomous crop row guidance using adaptive multi-roi in strawberry fields, *Sensors (Switzerland)* 20 (2020) 1–17. URL: www.mdpi.com/journal/sensors. doi:10.3390/s20185249.
- [3] A. Giusti, J. Guzzi, D. C. Cire, F.-I. He, J. P. Rodriguez, F. Fontana, M. Fässler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, L. M. Gambardella, A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots, *IEEE Robotics and Automation Letters* PP (2015) 1–1. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7358076>. doi:10.1109/LRA.2015.2509024.
- [4] M. Bakken, R. Moore, P. From, End-to-end Learning for Autonomous Navigation for Agricultural Robots, in: *ICRA 2018 Workshop on Robotic Vision and Action in Agriculture*, 2018. URL: <https://research.qut.edu.au/future-farming/wp-content/uploads/sites/3/2018/06/End-to-end-Learning-for-Autonomous-Navigation-for-Agricultural-Robots.pdf>.
- [5] M. Bakken, R. J. D. Moore, M. Bakken, End-to-end Learning for Autonomous Crop Row-following (2019). doi:10.1016/j.ifacol.2019.12.505.
- [6] D. M. Woebbecke, G. E. Meyer, K. V. Bargaen, D. A. Mortensen, Color indices for weed identification under various soil, residue, and lighting conditions, *Transactions of the American Society of Agricultural Engineers* 38 (1995) 259–269. URL: <https://pennstate.pure.elsevier.com/en/publications/color-indices-for-weed-identification-under-various-soil-residue->
- [7] J. A. Marchant, R. Brivot, Real-Time Tracking of Plant Rows Using a Hough Transform, *Real-Time Imaging* 1 (1995) 363–371. doi:10.1006/rtim.1995.1036.
- [8] B. Åstrand, A. J. Baerveldt, A vision based row-following system for agricultural field machinery, *Mechatronics* 15 (2005) 251–269. doi:10.1016/j.mechatronics.2004.05.005.
- [9] X. Zhang, X. Li, B. Zhang, J. Zhou, G. Tian, Y. Xiong, B. Gu, Automated robust crop-row detection in maize fields based on position clustering algorithm and shortest path method, *Computers and Electronics in Agriculture* 154 (2018) 165–175. URL: www.elsevier.com/locate/compag. doi:10.1016/j.compag.2018.09.014.
- [10] I. García-Santillán, J. Miguel Guerrero, M. Montalvo, G. Pajares, G. Pajares pajares, Curved and straight crop row detection by accumulation of green pixels from images in maize fields, *Precision Agriculture* 19 (2018) 18–41. URL: <https://doi.org/10.1007/s11119-016-9494-1>. doi:10.1007/s11119-016-9494-1.
- [11] A. Ahmadi, L. Nardi, N. Chebrolo, C. Stachniss, Visual Servoing-based Navigation for Monitoring Row-Crop Fields, in: *Proceedings - IEEE International Conference on Robotics and Automation*, 2020. doi:10.1109/ICRA40945.2020.9197114.
- [12] J. M. Guerrero, G. Pajares, M. Montalvo, J. Romeo, M. Guijarro, Support Vector Machines for crop/weeds identification in maize fields, *Expert Systems with Applications* 39 (2012) 11149–11155. doi:10.1016/j.eswa.2012.03.040.
- [13] M. Kise, Q. Zhang, F. Rovira Más, A stereovision-based crop row detection method for tractor-automated guidance, *Biosystems Engineering* 90 (2005) 357–367. doi:10.1016/j.biosystemseng.2004.12.008.
- [14] S. P. Adhikari, G. Kim, H. Kim, Deep neural network-based system for autonomous navigation in paddy field, *IEEE Access* 8 (2020) 71272–71278. doi:10.1109/ACCESS.2020.2987642.
