Plant Pest Detection via Deep-Learning Models

Ing. Dušan Šuľan

Master thesis
2021

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Ing. Dušan Šuľan**
Osobní číslo:	**A19728**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Detekce rostlinných škůdců pomocí modelů hlubokého učení**
Téma práce anglicky:	**Plant Pest Detection via Deep-learning Models**

Zásady pro vypracování

1. Vypracujte literární rešerši zabývající se modely pro detekci objektů v obraze.
2. Seznamte se s možnostmi automatické diagnostiky zdraví rostlin se zaměřením na detekci škůdců.
3. Vyberte několik modelů hlubokého učení vhodných pro detekci škůdců v hydroponickém skleníku.
4. Natrénujte vybrané modely na unikátních datech pořízených v hydroponickém skleníku.
5. Proveďte kvantitativní a kvalitativní srovnání výsledků realizovaných modelů.

Forma zpracování diplomové práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. GOODFELLOW, Ian, BENGIO, Yoshua, & COURVILLE, Aaron (2016). Machine learning basics. Deep learning [online]. MIT Press. 2016. Dostupné také z: https://www.deeplearningbook.org/.
2. RAMSUNDAR, Bharath a Reza Bosagh ZADEH. TensorFlow for deep learning: from linear regression to reinforcement learning [online]. Beijing: O'Reilly Media. 2018. ISBN 9781491980422. Dostupné také z: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1720096
3. ROSEBROCK, Adrian. Starter Bundle. In: Deep Learning for Computer Vision with Python. PyimageSearch.com. 2017.
4. ROSEBROCK, Adrian. Practitioner Bundle. In: Deep Learning for Computer Vision with Python. PyimageSearch.com. 2017.
5. LIU, L. et al. PestNet : an end-to-end deep learning approach for large-scale multi-class pest detection and classification. IEEE Access. 2019. 7. pp. 45301-45312.
6. RASCHKA, Sebastian a Vahid MIRJALILI. Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow . Second edition. Birmingham: Packt, 2017, xviii, 595 s. ISBN 978-1-78712-593-3.
7. CHOLLET, François. Deep learning v jazyku Python: knihovny Keras, Tensorflow . Praha: Grada Publishing, 2019, 328 s. Knihovna programátora. ISBN 978-80-247-3100-1.

Vedoucí diplomové práce: **doc. Ing. Zuzana Komínková Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Konzultant diplomové práce: **Ing. Alžběta Turečková**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **15. ledna 2021**
Termín odevzdání diplomové práce: **17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 16.5.2021

Dušan Šuľan v.r.
podpis diplomanta

ABSTRAKT

Tato diplomová práce řeší detekci škůdců pomocí modelů hlubokého učení. Cílem bylo natrénovat sítě hlubokého učení pro přesnou a rychlou detekci škůdců. Vyhodnocení probíhalo na základě ukazatelů Precision, Recall a rychlosti sítě. Výsledky ukazují, že nejrychlejší z testovaných síťí je Scaled YOLOv4 CSP Large a nejpřesnejší Faster R-CNN s Resnet152. Základní augmentace sítí zlepšili přesnost o 10%. V závěru je zformulováno zhodnocení quality datasetu, vyhodnocení modelů a zlepšení, které mohou pomoc zlepšit predikci škůdců v budoucnu.

Klíčová slova: hluboké učení, počítačové videní, object detection, detekce škůdců, Faster R-CNN, YOLO, Scaled -YOLO

ABSTRACT

This diploma thesis addresses the detection of pests using deep learning models. The aim was to train deep learning networks for accurate and fast pest detection. The evaluation was based on Precision, Recall, and network speed indicator. The results show that the fastest of the tested networks is Scaled YOLOv4 CSP Large and the most accurate is Faster R-CNN with Resnet152. Basic network augmentations improved accuracy by 10%. In the final part, an evaluation of the quality of the dataset, evaluation of models, and further possible improvements are formulated.

Keywords: deep learning, computer vision, object detection, pest detection, Faster R-CNN, YOLO, Scaled -YOLO

Acknowledgements

I would like to express my sincere gratitude to my research supervisor doc. Ing. Zuzana Komínková Oplatková, Ph.D. and my consultant Ing. Alžběta Turečková, for giving me the opportunity to do research and providing guidance throughout this research.

I am very much thankful to my girlfriend for her love, patient, understanding and support to complete this work. I am grateful to my parents for their love. The same for my brother and to all my family.

Motto

"*I never loss. I either win or learn."* Nelson Mandela

Declaration

I hereby declare that the print version of my Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

Obsah

INTRODUCTION **8**
THEORY **9**
1 MACHINE LEARNING **10**
1.1 DEEP LEARNING **10**
1.2 NEURAL NETWORKS **11**
1.2.1 NEURON 11
1.2.2 NEURAL NETWORKS LEARNING 12
1.3 LEARNING METHODS **13**
1.3.1 UNSUPERVISED LEARNING 14
1.3.2 SUPERVISED LEARNING 14
1.3.3 REINFORCEMENT LEARNING 14
1.4 GENERALIZATION **14**
2 COMPUTER VISION **16**
2.1 IMAGE CLASSIFICATION **16**
2.1.1 CONVOLUTION NEURAL NETWORK (CNN) 17
2.2 OBJECT DETECTION **18**
2.2.1 SLIDING WINDOW 19
2.2.2 IMAGE PYRAMID 19
2.2.3 PYRAMID OF FEATURES 20
2.2.4 IOU – INTERSECTION OVER UNION 21
2.2.5 NON-MAX SUPPRESSION 22
2.3 EVALUATION METRICS **23**
3 OBJECT DETECTORS **25**
3.1 ONE-STAGE DETECTORS **25**
3.1.1 OVERFEAT 25
3.1.2 LOCALIZATION AND DETECTION PIPELINE 26
THE RAW CLASSIFIER OUTPUTS A CLASS AND CONFIDENCE FOR EACH LOCATION. THE RESOLUTION OF THESE PREDICTIONS CAN BE INCREASED USING MULTI-SCALE CLASSIFICATION [18]. THE REGRESSION NETWORK THEN PREDICTS THE LOCATION OF THE OBJECT FOR EACH WINDOW OF A DIFFERENT SCALE AS SHOWN IN FIG. 21. 26
3.1.3 YOLO (YOU ONLY LOOK ONCE) 27
3.1.4 SCALED-YOLOV4 30
3.2 2 STAGE DETECTORS **33**

3.2.1 R-CNN 34
3.2.2 Fast R-CNN 35
3.2.3 Faster R-CNN 36
4 PEST DETECTION 38
4.1 PLANTVILLAGE PROJECT 38
4.2 PEST DETECTION AND IDENTIFICATION ON TOMATO PLANTS FOR AUTONOMOUS SCOUTING ROBOTS USING INTERNAL DATABASES 38
ANALYSIS 39
5 PREREQUISITES FOR TRAINING OF DETECTION SYSTEMS 40
5.1 HARDWARE USED FOR TRAINING 40
5.2 DATASET SPLIT 41
5.3 DATASET QUALITY 41
5.4 EVALUATION STRATEGY 43
5.4.1 Prediction confidence (score) 43
5.4.2 IOU 43
5.4.3 Precision, Recall and mean Average Precision 43
6 PEST DETECTION WITH FASTER R-CNN WITH FPN 44
6.1 GENERAL SETUP FOR ALL NETWORKS 44
6.1.1 Anchor generator 44
6.1.2 Weights training algorithm 44
6.1.3 Training setup 45
6.1.4 Augmentations 46
6.2 PEST DETECTION RESULTS 46
6.2.1 Accuracy versus speed 47
7 SCALED-YOLO V4 48
7.1 GENERAL SETUP FOR ALL NETWORKS 48
7.1.1 Training 48
7.1.2 Training setup 49
7.1.3 Augmentations 50
7.2 PEST DETECTION RESULTS 50
7.2.1 Accuracy versus speed 51
8 OVERALL OVERVIEW 52
8.1 BENCHMARK PAPER COMPARISON 52
9 DATASET IMPROVEMENTS 54
10 FUTURE WORK 55

10.1 SCALED-YOLO V4 55
10.2 FASTER R-CNN 55
10.3 DATASET QUALITY 55
CONCLUSION 56
BIBLIOGRAPHY 57
LIST OF ABBREVIATIONS 61
LIST OF FIGURES 62
SLIST OF TABLES 65

INTRODUCTION

The history of artificial intelligence has begun in 1943, when the first ideas and principles of artificial neural networks were founded. During 1970s, more advanced methods and algorithms emerged in the field, but most of them remained a theory. Advances in computer performance have made it possible to use this theoretical knowledge in practice. In the last years machine learning and especially deep learning made significant progress, which attracts a lot of students, researchers, and supporters from governments and business. This created a highly competitive market and the situation gave rise to deep learning algorithms, especially in the area of computer vision. Computer vision can be used in many fields like industry, traffic, and agriculture, which is also the topic of this work. Europe agriculture must consistently increase productivity in the spirit of Industry 4.0 to catch up with low prices from third world countries agriculture with cheap labor. Early automatic pest detection can help in the case of productivity and quality of plants.

The first chapters of the theoretical part of this work describe deep learning as part of artificial intelligence. Further chapters describe computer vision, object detection, main object detection systems, and possible evaluation metrics. Knowledge from the theoretical part is then leveraged in a practical one.

The practical part of the work uses recent deep learning object detection systems on a dataset with tomato pests and compare the results. Chapter 5: PREREQUISITES FOR TRAINING OF DETECTION SYSTEMS describes the software, dataset quality, and other important prerequisites for proper training of detection systems. The next chapters are about training and comparison of the best detection system in terms of accuracy and speed. The thesis uses the dataset of tomato plants with pests Bemisia tabaci (BT), Trialeurodes vaporariorum (WF), and their eggs.

I. THEORY

1 MACHINE LEARNING

Machine learning (ML) is a part of artificial intelligence (AI), which was created on the idea that systems can learn from data. Concretely ML are methods of data analysis that automates analytical model building, identify patterns and make decisions with minimal human intervention.

The first machine learning applications could work only with a small amount of data because of technology limitations. As computation power was increasing, more ML applications could be used in practice. Especially deep learning, as part of ML, made significant progress in recent years.

Machine learning models learn (adapt) iteratively from training data to produce reliable, repeatable decisions and results. ML is a great example of exponential technologies because it is not new, but one that has gained fresh momentum and radically increasing productivity in affected areas in recent years.

1.1 Deep learning

Deep learning is a branch of machine learning, which uses multilevel networks and therefore can solve nonlinear complex problems. One of the best definitions of deep learning was written by Yann LeCun, Yoshua Bengio, and Geoffrey Hinton:

"Deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but nonlinear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. The key aspect of deep learning is that these layers are not designed by human engineers: they are learned from data using a general-purpose learning procedure" [1]

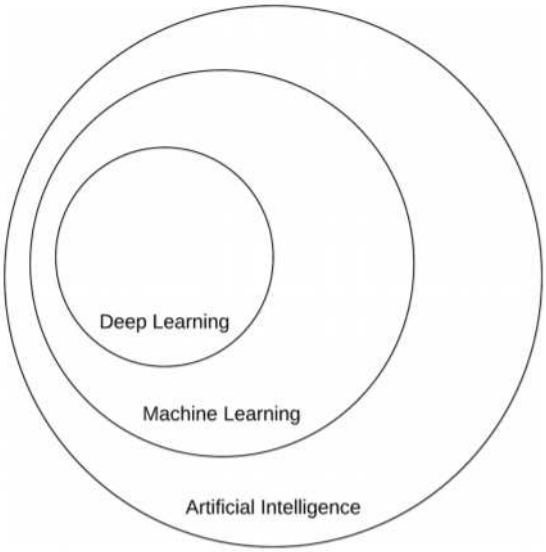

Figure 1: Deep learning and machine learning as parts of artificial intelligence [1]

1.2 Neural networks

Traditional programming approach tasks are in style: if parameter A is in a certain interval and another parameter B is in a different interval, then we diagnose two different classes. Thus, there is a clear, explainable relationship between inputs and outputs. This is not the case when using one of the interesting alternative programming techniques - artificial neural networks. Neural nets do not work algorithmic. They set up weights, tolerate deviations and changes.

1.2.1 Neuron

Artificial neural networks simulate human brain function with parallel distributed processing systems. The basic element of the natural and artificial neural system is a neuron or perceptron. An example of one neuron is shown in Fig. 2. Neurons are interconnected and signals are transferred to each other. Each neuron can have multiple inputs, but only one output (this output can be sent to more than one other neuron) [2].

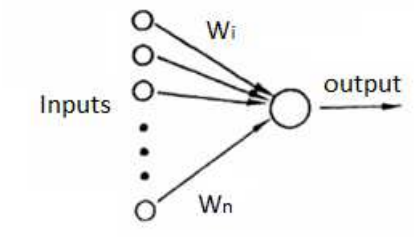

Figure 2: Simple neuron with input weights and one output [2]

Inputs for neurons can be either output from other neurons or information from outside. At the first, neuron receives the inputs. Then it multiplies their values by the weights. After that, it adds these products and if the result is greater than the specified threshold, the result is transformed by a predetermined transfer function and sent to the output. Perceptron can solve only linearly separable problems. Simple neuron behaves like unit step transfer function as shown below in the Fig. 3.