- [15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes Dataset for Semantic Urban Scene Understanding, in: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, IEEE Computer Society, 2016, pp. 3213–3223. doi:10.1109/CVPR.2016.350.
- [16] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: Common objects in context, in: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8693 LNCS, Springer Verlag, 2014, pp. 740–755. URL: https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48. doi:10.1007/978-3-319-10602-1.
- [17] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, D. Scaramuzza, DroNet: Learning to Fly by Driving, *IEEE Robotics and Automation Letters* 3 (2018) 1088–1095. URL: <http://ieeexplore.ieee.org/document/8264734/>. doi:10.1109/LRA.2018.2795643.
- [18] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to End Learning for Self-Driving Cars (2016). URL: <http://arxiv.org/abs/1604.07316>.
- [19] A. Giusti, J. Guzzi, D. C. Cireşan, H. Fang-Lin, J. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, L. M. Gambardella, A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots, *Robotics and Automation Letters* 1 (2016) 661–667. doi:10.1109/LRA.2015.2509024.
- [20] Bakken, Marianne and Ponnambalam, Vignesh and Gjevestad, Jon Glenn Omholt and Moore, Richard and From, Pål Johan, Robot-supervised learning of crop row segmentation, in: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization (2016). URL: <http://arxiv.org/abs/1610.02391>. doi:10.1109/ICCV.2017.74.
- [22] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K. R. Müller, Unmasking Clever Hans predictors and assessing what machines really learn, *Nature Communications* 10 (2019) 1–8. URL: <http://dx.doi.org/10.1038/s41467-019-08987-4>. doi:10.1038/s41467-019-08987-4.
- [23] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, *PLoS ONE* 10 (2015) 1–46. doi:10.1371/journal.pone.0130140.
- [24] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings* (2014) 1–8.
- [25] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net, in: *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015.
- [26] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should i trust you?" Explaining the predictions of any classifier, in: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, 2016. doi:10.1145/2939672.2939778.
- [27] M. Bakken, J. Kvam, A. A. Stepanov, A. Berge, Principal Feature Visualisation in Convolutional Neural Networks, in: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12368 LNCS, Springer Science and Business Media Deutschland GmbH, 2020, pp. 18–31. URL: https://link.springer.com/chapter/10.1007/978-3-030-58592-1_2. doi:10.1007/978-3-030-58592-1.
- [28] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation (2015) 1–14. URL: <http://arxiv.org/abs/1511.00561>. doi:10.1109/TPAMI.2016.2644615.
- [29] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, J. Yosinski, An intriguing failing of convolutional neural networks and the CoordConv solution, *Advances in Neural Information Processing Systems 2018-December* (2018) 9605–9616.
- [30] D. Scaramuzza, A. Martinielli, R. Siegwart, A toolbox for easily calibrating omnidirectional cameras, in: *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 5695–5701. doi:10.1109/IROS.2006.282372.