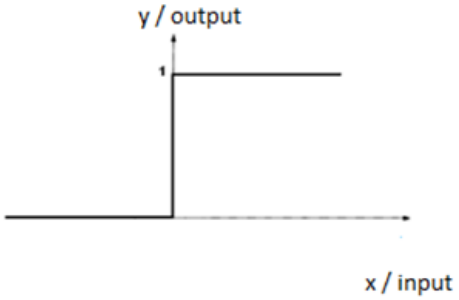

Figure 3: Step transfer function, which represents a simple neuron

1.2.2 Neural networks learning

The first neural network was created in 1958 by Frank Rosenblatt. The network used a simple learning algorithm based on fixed increments. Advanced networks were developed to solve also nonlinear complex problems [4]. An example of the complex network is shown in Fig. 4.

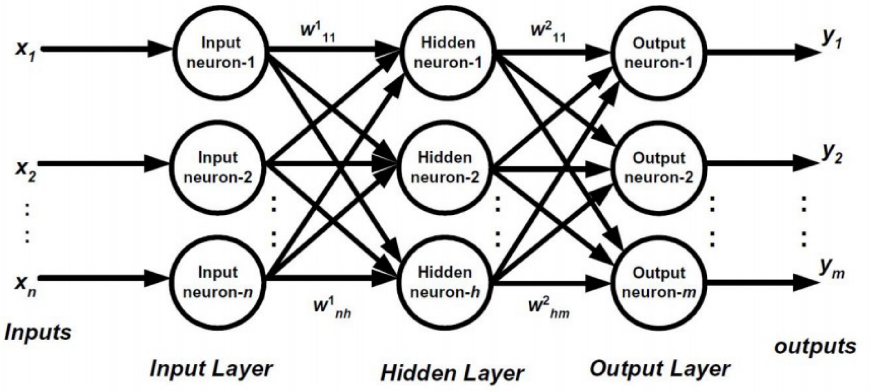

Figure 4: The example of a complex network with input, output and hidden layers [3]

The main problem of these networks was missing correctly propagated learning from the output through shallow to deep networks. Fortunately, the already existing chain rule was used in the Backpropagation algorithm. The Simple chain rule formula (1) is as follows:

$$\frac{dz}{dx} = \frac{dz}{dy} * \frac{dy}{dx} \tag{1}$$

The Backpropagation algorithm, which uses the chain rule, was originally introduced in the 1970s and fully appreciated in 1986 in a famous paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams. Calculation. The algorithm calculates weights of the final layer base on the output error and then the calculation continues till the first layer [6].

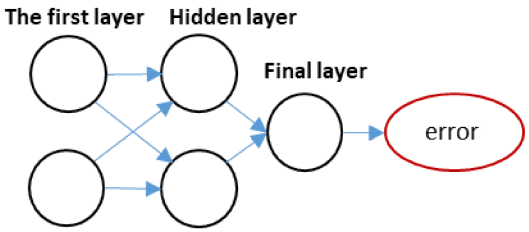

Figure 5: Schema of the neural network with output error

1.3 Learning methods

There are three main learning methods: unsupervised, supervised, and reinforcement learning. The main difference between them is the type of data that they need.

1.3.1 Unsupervised learning

Unsupervised learning needs only input data. These technics try to find structures in data or feature generation. Here are some examples of unsupervised learning methods: Dimensionality reduction, feature selection, and clustering. Clustering is an unsupervised technique where the goal is to find natural groups or clusters in a feature space and interpret the input data. Dimensionality reduction is commonly used where the goal is to reduce the number of variables under consideration.

1.3.2 Supervised learning

Supervised learning needs input data and output labels. These methods map inputs to outputs. The main applications are image classification, categorization, and time-series predictions.

1.3.3 Reinforcement learning

Reinforcement learning does not need input features and output labels. The system learns through interactions with the environment and received feedback. Feedback can be positive or negative base on the established rules.

1.4 Generalization

Generalization is the concept that humans and other animals use past learning in present situations of learning if the conditions in the situations are regarded as similar. In the case of machine learning, optimal generalization cause that our model can better predicts output labels for new data. It is necessary to consider whether it is necessary to obtain a trend or an exact approximation, which will not be too focused on specific data but will be robust to the given data. Examples of under-fitting, over-fitting, and appropriate fitting are depicted in Fig. 6:

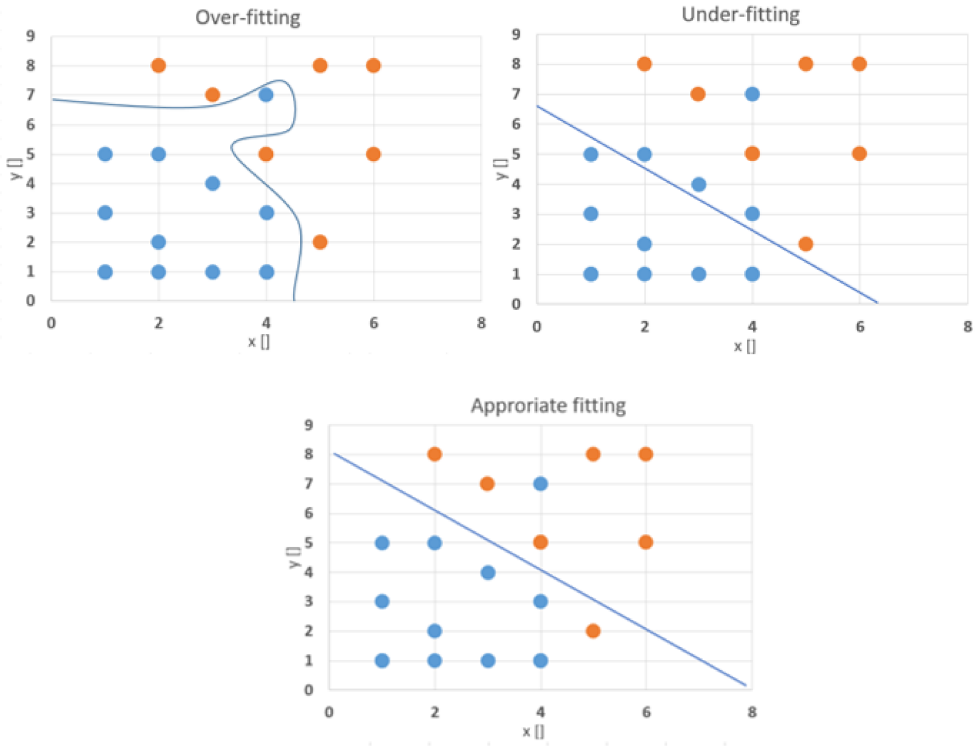

Figure 6: Examples of over-fitting, under-fitting, and appropriate fitting [7]

The generalization of models is changing through the learning process. If we would train our model too much, at one point generalization will start decreasing and the model will start to overfit because the model is too adapted to train data. Thus, training should be stopped before validation loss starts increasing as shown in Fig. 7. In addition, generalization can be also improved with the amount of training data and data augmentations.

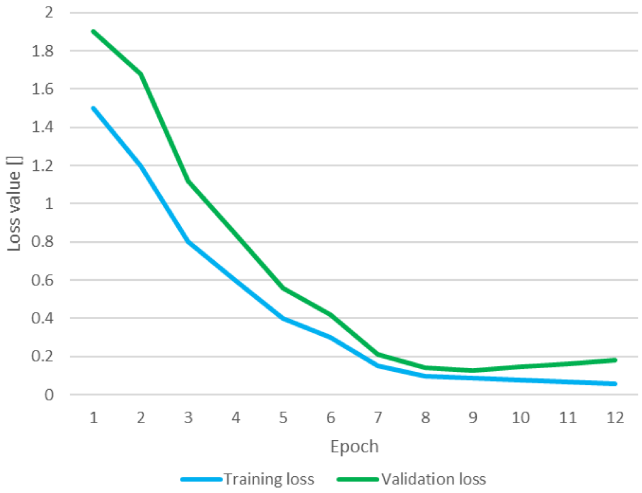

Figure 7: Validation and training loss through the training

2 COMPUTER VISION

Computers could store and visualize images, but we did not know how to get meanings from them for a long time. Firstly, there were used mainly image processing methods and manual feature extractions. Lately, scientists try to imitate the human brain based on the fact that people have a good understanding of what they see with no effort. All these techniques fall under computer vision. Thus, computer vision is a field of study that seeks to develop techniques to help computers to understand the content of images [8].

> *At an abstract level, the goal of computer vision problems is to use the observed image data to infer something about the world* [8]

Here are possible applications of computer vision:

- Image Classification - is the identification of the category to which an image belongs. An image can have only one class.
- Object Identification
- Object Recognition
- Object Verification
- Object Detection - is necessary when you want to locate an object in an image. One image can contain more objects.
- Object Landmark Detection
- Object Segmentation - Segmentation can locate an element on an image to the nearest pixel. For some cases, it is necessary to be extremely precise, as for the development of autonomous cars.

The main topic of the thesis is Object Detection, but the author of the thesis will discuss here also Image Classification as this is the precursor and base for Object Detection.

2.1 Image Classification

The task of Image Classification is the identification of one object on one image. It seems to be easy, but this is not the case. The main challenges are objects at different scales and shift-invariant. Shift invariant means, that the neural network should tolerate shifts of whole objects or parts of the objects. An example of shifted parts of objects is shown in Fig. 10. Shift invariant was solved by Convolution Neural Network (CNN).

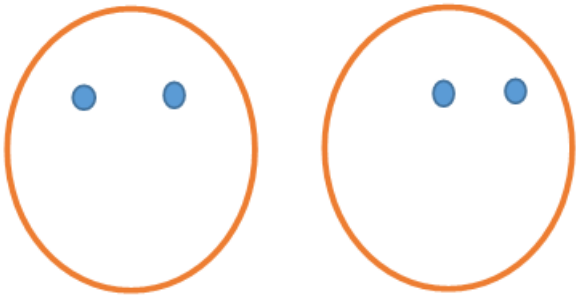

Figure 8: Two objects with shifted blue parts. Convolution Neural Network (CNN) can recognize these objects as objects of the same class.

2.1.1 Convolution Neural Network (CNN)

Convolution Neural Networks are the special hierarchical architecture of artificial neural networks focused on computer vision tasks. And they are part of deep learning and are inspired by visual parts of the human brain. The first layers of the CNN recognize small parts of the object (eyes and mouth in case of face recognition) and the last layers recognize if the face contains all small parts already recognized by the first layers (whole face in case of face recognition). The core of the CNN is the convolution layers and pooling layers. Fig. 9 shows both types [9], [10].

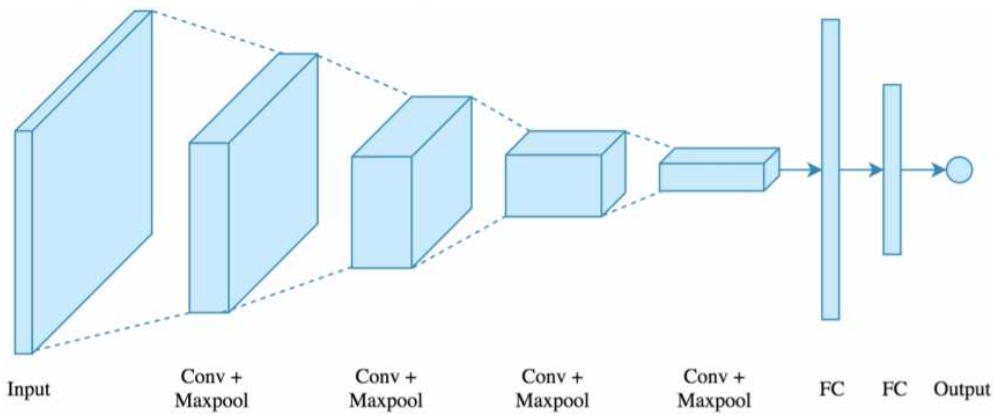

Figure 9: Example of CNN [9]

2.1.1.1 Convolution layers

Convolution layers work in a sliding window manner: they move small square windows (the most used are 3x3 kernels) in the image, loop through every possible position, and extract

useful features. Every single neuron in such a layer is a small square filter that performs a convolution operation [10]. The main point is that one convolutional kernel learns features of objects in all positions on images as it is looping through the whole image. A simple example of convolutional operation is shown in Fig. 10.