- [31] L. Grimstad, P. J. From, Thorvald II - a Modular and Re-configurable Agricultural Robot, *IFAC-PapersOnLine* 50 (2017) 4588–4593. doi:10.1016/j.ifacol.2017.08.1005.
- [32] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, et al., *imgaug*, <https://github.com/aleju/imgaug>, 2020. Online; accessed 01-Feb-2020.

Appendices

APPENDIX A

SUPPLEMENTARY MATERIAL PAPER IV

Principal Feature Visualisation in Convolutional Neural Networks

Supplementary Material

Marianne Bakken^{1,2}[0000-0003-4958-8194], Johannes Kvam¹,
Alexey A. Stepanov¹, and Asbjørn Berge¹

¹ SINTEF Digital, Forskningsveien 1, 0373 Oslo, Norway
`marianne.bakken@sintef.no`
<https://www.sintef.no/en/>

² Norwegian University of Life Sciences (NMBU), 1432 Ås, Norway

This document contains supplementary material for the paper *Principal Feature Visualisation in Convolutional Neural Networks* published at ECCV 2020 (paper ID 4198). Please see the paper for details on experimental setup.

Our code is available at <https://github.com/SINTEF/PFV>.

1 PASCAL VOC examples

Same setup as in Section 5.2 of the paper.

1.1 Batches of 20 classes

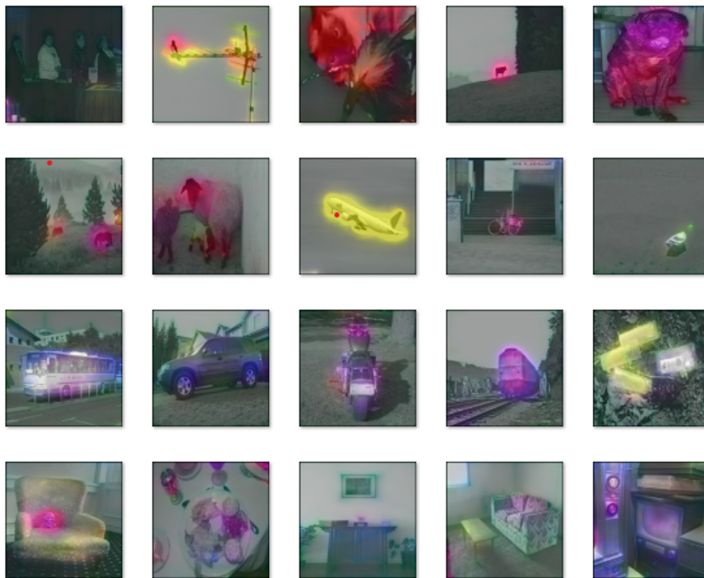


Fig. 1. PFV on batch of all 20 classes. Batch number 1.

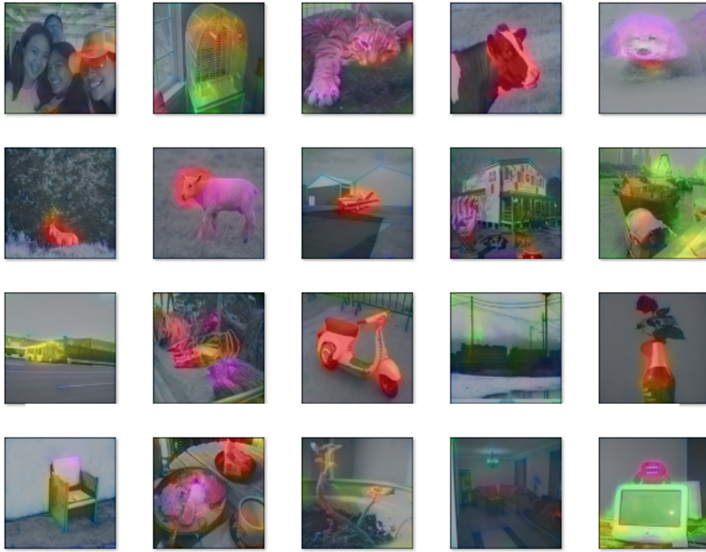


Fig. 2. PFV on batch of all 20 classes. Batch number 2.

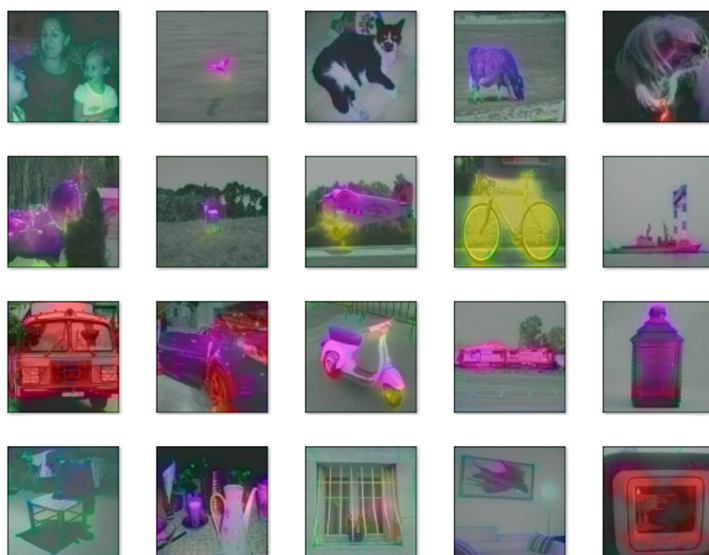


Fig. 3. PFV on batch of all 20 classes. Batch number 3.