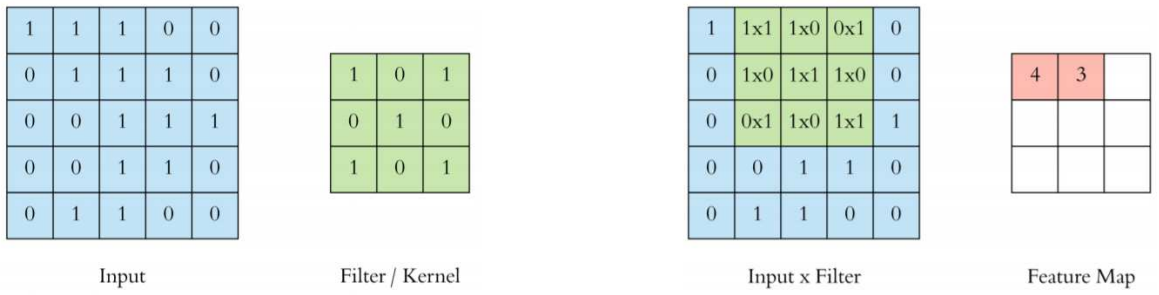

Figure 10: Input, CNN kernel, and output feature map [11]

2.1.1.2 Pooling layers

Pooling layers group neighbors in neural networks based on specific calculations. The most used is maximum (max-pooling layers) or average grouping (average pooling layers). The pooling layer is the reason for the shift-invariant character of CNN. It is because does not matter in what neighboring neuron is a maximum value (in case of max-pooling), the output of the pooling layer will be the same. An example of simple max-pooling layer is depicted in Fig. 11.

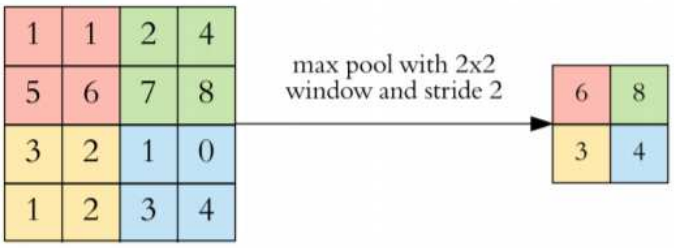

Figure 11: Input and output after max pooling with 2x2 window with stride 2 [11]

2.2 Object Detection

Object Detection is a more advanced method than image recognition. Because now we need to not only identify one object on one image, but we need to detect all objects with corresponding coordinates. Fortunately, networks already developed for image recognition can be used with some additions for Object Detection tasks.

2.2.1 Sliding window

The most intuitive approach how to solve object detection problems is to use the sliding window method. The method must be combined with image classification systems like AlexNet, which will recognize an object in every window. The sliding window is a rectangle of fixed width and height that slides across an image. Therefore, the main parameters are height, width, and step size. Step size is the number of pixels that we are skipping for every step while looping through an image. An increase in step size decreases our computation time but can also decrease the accuracy of our object classification and vice versa. Starting and ending position of sliding window is shown in Fig. 12 below [12].

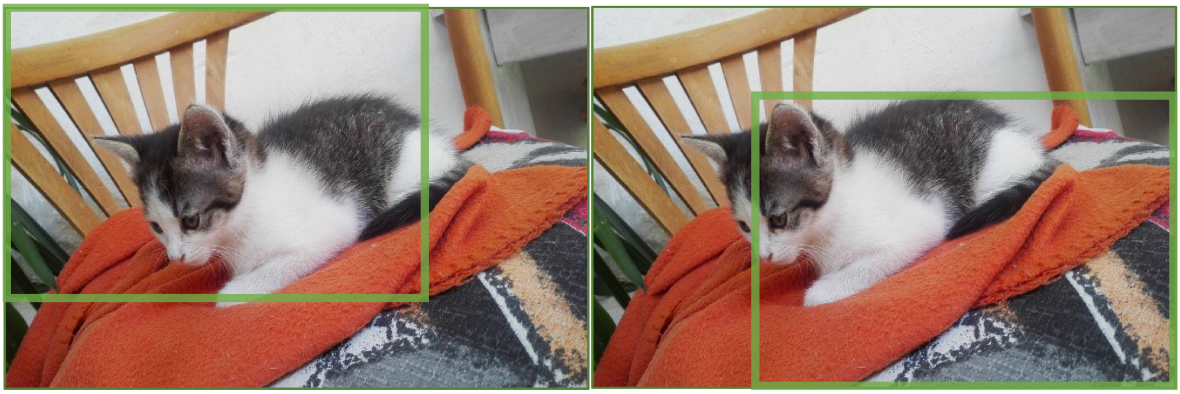

Figure 12: Starting and ending position of sliding window technique.

In practice, we have the same object at different scales within different images. This will cause low accuracy for a simple classifier, while the classifier will see the same object, but within different scales. Thus, we need to use also another method, called image pyramid.

2.2.2 Image pyramid

Detection of objects at different scales is one of the main object detection problems. Image pyramids help to solve this intuitively. The main idea is to convolve different scales of one image. The schema of the image pyramid [13] is shown in Fig. 13.

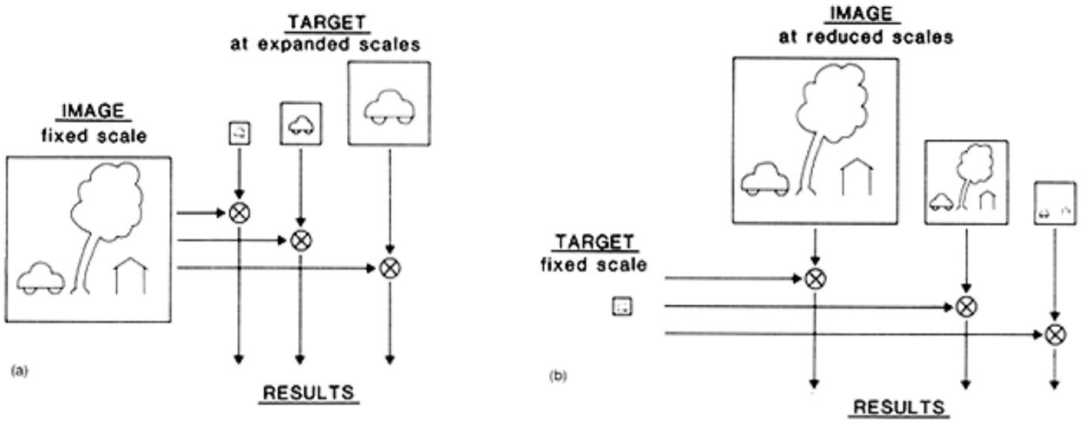

Figure 13: Examples of two image pyramid methods. The left one with images at expanded scales and the right one with pyramids at reduced scales [13]

The main issue of the image pyramid is high calculation cost because all scaled images must go through a neural network in the training and also testing phase.

2.2.3 Pyramid of features

The feature pyramid is using similar logic as the image pyramid. In the case of feature pyramid, a neural network for classification does not proceed with a few images of different scales. Neural network proceeds only one image at once. As an image is going through the network width and height dimensions of the features are decreasing. The features are extracted and then used for predictions. Visual comparison of the pyramid of images and pyramid of feature maps is shown in Fig. 14.

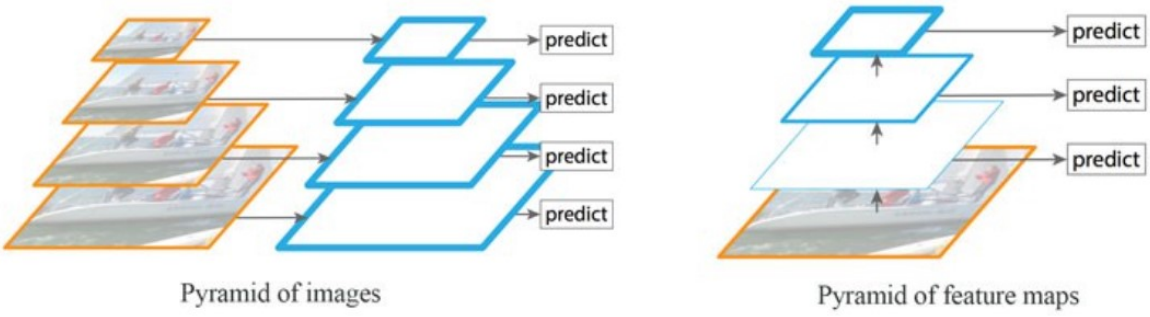

Figure 14: Visual comparison of the image pyramid and feature pyramid methods [14]

2.2.4 IoU – Intersection Over Union

Intersection over Union is simply an evaluation metric. In the case of the classification of objects, we have only True or False predictions. But we cannot use the same method for predicted object locations because then we lose information on how bad or good our prediction is. Intersection over Union solved this problem [15]. The typical example of prediction is shown in Fig. 15.

Figure 15: Example of predicted bounding box and ground-truth box

For evaluation, we need a predicted bounding box and a ground-truth bounding box. The image below shows both boxes and IoU calculation [15].

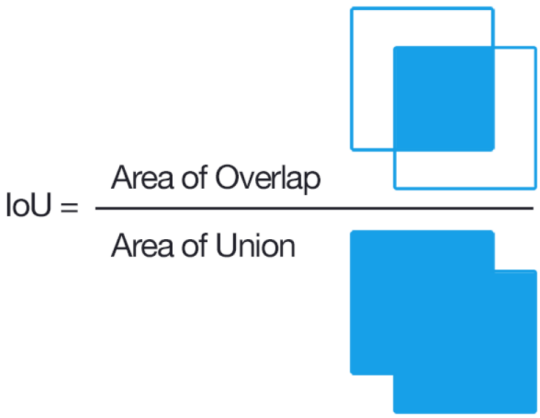

Figure 16: Visualization of Intersection over Union [15]

Intersection over Union is simply a ratio between Area of Union and Area of overlap. That means IoU is 1 when ground-truth and predicted box are the same. IoU = 0 means that boxes are not overlapping each other [15].

2.2.5 Non-max suppression

Now we know how to recognize objects in different locations and different scales with sliding windows and image/feature pyramid methods. But with these 2 techniques we can have more region proposals for one object as shown in the picture below:

Figure 17: The example with more overlapping predictions for one object. Non-max suppression should be used to get the best prediction for the object

Non-max suppression filters proposals for one object based on defined criteria. The most used is filtering based on confidence scores of predictions and IoU between overlapping prediction regions. NMS algorithm takes detections with the highest scores and removes all overlapping lower-scoring detections which have an IoU greater than a pre-defined threshold [15].

2.2.5.1 Anchor boxes

The first detection systems cannot correctly predict more overlapping objects (practice example is shown in Fig. 18) and the researcher were developing methods to solve it. Problem was that the first neural networks can predict only one class for one coordinate on an image.

Then anchor boxes come on the playground. This technique generates more anchor boxes for every coordinate. Normally there are 9 anchor boxes per one coordinate. Anchor boxes have different aspect ratios and different scales (scales and ratios should be defined based on relevant objects in the dataset). This allows predicting more objects and more classes for every coordinate on the image. Nowadays, anchor boxes are a standard method used by top object detection systems as Faster R-CNN, YOLO, and others [15].

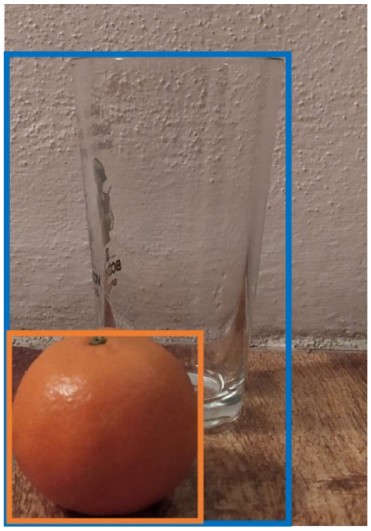

Figure 18: Example of overlapping objects of different aspect ratios and different scales

2.3 Evaluation metrics

There are more standard evaluation metrics for object detections:

- **PASCAL VOC Challenge** used the Precision x Recall curve and Average Precision.
- **COCO Detection Challenge** used the 12 metrics used for characterizing the performance of an object detector on COCO.
- **IMAGENET OBJECT LOCALIZATION CHALLENGE** defines an error for each image considering the class and the overlapping regions between ground truth and detected boxes. The total error is computed as the average of all errors.

From the metrics mentioned above, COCO metrics are recently most used for comparisons of object detection systems in scientific articles. The metrics are calculated for the best 100 predictions. Here are the main COCO object detection metrics:

- **Precision** (2) - is the ability of a model to identify only the relevant objects. It is given by:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

- **Recall** (3)- is the ability of a model to find all the relevant cases (all ground-truth bounding boxes). It is the percentage of true positive detected among all ground-truth boxes and the formula is as follows:

$$Recall = \frac{True\ Positive}{True\ Positives\ + False\ Negatives} \quad (3)$$

- **Precision x Recall curve** – shows a relationship between Recall and Precision for one detection system with different confidence threshold scores. An example is shown in Fig. 19.
- **Average Precision** – is not average of the precision. It is the area under the curve (AUC) of the Precision x Recall curve [16].

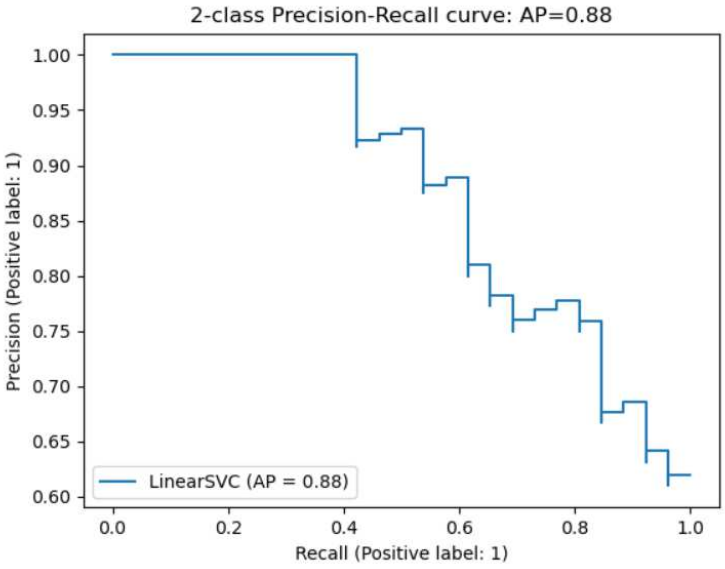

Figure 19: Precision x Recall curve for one class [17]

3 OBJECT DETECTORS

Object detection is the main part of computer vision and is getting more attention in recent years. The object detection systems can predict classes and coordinates of all objects on the image. This is useful mainly when we have more relevant objects on one image and simple image recognition methods cannot help in this case.

There are two main categories of object detection systems: 1 and 2 stage detectors. The author of the thesis will describe the top detectors of both categories in this chapter [18].

3.1 One-stage detectors

One-stage detectors are in general faster than two-stage detectors with similar accuracy. This chapter describes OverFeat, YOLO, and Scales YOLOv4 in detail.

3.1.1 OverFeat

The idea of OverFeat is to use a convolutional network to simultaneously classify, locate and detect objects with a single Convolutional Neural Network. This approach can simplify training and increase speed and classification, detection, and localization accuracy.

OverFeat uses the image pyramid method and the sliding window is replaced by CNN. Image of 14x14 input is transferred through CNN to 1x1 output picture as shown in Fig. 20. For bigger inputs more regions of 14x14 are transferred to spatial output, so we can locate objects, as you can see on the image. In addition, because the fully connected layers cannot handle different input sizes, they were replaced by a 1x1 convolution filter [18].

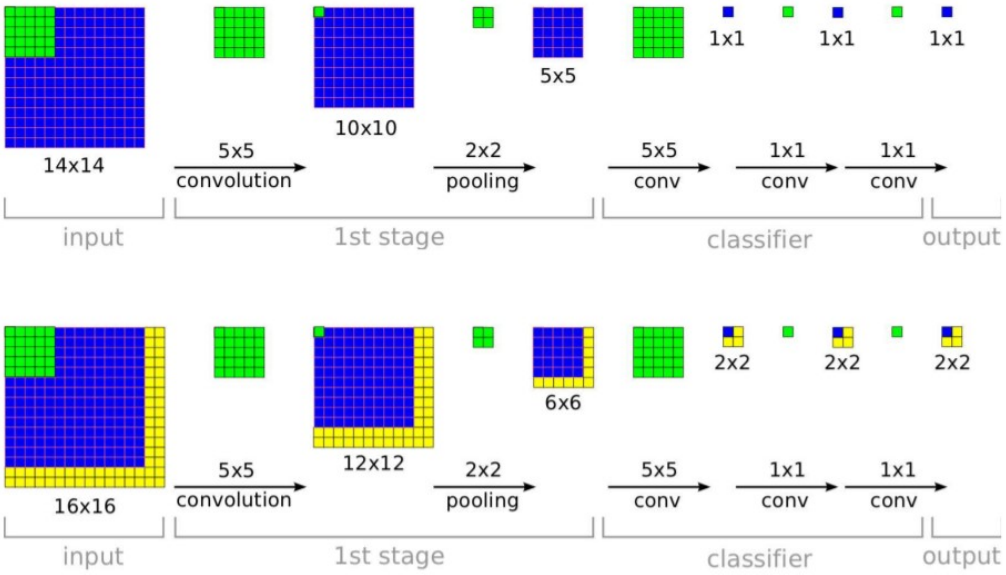

Figure 20: OverFeat detection system with information flow from input to output [18]

3.1.2 Localization and Detection pipeline

The raw classifier outputs a class and confidence for each location. The resolution of these predictions can be increased using Multi-Scale Classification [18]. The regression network then predicts the location of the object for each window of a different scale as shown in Fig. 21.

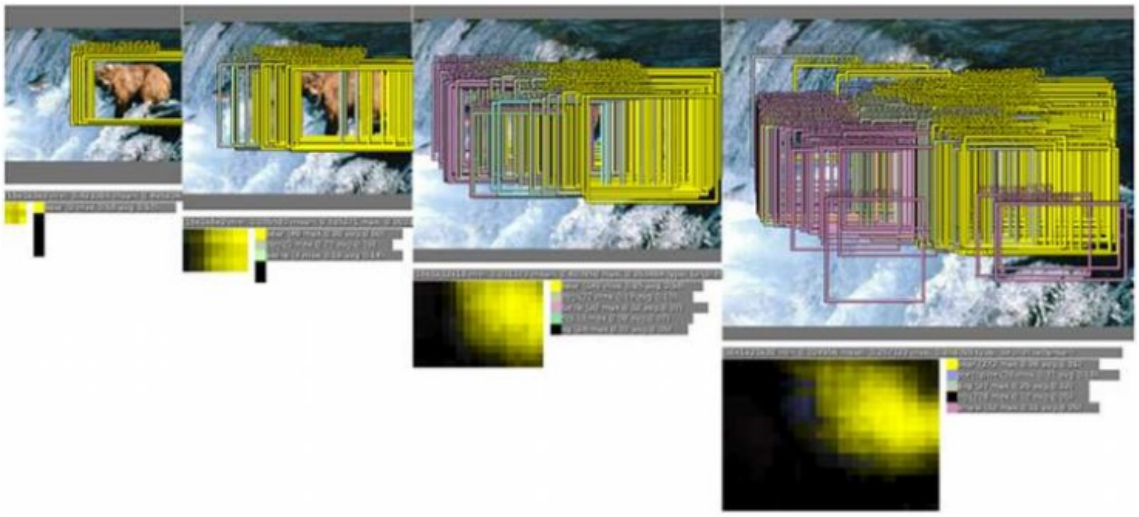

Figure 21: Visualization of localization of objects at different scales using Multi-Scale Classification [18]

These bounding boxes are then merged and accumulated with non-max suppression to a small number of objects. The final result is shown in Fig. 22.

Figure 22: The final prediction of the network [18]

3.1.3 YOLO (You Only Look Once)

YOLO is the state-of-the-art detection system developed in 2016. Next versions of YOLO outperform all other systems in case of speed and retain very good accuracy [19]. A comparison of the best non-scaled YOLOs is shown in Fig. 23. The image express correlation between inference time of the detection systems and mean Average Precision on MS-COCO test-dev dataset.

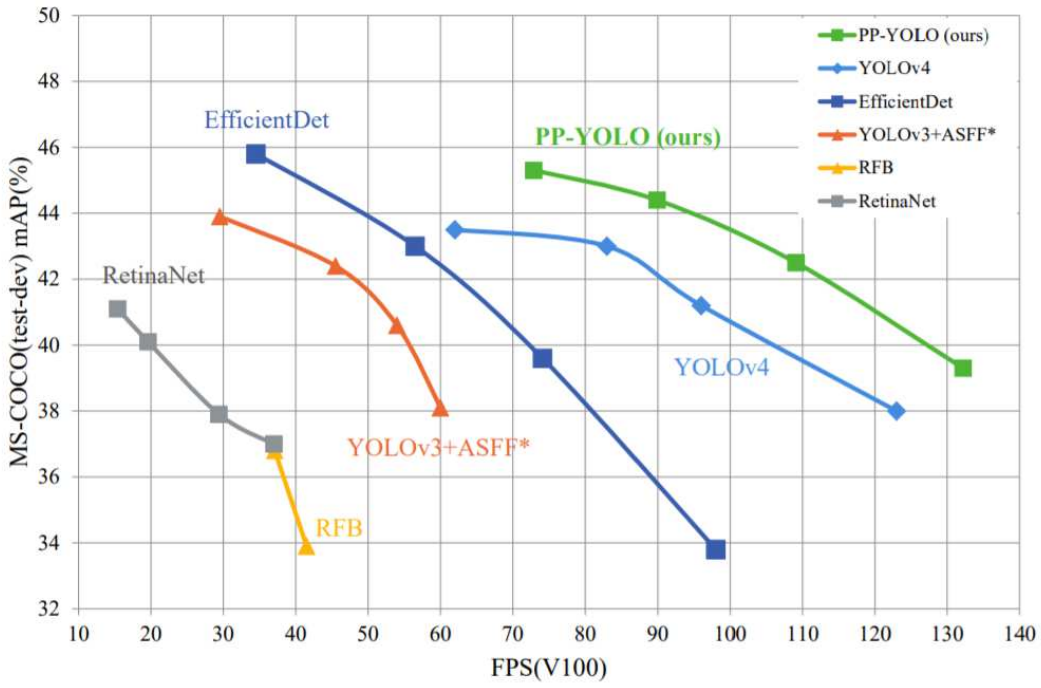

Figure 23: Comparison of a few state-of-the-art networks based on speed and mean Average Precision on COCO benchmark dataset [19]

YOLO's main advantages are speed and generalization. Unlike sliding window and region proposal-based techniques, bounding box predictions are made based on all pixels in the image. Thus, it encodes also contextual information about classes [20], [21].

YOLO frame detection as a regression problem. Thus, it does not need an additional network for classification and needs only one run of a single neural network on images to predict detections. Processing images consists of three steps [20], [21].

1. The system resizes the input image to 448 × 448.
2. runs a single convolutional network on the image.
3. Non-max suppression.

Visualization of the pipeline is as follows:

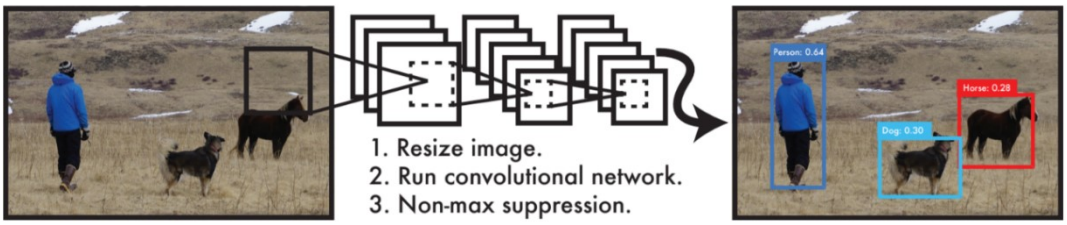

Figure 24: Three steps of the YOLO pipeline: resize the image, run the convolutional network, and non-max suppression [20]

In the details, YOLO at the first divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities as shown in the middle of Fig. 25. These predictions are encoded as an S × S × (B ∗ 5 + C) tensor on output. Then, bounding boxes with the highest probabilities are chosen and final predictions of locations are made, shown in the right Fig. 25.

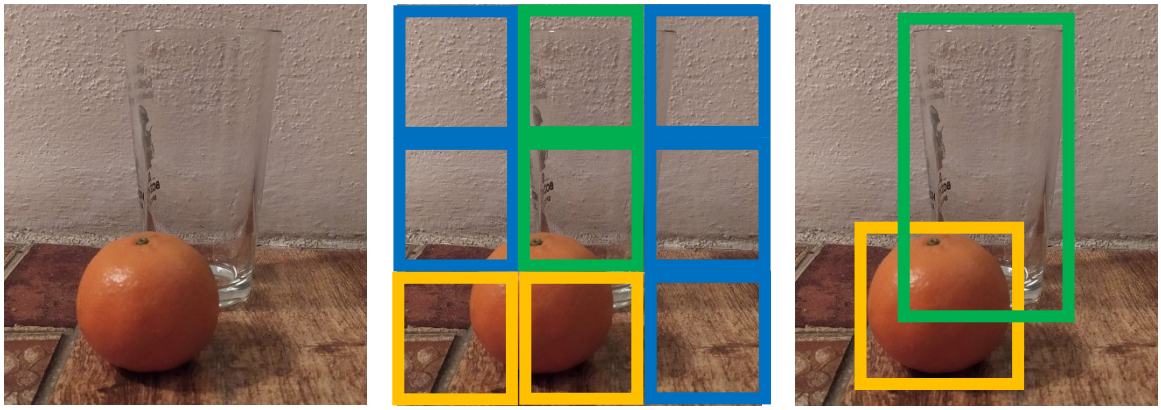

Figure 25: An input image on the left, locations with predictions in the middle and final predictions in the right part of the example [20]

YOLO uses the anchor boxes technique, so it can predict at every coordinate more than one bounding box, usually 9.