1.2 Batches of 3 classes

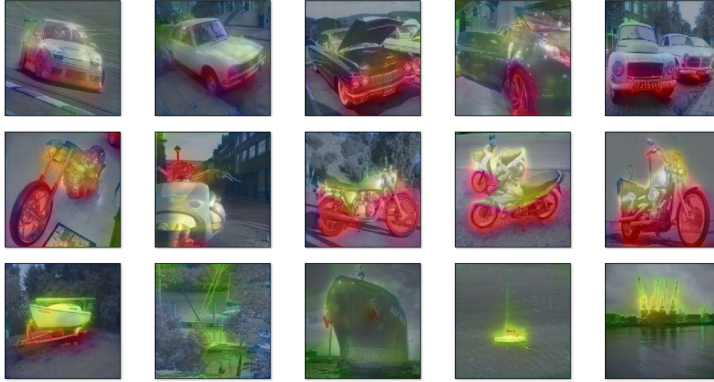


Fig. 4. PFV on batch of 3 classes: Motorbike, boat and car



Fig. 5. PFV on batch of 3 classes: aeroplane, bird and sheep

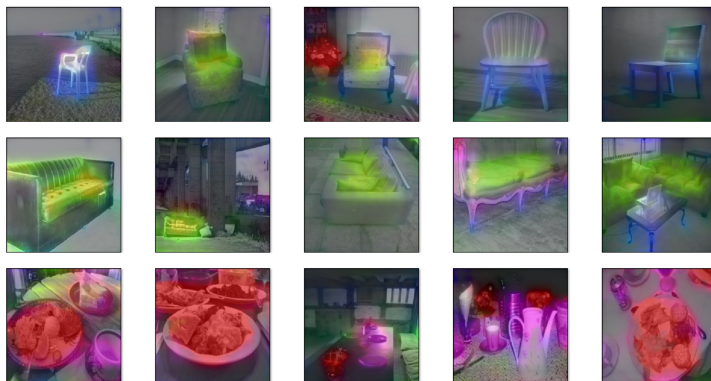


Fig. 6. PFV on batch of 3 classes: chair, sofa, diningtable



Fig. 7. PFV on batch of 3 classes: boat, motorbike, car

APPENDIX B

SUPPLEMENTARY MATERIAL PAPER III

B.1 Implementation of label projection for segmentation

The main idea behind the automatic labelling approach is to generate approximate segmentation labels by projecting the geometry of the scene into the image frame, based on the intrinsic camera model and the pose of the camera with respect to the scene (obtained from dual-antenna RTK GNSS). In this section, the implementation is explained in more detail than it was room for in the paper.

B.2 Label projection

The most general form of the virtual map is a set of separate regions with semantic labels given in a static reference frame F_{map} . Such a map can for instance be generated by combining prior knowledge about the structure of the scene with measured GNSS positions.

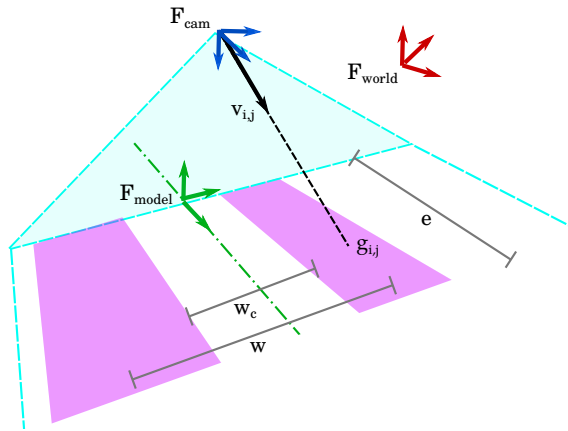


Figure B.1: Illustration of the label projection principle: A camera pixel is represented as a vector $v(i, j)$ in camera frame, projected to ground point $g_{i,j}$ in map frame F_{map} , and assigned a label based on which map region it hits in a map. For the special case of the *field map*, the regions are adjacent rectangles specified by the lane spacing w and the crop width w_c .