YOLO's network architecture is inspired by the GoogLeNet [22] model for image classification. The network has 24 convolutional and pooling layers followed by 2 FC layers. Alternating 1 x 1 CNN layers reduce the feature space [20]. The network is shown in Fig. 26.

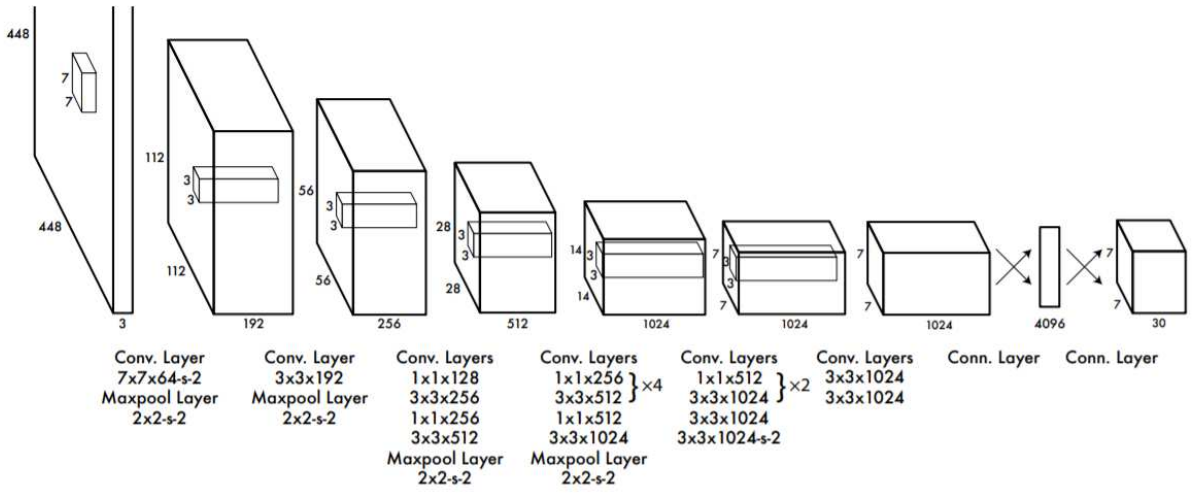

Figure 26: Detection network with 24 CNN layers and 2 FC layers [20]

YOLO (the first version) model struggles with small objects that appear in groups and also with a generalization of objects in unusual aspect ratios or configurations [20]. This issue occurs because YOLO looks at the whole image.

3.1.4 Scaled-YOLOv4

Scaled-YOLOv4 is recently one of the best-published neural networks for object detection. Scaled-YOLOv4 has 55.8% AP on the COCO test-dev dataset. Only Swin-L [23] and Cascade Eff-B7 NAS-FPN [24] from the year 2021 are more accurate. But it is still the best in terms of the ratio of speed to accuracy.

Scaled-YOLOv4 consists of a few neural networks: CSP, P5, P6, P7. On the chart below, you can see a comparison of these networks and others like EfficientDet [25], SpineNet [26], other-YOLOv4 detection systems. Accuracy is on axis y and the speed of the system on axis x.

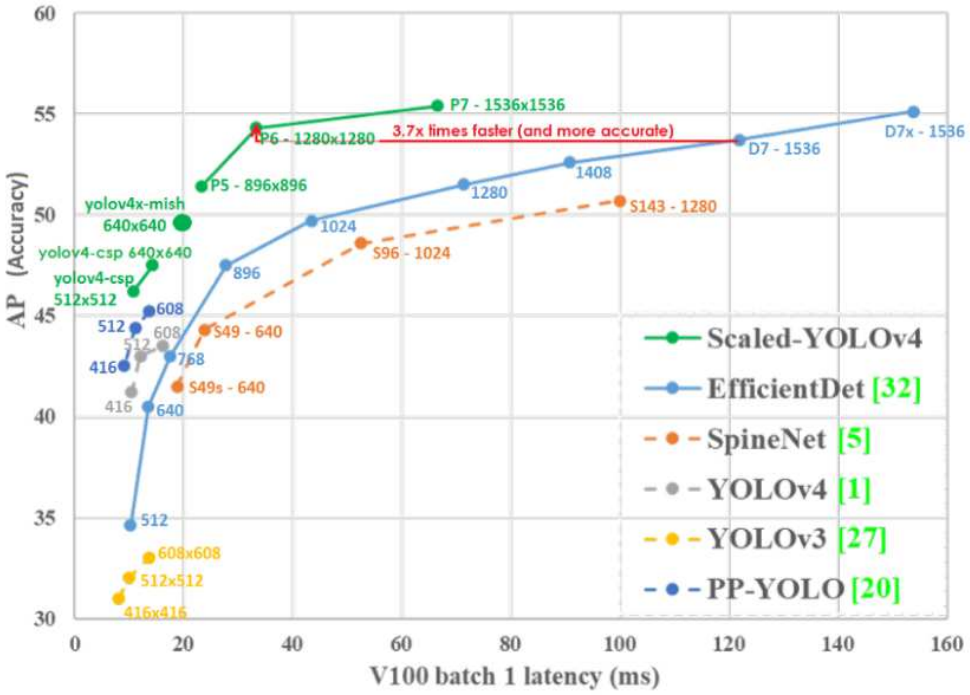

Figure 27: Comparison of accuracy and latency for different YOLOs, SpineNet and EfficientDet (GPU V100 for batch size = 1) [27]

As you can see on Fig. 27, P7 is best in terms of accuracy, and in general, all Scaled-YOLOv4 systems are best in the ratio between speed and accuracy [27]. These improvements were caused by the changes described later in the chapter.

Scaled-YOLOv4 has a few main improvements over YOLOv4. At the first, it uses network scaling methods to get from YOLOv4-CSP large-scale networks as P6 and P7. Larger networks are trained on images with larger resolutions to achieve higher mean average precision. On the other side inference time (same for training time) of the network is also increasing as shown in Table 1 below. Table comparing different Scaled-YOLOv4 networks base on FPS, size of the image, and different mAPs on test dataset [24].

Table 1: Speed and accuracy comparison of large Scaled-YOLO networks on COCO benchmark dataset (mAPS = Average Precision on small objects, APM = Average Precision on medium objects, APL = Average Precision on large objects) [28]

Model	Test Size	mAPtest	mAP50test	mAP75test	mAPStest	mAPMtest	mAPLtest	batch1 throughput
YOLOv4-P5	896	51.4%	69.9%	56.3%	33.1%	55.4%	62.4%	41 fps
YOLOv4-P5	TTA	52.5%	70.3%	58.0%	36.0%	52.4%	62.3%	-
YOLOv4-P6	1280	54.3%	72.3%	59.5%	36.6%	58.2%	65.5%	30 fps
YOLOv4-P6	TTA	54.9%	72.6%	60.2%	37.4%	58.8%	66.7%	-
YOLOv4-P7	1536	55.4%	73.3%	60.7%	38.1%	59.5%	67.4%	15 fps
YOLOv4-P7	TTA	55.8%	73.2%	61.2%	38.8%	60.1%	68.2%	-

3.1.4.1 Network scaling method

Scaling of YOLO is done by Cross-Stage-Partial (CSP) connections [29]. CSP scaling was inspired by EfficientDet [25] scaling method. CSP connections are efficient, simple, and can be applied to any neural network. CSP working as follows: half of the output signal goes along the main path (generates more semantic information with a large receiving field) and the other half of the signal goes bypass (preserves more spatial information with a small perceiving field). A simple comparison of ResNeXt and CSP ResNeXt is shown in Fig. 28.

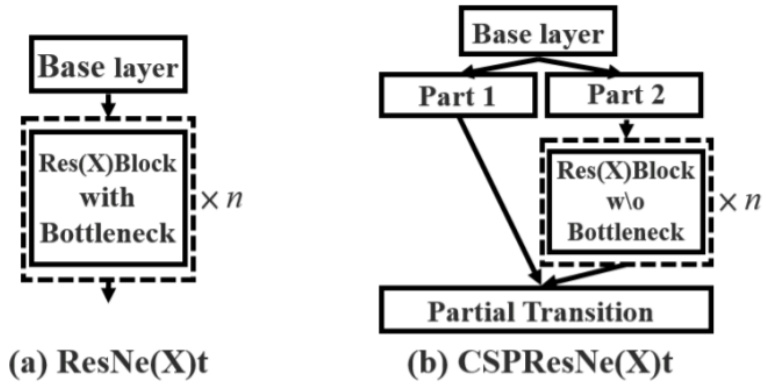

Figure 28: The simplest example of a CSP connection (on the left is vanilla ResneXt network, on the right is a CSP network with ResneXt) [29]

Below you can see the special case of CSP connections used in Scaled YOLOv4. Here the connections allow part of the signal to skip 3 convolutional layers and one SPP layer.

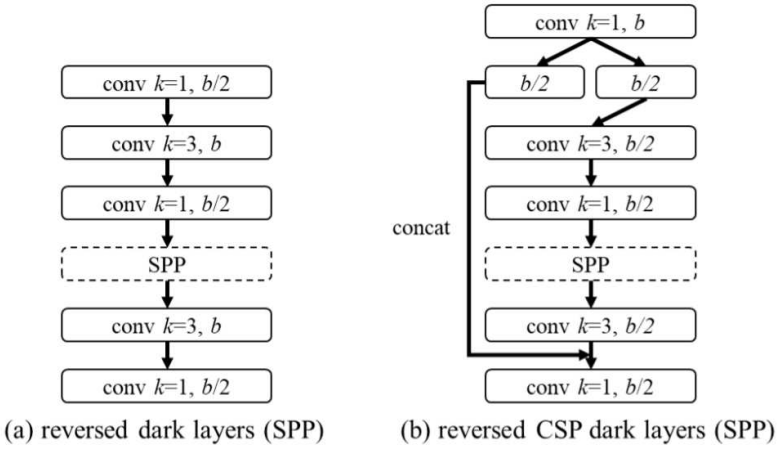

Figure 29: An example of a CSP connection in Scaled YOLOv4-CSP / P5 / P6 / P7 (on the left are reversed dark layers with SPP, on the right is reversed CSP dark layers with SPP) [29]

3.1.4.2 Other improvements

Here are other main improvements of Scaled YOLOv4 over YOLOv4:

- Exponential Moving Average (EMA) is used during training. EMA improved training with small batches.
- Improved loss calculation
- Different neural networks are developed for different image resolutions
- Mosaic augmentation [27]

3.2 2 stage detectors

Two-stage object detectors were developed in parallel with one-stage detectors. In general, 2 stage detectors are slower because of more stages, but more accurate. These systems consist of 1 part for finding regions of interest (RoI) and the second part for classification (sometimes include also part for adjustment of location) of an object in actual RoI.

3.2.1 R-CNN

Region proposals with CNN come after over-feat. A new idea of R-CNN is the usage of a selective search algorithm to find region proposals in R-CNN.

R-CNN consists of three modules. The first generates 2000 category-independent region proposals. The second module is a convolutional neural network, which produces features from each region. The third module is a set of class-specific linear SVMs. SVMs generate a score for each feature vector and each class. After that, non-maximum suppression is used to filter proposals. The system also predicts offset values to adjust the bounding box of the region proposal. Detailed model is shown in Fig. 30.

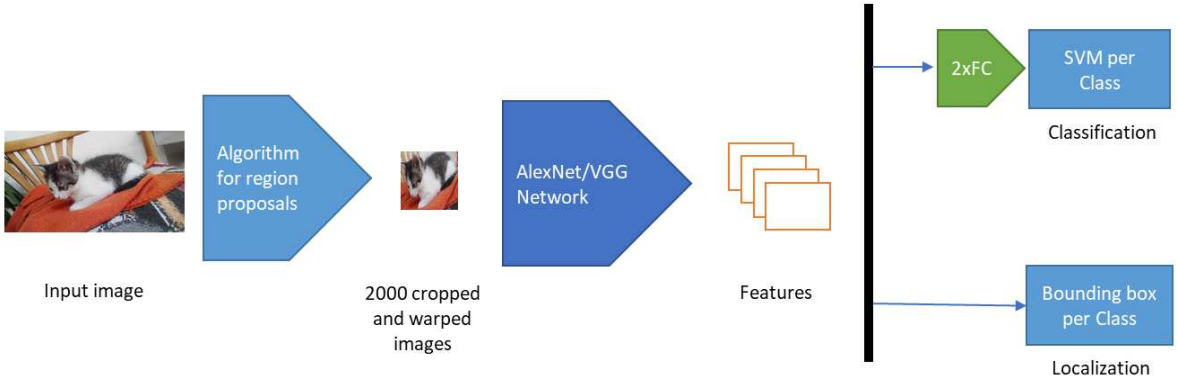

Figure 30: Main modules of R-CNN. Namely generator of region proposals, feature extractor, classification network, and bounding box regressor [30]

R-CNN has few fundamental drawbacks because of its architecture. First, we need to warp every image because of AlexNet can only work with region proposals with 1 aspect ratio. In addition, the warped image has different features than the original so we get worse accuracy. To overcome this, we need to train AlexNet on warped images. The second problem is that it is not possible to train the whole system in one run. R-CNN first finetunes a ConvNet on object proposals using log loss. Then, it fits SVMs to CNN features. These SVMs act as object detectors, replacing the Softmax classifier learned by fine-tuning. And lastly, bounding box regressors are learned. The third, selective search algorithm is a fixed algorithm, so it cannot learn [30], [31].

3.2.2 Fast R-CNN

Fast R-CNN builds on R-CNN and SPPnet to efficiently classify object proposals. Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy [31].

Fast R-CNN takes Region Proposals module and AlexNet/VGG CNN from R-CNN. But our convolutional neural network now proceeds 1 image once and not 2000 proposals. This is the reason why Fast R-CNN is 213 times faster at test-time than R-CNN.

After proposals generation, CNN generates feature maps, which are filtered for every region proposal. After that, our filtered features are going through the SPP Region of Interest (RoI) pooling layer 7x7. The pooling layer can proceed with inputs of different sizes and will produce output with 1 size. Thus, image warping is not needed here. Lastly, the SVM is replaced with a Softmax layer, so we do not need 3 stages of training and can train our CNN, Softmax for Classification, and Bounding box regressor at once. Additionally, Fast R-CNN uses combined losses of Softmax layer for classification and Bounding box regressor, which caused improved accuracy compared to R-CNN. Visualization of Fast R-CNN modules is as follows:

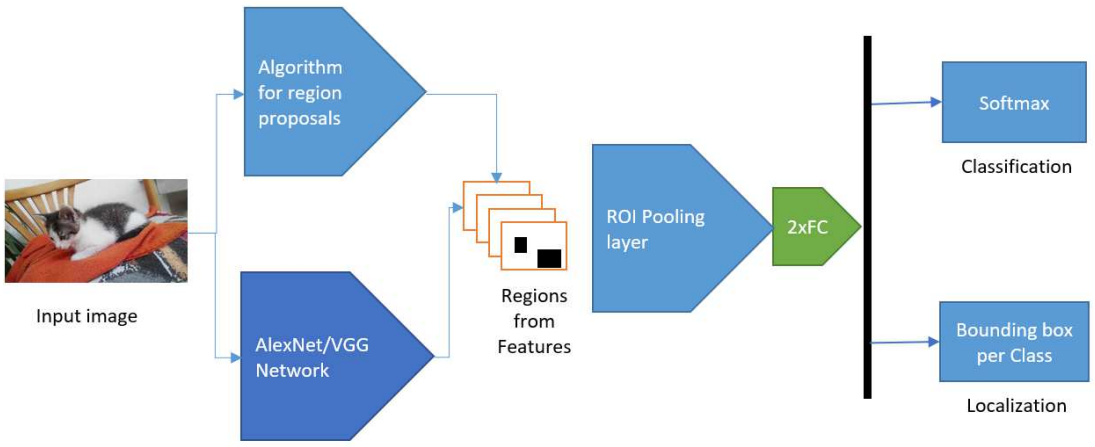

Figure 31: Visualization of modules of Fast R-CNN. Namely generator of region proposals, feature extractor, ROI Pooling layer, classification network, and bounding box regressor.

Another possibility of how to increase accuracy or increase speed of Fast R-CNN is to change a number of region proposals. Accuracy is increasing only slightly from 1000 to

3000 region proposals and then start to decrease because the RoI pooling layer is fed by more regions with no objects, thus can produce more wrong detections. Comparison of mean Average Precision and Average Recall versus number of object proposals is shown in Fig. 32 below.

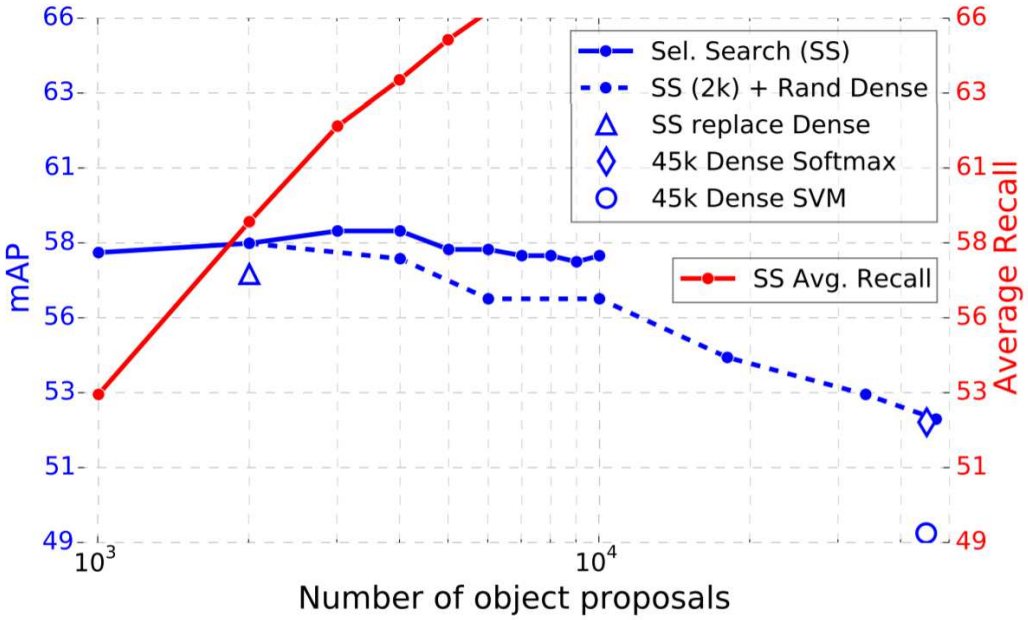

Figure 32: Comparison of mean Average Precision and Average Recall versus number of object proposals [31]

3.2.3 Faster R-CNN

Faster R-CNN is the third generation of R-CNN. It uses 2 networks: attention network – region proposal network(RPN) and network for object detection. Unlike Fast R-CNN, here we do not use selective search to generate 2000 region proposals. Faster R-CNN is faster because RPN uses already computed features from the object detection network [32].

RPN at the first generates object presence probability for all anchor boxes, then it uses non-max suppression to filter anchor boxes with IOU more than some coefficient base on scores of these anchor boxes. And at the latest, it filter 2000 best prediction boxes based on the score [32].

3.2.3.1 Anchors

Anchors play an important role in Faster R-CNN. In the default configuration of Faster R-CNN, there are 9 anchors at every position of an image. There is similar number of

combinations of positions as for sliding window and pyramid technique. Fortunately, here we we have region proposal network, which eliminates a lot of anchor boxes.

3.2.3.2 Region Proposal Network (RPN)

RPN is main improvent over Fast R-CNN. The output of a region proposal network are proposals. In nutshell, RPN predicts if anchor is background or foreground. Then defined proposals are moved to a classifier and regressor [32].

3.2.3.3 ROI Pooling layer

After RPN, we get proposals with different sizes (same like for Fast R-CNN). Thus, we have different sized feature maps. Region of Interest Pooling reduces the feature maps into the same size. Therefore the output of ROI Pooling is always k regardless the size of input [32], [31]. Simplified schema of Faster R-CNN is as follows:

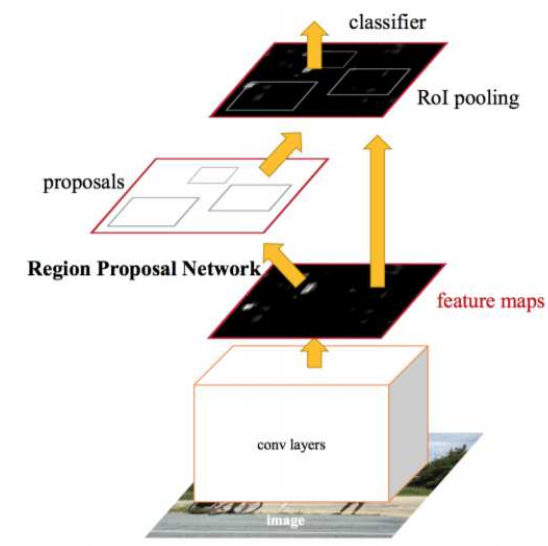

Figure 33: The latest incarnation of the R-CNN family, Faster R-CNN, introduces a Region Proposal Network (RPN) that bakes region proposal directly in the architecture, alleviating the need for the Selective Search algorithm [32]

4 PEST DETECTION

Pest recognition is an active research topic in the last years. In most cases, computer vision, machine learning, and deep learning methods are selected and used to detect plant disease.

Computer vision and mainly object recognition improved a lot in the last years. Traditional methods for image classification tasks have been based on features detection algorithms, such as DoG, Salient Regions, and SIFT [33]. These methods have used manually extracted features, then some learning algorithms are trained on these features. The performance of the approaches depends on predefined hand-engineered features. Thus, it depends a lot of times on the experiences and skills of engineers in the field. In addition, feature engineering is a hard process which needs to be revisited when the problem or the dataset change.

Fortunately, deep learning solved the feature extraction problem with automatic feature extraction. For this work is used a dataset from Pest Detection and Identification on Tomato Plants for Autonomous Scouting Robots Using Internal Databases [33].

4.1 PlantVillage project

The PlantVillage project collected thousands of images of healthy and diseased plants images. PlantVillage dataset is used for training deep learning models for different disease diagnoses. It contains 18 160 tomato images of different diseases as septoria leaf spot, tomato yellow leaf curl virus, target spot, early blight, etc. Unfortunately, the dataset cannot be used for pest detection of insect stage (an insect or egg level). Thus, we cannot use it to improve our training [33].

4.2 Pest Detection and Identification on Tomato Plants for Autonomous Scouting Robots Using Internal Databases

This dataset is labeled at insect and egg level, it is composed of original pictures of plants. The deep learning approach in the work is focused on the detection of the disease using object detection frameworks. Deep learning models are based on Faster RCNN and SSD [35]. Results of their work showed great accuracy results of Faster R-CNN systems with augmentation. Thus, Faster R-CNN will be used also in the next chapters of the practical part of this thesis [33].

II. ANALYSIS

5 PREREQUISITES FOR TRAINING OF DETECTION SYSTEMS

The first part of this chapter describes hardware and software needed for training. The second part is about dataset split and dataset quality. And the last part discusses used evaluation metrics.

5.1 Hardware used for training

Cloud computing is the best way to run high-performance computer technology without investing in the purchase of GPUs. These services include Azure, AWS, Oracle, and Google Colab. The author of the thesis chose Google Colab because it is a low-cost solution with a lot of manuals.

Google offers 2 possible setups for neural network training. Google Colab setup offers Tesla K80 and a maximum of 12 hours of continuous training and Google Colab Pro Tesla V100 with a maximum of 24 hours of continuous training. Complete comparison in table form is as follows:

Table 2: Comparison of Google Colab and Google Colab Pro

Parameter	Google Colab	Google Colab Pro
GPU	Tesla K80 16GB	Tesla V100 16GB
Max training time	12 hours	24 hours
Disk space	34 GB	34 GB
RAM	12,8 GB	26,3 GB
Price per Month	0 USD	9.99 USD

The author of the thesis decided to train models on Google Colab Pro with Tesla V100 and the author of the thesis recommends doing it for all, who need to train really deep networks as Scaled-YOLOv4 P6 or P7. Detailed training times of the networks will be described later in the details in the next chapters.

5.2 Dataset split

Dataset consists of 4330 images. Images were randomly (random seed 0) divided into a training dataset with 3900 images and a testing dataset with 430 images. These divided datasets were used for all detection systems discussed later.

5.3 Dataset quality

Dataset quality is one of the most important prerequisites for accurate neural network training. Pest benchmark dataset suffers from low dataset quality. One of the troublesome issues is missing labels for pests as is shown in Fig. 34 (different colors correspond with different pest classes). Missing labels in the training dataset cause the detection system considers areas with pests as a background which harms the network performance. On other hand. missing labels in testing dataset cause, that network predicts bounding boxes where the real pests are, but ground-truth bounding boxes are missing. This manifests in a lower overall precision value.

Figure 34: The example of missing ground-truth bounding boxes

The second issue is missing ground-truth bounding boxes, which do not fit correctly real objects as is shown in figure 35 below. There you can see, that bounding boxes are more than 6 times bigger than real pests. Thus, the IoU of the ground-truth bounding box and the actual bounding box (if correctly assigned) is only 0.166.

Figure 35: The example of bigger ground-truth bounding boxes than real bounding boxes¨

The last but also troublesome issue relates to labeling the group of pets as one object. Because of this, the network predicts groups of pests in coordinates where are only smaller bounding boxes with no pest, and predictions are evaluated as wrong and vice versa. Examples of bounding boxes of groups are shown in Fig. 36.