B. Supplementary material Paper III

The projection from the camera to the virtual map is performed in the following way. For each pixel (i, j) in the camera image, we compute the corresponding vector $v_{i,j}$ in the camera frustum using an intrinsic model of the camera obtained through calibration. $v_{i,j}$ is transformed from the camera coordinate frame F_{cam} to the map coordinate frame F_{map} , to find the ground point $g_{i,j}$ where $v_{i,j}$ intersects the map surface.

The virtual map in our implementation is a set of N convex polygons $\{P_n\}$ in the plane $z = 0$ in the F_{map} coordinate frame. Each polygon is associated with a label l_n . When assigning the label for a ground point $g_{i,j}$, we check if it is placed inside any polygon in $\{P_n\}$, by comparing the position of $g_{i,j}$ with each edge of the polygon using the orient2D geometric predicate. If it does not fall inside any polygon, the background label is assigned to that pixel.

B.3 Field map

To generate our virtual scene map, we must define the polygon boundaries for the crop regions. We do this by measuring the GNSS centreline of the driven crop row at approximately 0.25 m intervals and then generating a piece-wise rectangular polygon for the row using an assumed average crop row width – which will vary across fields and seasons. Polygons for neighbouring rows are similarly generated by using an assumed crop row spacing – which is typically fixed for a particular site because it corresponds to the track width of the tractor.

For the crop row segmentation case, a few approximations are done to simplify the generation of the virtual map and to obtain the transformation between the camera and the map coordinate frames.

We define a *field map* with a set of polygons as described above, more specifically a set of adjacent rectangles with fixed widths and limited extent representing straight crop rows and lanes in the field as illustrated in the figure above. To align the rectangles with the rows of the field, we use a set of crop row positions given in F_{world} , typically obtained with a GNSS receiver beforehand. For each position, a local map coordinate frame $F_{\text{map},n}$ is placed in the middle of the row, aligned with a piece-wise linear fit of the nearest points.

The geometry of the field map can be adjusted with the following parameters: extent, lane spacing w and crop width w_c as illustrated in the figure above. Typically, the lane spacing is fixed, as it corresponds to the wheel spacing of the tractor. The crop width will vary and has to be measured separately for different fields and seasons. The extent of the mask in the y direction can be adjusted depending on the field of view of the camera or how far ahead the linear approximation will hold.

The projection from pixel to ground point relies on accurate transformation between F_{cam} and $F_{\text{map},n}$. Typically using a GNSS receiver mounted on the robot platform during recording, we obtain accurate position and heading of F_{cam} in world frame F_{world} . It is then transformed to the nearest map frame $F_{\text{map},n}$, to compute the projection as described above.

APPENDIX C

VIDEO MATERIAL

Videos submitted as supplementary material to papers:

C.1 Paper III

The video shows driving pattern during recording and field trials, and animations of the generated labels and final segmentation results as the robot drive along the row.
<https://www.youtube.com/watch?v=CkI3bfmEHMY>

C.2 Paper V

The videos show animations of the predicted heading angle and segmentation masks from the field trials. The first shows a drive with emulated heading angle, the second shows a real slalom drive.
<https://youtu.be/kKmP4x-j5P8>
<https://youtu.be/XIw4MbgNdoI>

ISBN: 978-82-575-1849-3

ISSN: 1894-6402



Norwegian University
of Life Sciences

Postboks 5003
NO-1432 Ås, Norway
+47 67 23 00 00
www.nmbu.no