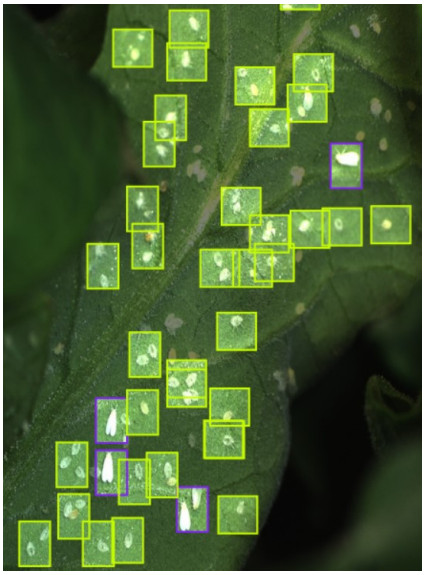

Figure 36: Example of bounding boxes with groups of pests

The author of this thesis tried to improve the test dataset for evaluation in his experiments to obtain a more realistic overview of the network performance. Details about experiments are in chapter 9 Dataset improvements.

5.4 Evaluation strategy

In general, predictions are correct if predicted bounding box lies close to ground-truth bounding box and predicted class is the same as the real class of an object. There are more possible methods to define which predicted objects are evaluated and which not.

5.4.1 Prediction confidence (score)

The author of the thesis used the most used and standardized benchmark evaluation method, what consider the best 100 predictions (base on confidence score) for every image for evaluation in networks comparison parts in upcoming chapters. And the author of this thesis used only filtered predictions based with confidence threshold > 0.6 for comparison my results with results from benchmark paper [33], achieved on the same dataset.

5.4.2 IOU

IOU is a metric is used in most state of art object detection algorithms. The metric finds the difference between ground truth annotations and predicted bounding boxes. Here we have two main standards IOU thresholds. The first considers all predictions with IOU > 0.75 between predicted and ground-truth box as correct and second all predictions with IOU > 0.5. The author of the thesis used the second evaluation for 3 reasons:

1. Ground-truth boxes do not fit correctly on objects in the dataset.
2. Prediction of correct coordinates is not so important in practice as pests' predictions.
3. Pest benchmark paper [33] used this logic, so the comparison will be easier.

5.4.3 Precision, Recall and mean Average Precision

There are a few standard evaluation metrics for object detection from different competitions. The author of this thesis used COCO metrics as they are recently most used for comparisons of object detection systems. Precision and Recall are calculated for 100 predictions with the biggest prediction scores. And the author of this thesis used also the Precision x Recall curve. All these metrics were described in the detail in the theoretical part of this work.

6 PEST DETECTION WITH FASTER R-CNN WITH FPN

Faster R-CNN is one of the most used and researched two-stage object detection systems. The first versions used images with low resolution and not so deep backbones as AlexNet. But our pest dataset consists of images with high resolution 2064 x 1544 pixels with some pests, which have only 30 x 30 pixels. Thus, the author of the thesis decided to use more advanced and deeper backbones for Faster R-CNN with shortcuts between layers: namely Resnet and ResneXt. Feature pyramid network was also used for better propagation of features from deeper to shallow layers to increase more small object detection accuracy.

Some parts of the code were derived from the torch tutorial [36].

6.1 General setup for all networks

This chapter describes main training parameters for Faster R-CNN as anchor generator, training scheduler, and augmentations.

6.1.1 Anchor generator

All Faster R-CNN systems used 15 generated anchors, concretely anchors with a size of 16, 32, 64, 128, and 256 pixels and ratios width: height 0.5, 1, and 2. Anchors should fit as much as possible to all labeled objects in images. But regressor of Faster R-CNN can handle also objects with different sizes and ratios and predict correct location. Thus, deletion of some anchor size has only a minimal effect on final accuracy results. This is the case mainly for anchors with sizes of 128 and 256 pixels, which were used only for tomato class detections and not for pest detections.

6.1.2 Weights training algorithm

Stochastic Gradient Descent (SDG) algorithm was used for training with learning rate = 0.005, momentum = 0.9, and weight decay = 0.0009 for all Faster R-CNN training. Learning rate scheduler was also used for adjustment of learning rate through training. Parameters of scheduler are step size = 2 and gamma = 0.6. These parameters mean that every second epoch is a learning rate multiplied by gamma = 0.6. After a few epochs learning rates is lower and lower and the small learning step adjusts slowly weights of neural networks and converges to the optimum.

In addition, also Cosines Annealing method was used for the learning rate schedule. The method uses the annealing method to adapt the learning rate for faster convergence and

jumps from local minimums. This method achieved better accuracy results, but the author of this thesis does not use it for final comparison in chapter 8, because training is too unstable and not easily replicable. Losses during training are shown in Fig. 37 below.

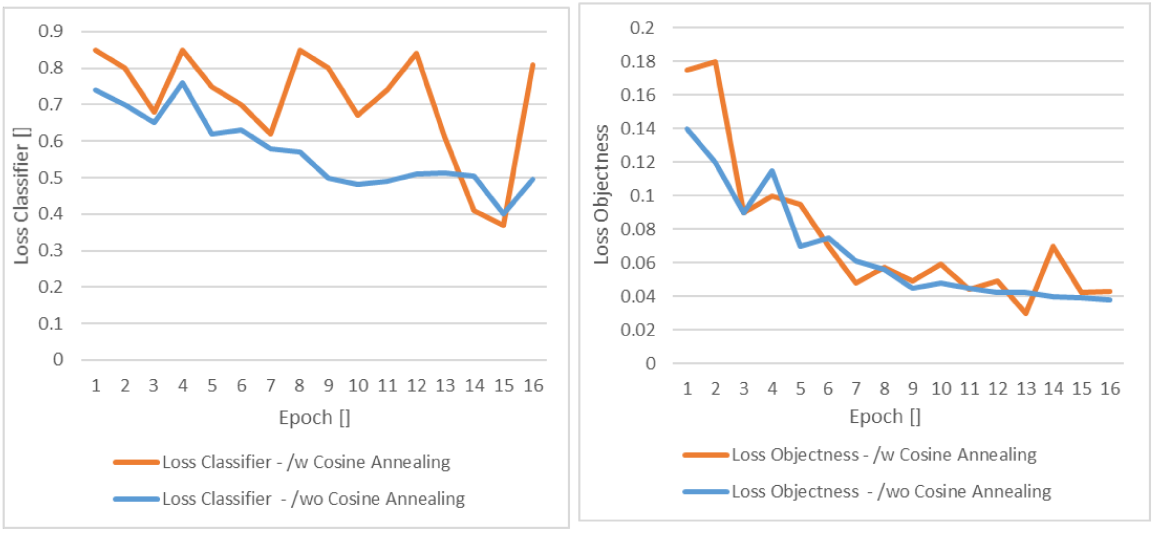

Figure 37: Comparison of training losses with and without Cosines Annealing scheduler

6.1.3 Training setup

Net training of Faster R-CNN took always less than 24 hours, which is important for training on Google Colab Pro can be done in one run without interruptions. ResneXt32 and Resnet152 took more time because there are deeper than Resnet50. The batch size was chosen to the possible maximum, based on the maximum capacity of GPU memory, which is 16 GB. Because in general, bigger batch size causing the increased quality of training due to used batch normalization method.

Table 3: Comparison of Faster R-CNN training with different backbones.

Backbone	Batch size	Training time per epoch	Number of epochs	Overall training time
ResneXt32	4	60 min	20	20 hours
Resnet50	8	30 min	20	10 hours
Resnet152	4	70 min	20	23 hours

All backbones pretrained on the COCO benchmark dataset were used and 5 backbone layers were then trained on the training dataset. Also, training without a pretrained backbone was

done, but the neural network did not converge in these experiments, which caused wrong results with overall Recall and Accuracy under 0.1.

6.1.4 Augmentations

The Albumentation framework [34] was used for Faster R-CNN augmentations. Concretely functions, that cause changes of brightness with probability 0.3 and horizontal plus vertical flip, both with probability 0.5.

The author of the thesis tried also more advanced methods as shifts, crops, and rotations. But accuracy was not better and the network suffers from NaN loss through training, which caused the end of the training.

Resnet152 is not augmented as augmentations of the data caused system exit error during training. But it could be part of future work.

6.2 Pest detection results

In general, Faster R-CNN with FPN performs quite well for bigger objects, namely BT and WT classes but egg prediction accuracy was relatively low even with consideration of dataset quality. Detailed results on the test dataset can be seen in Tab. 4.

Table 4: The test set precision and recall results of Faster R-CNN models with different backbone networks and training setups (AP = Average Precision, AR = Average Recall, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci)

Backbone	**WF egg AP**	**WF egg AR**	**BT egg AP**	**BT egg AR**	**WF AP**	**WF Recall**	**BT AP**	**BT AR**	**All AP**	**All AR**
Resnet50	0.108	0.495	0.032	0.222	0.403	0.791	0.345	0.715	0.194	0.552
Resnet50 augment	0.107	0.491	0.033	0.201	0.426	0.774	0.363	0.719	0.210	0.508
Resnet152	**0.121**	**0.509**	**0.038**	**0.249**	**0.504**	**0.808**	**0.346**	**0.679**	**0.223**	**0.522**
ResneXt32	0.116	0.492	0.068	0.308	0.459	0.728	0.261	0.595	0.210	0.501
ResneXt32 augment	0.117	0.490	0.070	0.299	0.485	0.703	0.270	0.601	0.216	0.484
Resnet50 annealing	**0.139**	**0.522**	**0.041**	**0.205**	**0.463**	**0.798**	**0.335**	**0.697**	**0.221**	**0.568**

As results showed, backbone Resnet50 with cosine annealing scheduler and without augmentation have the best overall results. But already mentioned unstable training with big deviations in accuracy and training losses caused by cosine annealing scheduler will cause

low accuracy on the validation dataset. Therefore, the author of the thesis considered Resnet152 without augmentation as the best backbone for Faster R-CNN. The reason behind it is that Resnet152 is a deeper network, so can better leverage small objects features from deep layers through a neural network.

The best in egg predictions were ResneXt with augmentation and Resnet152. Resnet152 was the best also in the case of WF and Resnet50 in the case of BT.

Disbalance between precision and recall is mainly caused by false negative predictions caused by missing labels for pests as was mentioned in chapter 5.

6.2.1 Accuracy versus speed

Experiments also showed that an additional increase in depth of the backbone is causing an overall increase in accuracy in the case of Resnet152 but slowing down training and inference speed. Comparison of different backbones of Faster R-CNN base on speed and accuracy is shown on Fig. 38 below.

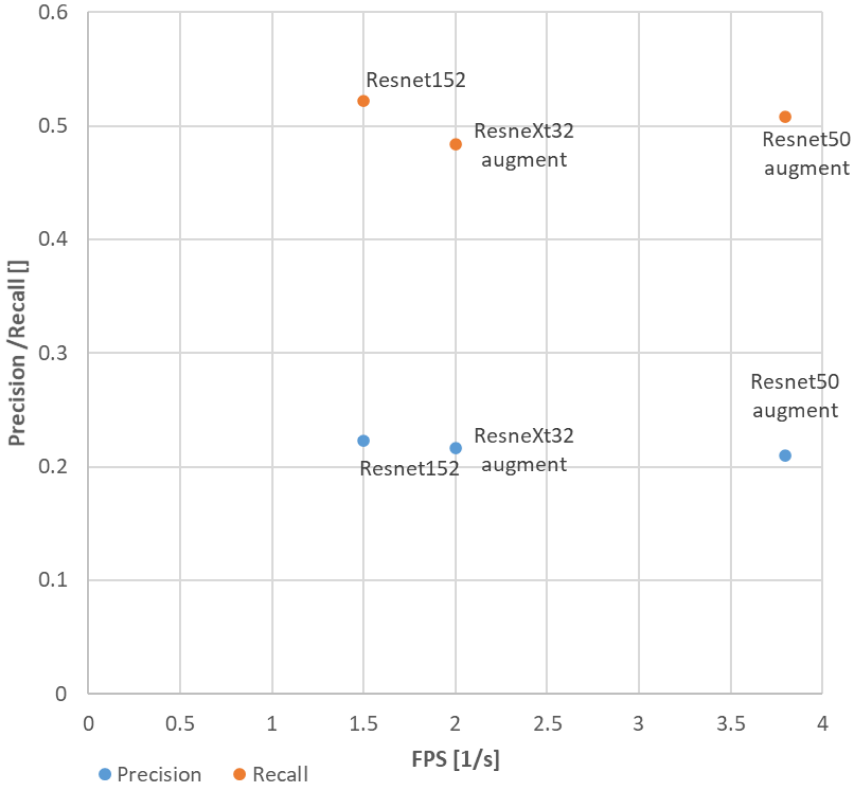

Figure 38: Precision and Recall versus speed in frames per second on GPU Tesla V100

7 SCALED-YOLO V4

YOLO is the most used and researched one-stage object detection system. The first versions used a simple version of Darknet as a backbone. However, the first versions were not good at predictions of small objects and small objects in the group. This was a case till YOLOv3 and YOLO v4. The latest advancements came with Scaled-YOLOv4. YOLOv4 was scaled to get better accuracy and speed. Thus, this detection system was the best candidate for our experiments [24].

Code for Scaled-YOLO models is provided by authors on the website.

7.1 General setup for all networks

Scaled-YOLO has a lot of parameters, which can improve training and final accuracy. The author of the thesis used original parameters provided by the authors of the Scaled YOLOv4. These parameters are already optimized for standard detection training and inference. In addition, there is only low space for additional experiments, because the training time of more accurate networks as Scaled-YOLO P6 is too high as you will see later in Tab. 5.

7.1.1 Training

Training of Scaled YOLOs was more stable and predictable than the training of Faster R-CNN, but take more time. The training was interrupted because of the limitation of Google Colab Pro. This caused a few strange deviations in losses (train part) and accuracy (test part) during training as is shown in Fig. 39 below. Deviations affect mainly Scaled-YOLOv4 P7 as training time was the biggest here.

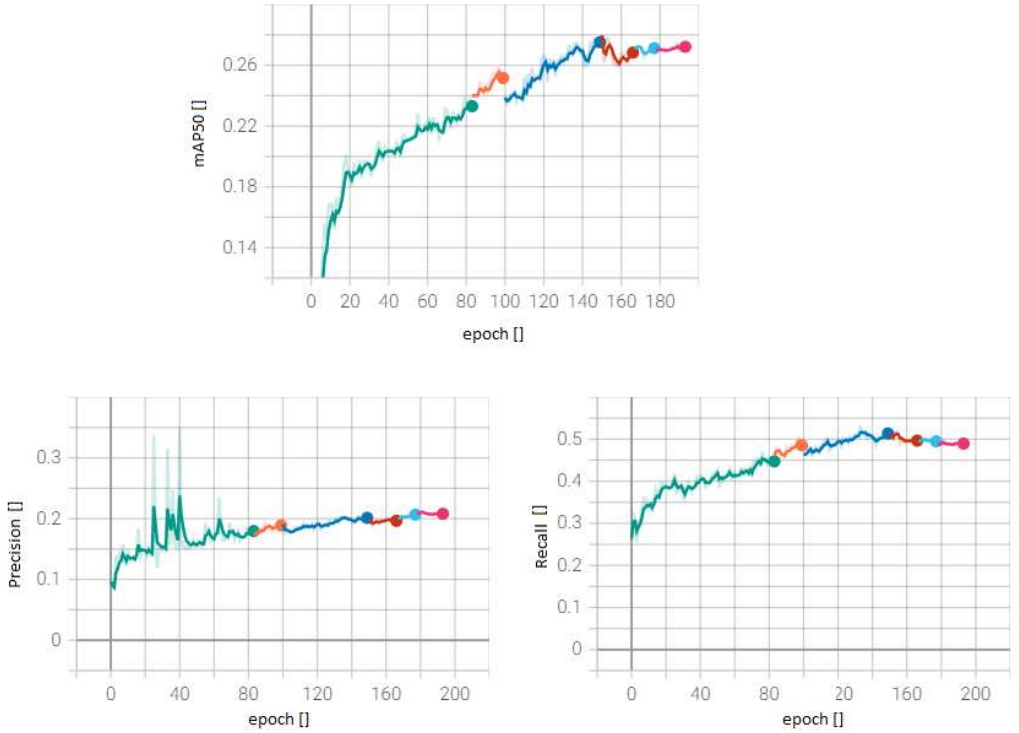

Figure 39: Average Precision, Precision, and Recall during training of Scaled YOLOv4 P6. Different colors represent different training started from the last saved point

7.1.2 Training setup

Training of Scaled-YOLOv4 networks is more time-consuming than the training of Faster R-CNN. Mainly due to the fact, that this YOLO needs more epoch to converge. For the batch size the author of the thesis used the same logic as for Faster R-CNN, batch size was chosen to possible maximum, based on maximum RAM capacity of GPU - 16 GB.

Table 5: Comparison of training of different Scaled YOLO.

Scaled-YOLO v4	Layers	Parame-ters	Image size	Batch size	Training time per epoch	Number of epochs	Overall training time
CSP	334	5.25e+07	608x608	4	10 min	100	17 hours
P6	417	1.26e+08	1216x1216	2	40 min	200	140 hours
P7	503	2.86e+08	1536x1536	1	90 min	150	225 hours

As you can see in the Tab. 5 above, training of P7 take 225 hours of GPU time and because of the limitation on Google Colab Pro, you should expect +50% more time needed for training. This is approximately 15 days of training.

7.1.3 Augmentations

The author of the thesis used standard Scaled-YOLOv4 augmentations, namely random flips, color augmentations, and mosaic augmentation from authors of Scaled-YOLO. All mentioned networks (CSP, P6, P7) were training with augmentation for comparison of these networks. Especially P6 was trained also without augmentation and with a combination of non-augmented and augmented training to see the impact of the augmentation.

7.2 Pest detection results

Scaled-YOLOv4 results confirm that bigger networks with augmentation have better accuracy, except P7. The accuracy of P7 is strange as it should have the best accuracy. A potential reason for this is low batch size 1 and a lot of interruptions of training by Google Colab Pro limitations, which caused already mentioned deviations. In addition, there was not time for additional experiments with P7 because of its big training time.

The results showed that P6 has the best accuracy and the impact of augmentation is approximately +10%. Scaled-YOLOv4 P6 Combo had similar accuracy as augmented P6 and both have the best accuracy in our comparison. Detailed results on the test dataset are presented in Tab. 6:

Table 6: The test set precision and recall results of different Scaled-YOLOv4 models and training setups (AP = Average Precision, AR = Average Recall, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci)

Detection system	**WF egg AP**	**WF egg AR**	**BT egg AP**	**BT egg AR**	**WF AP**	**WF Recall**	**BT AP**	**BT AR**	**All AP**	**All AR**
YOLO CSP augment	0.144	0.544	0.147	0.197	0.307	0.662	0.225	0.674	0.176	0.433
YOLO P6	0.160	0.552	0.181	0.235	0.327	0.651	0.248	0.611	0.192	0.458
YOLO P6 augment	**0.174**	**0.585**	**0.193**	**0.344**	**0.358**	**0.681**	**0.255**	**0.653**	**0.209**	**0.492**
YOLO P6 Combo	0.167	0.599	0.202	0.330	0.352	0.681	0.267	0.638	0.209	0.485
YOLO P7 augment	0.148	0.532	0.150	0.140	0.319	0.656	0.237	0.659	0.178	0.441

7.2.1 Accuracy versus speed

Scaled YOLOs are in general faster than Faster R-CNNs. In the case that the speed is the most important parameter CSP Augment should be chosen since it is 4.5x faster than P6. Visualization of Precision and Recall versus FPS is shown below in Fig. 40.

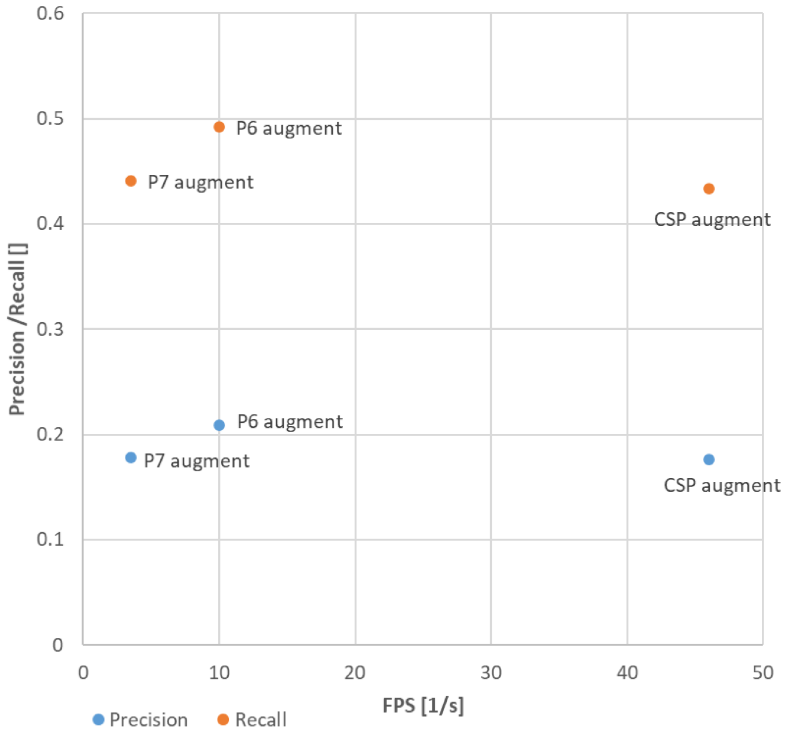

Figure 40: Precision and Recall versus speed in frames per second on GPU Tesla V100

8 OVERALL OVERVIEW

This chapter will summarize results from Faster R-CNN and Scaled-YOLOv4 training and compare one of the networks with results from the benchmark paper [33].

Only networks with the best results were chosen for this comparison. Results showed that networks in the final overview had different advantages and disadvantages. Faster R-CNN Resnet152 had the best detection results in terms of accuracy and Scaled-YOLOv4 CSP was best in terms of speed. Scaled-YOLOv4 networks were, in general, better at inference time and egg predictions and Faster R-CNNs were better at the prediction of BT and WF classes. Summarized results on the test dataset are presented in Tab. 7:

Table 7: Overall comparison of precision, recall, and FPS for the best networks (AP = Average Precision, AR = Average Recall, FPS = Frame per Second, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci, FPS = Frames per Second)

Detection system	WF egg AP	WF egg AR	BT egg AP	BT egg AR	WF AP	WF Recall	BT AP	BT AR	All AP	All AR	FPS [1/s]
Resnet50 augment	0.107	0.491	0.033	0.201	0.426	0.774	0.363	0.719	0.210	0.508	3,8
Resnet152	0.121	0.509	0.038	0.249	0.504	0.808	0.346	0.679	0.223	0.522	1.5
YOLO CSP augment	0.144	0.544	0.147	0.197	0.307	0.662	0.225	0.674	0.176	0.433	45
YOLO P6 augment	0.174	0.585	0.193	0.344	0.358	0.681	0.255	0.653	0.209	0.492	10

The results are based on the test dataset. But there was no additional Scaled-YOLOv4 adjustment for better prediction based on test data results. And there were only minimal adjustments for Faster R-CNN. Thus, the accuracy for the validation dataset would be similar.

8.1 Benchmark paper comparison

Object detection experiments were done with the dataset by authors of benchmark paper [33], as was already mentioned in the theoretical part of this work. But authors used a little bit different methodology for evaluations. Thus, the author of the thesis changed the confidence threshold to 0.6 for Scaled-YOLOv4 P6 to compare it.

Results showed that our Scaled YOLOv4 P6 had comparable or better accuracy and better results in terms of speed. The overview is shown in Tab. 8 below. Frame per second is the

only expected value for networks from the benchmark paper and was not part of my experiments.

Table 8: Comparison of my result of Scaled YOLOv4 P6 (bolt text) with adjusted prediction confidence threshold and results from benchmark paper. The stated frame per second values are expected values for each model; the authors of benchmark paper do not discuss the model speed (AP = Average Precision, AR = Average Recall, FPS = Frame per Second, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci, FPS = Frames per Second)

Detection system	**WF egg AP**	**BT egg AP**	**WF AP**	**BT AP**	**BT AR**	**FPS [1/s]**
Faster RCNN Resnet101 /w augmentation	0.54	0.16	0.72	0.34	0.200	2.5
SSD MOBILENET V2 /w augmentation	0.63	0.13	0.63	0.35	0.16	9
Faster RCNN Inception V2 /w augmentation	0.71	0.12	0.74	0.27	0.27	2.5
SSD INCEPTION V2	0.59	0.13	0.69	0.33	0.16	6
Scaled-YOLOv4 P6 Combo confidence threshold 0.6	**0.607**	**0.348**	**0.762**	**0.660**	**0.175**	**10**

9 DATASET IMPROVEMENTS

Dataset quality is a major prerequisite for proper training. As was mentioned in chapter 5, the dataset consists of images with wrong and missing labels. Thus, an experiment with an improved dataset was made. Only the worst 85 images (from 430) were deleted. The worst images consist of tomato class labels and missing pest labels. Detailed results of Scaled-YOLOv4 P6 for classes are shown in Tab. 9:

Table 9: Comparison of results on the test dataset and improved test dataset (AP = Average Precision, AR = Average Recall, FPS = Frame per Second, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci)

Detection system	WF egg AP	WF egg AR	BT egg AP	BT egg AR	WF AP	WF Recall	BT AP	BT AR	All AP	All AR
YOLO P6 augment	0.174	0.585	0.193	0.344	0.358	0.681	0.255	0.653	0.209	0.492
YOLO P6 augment on changed test dataset	0.186	0.590	0.193	0.338	0.383	0.677	0.290	0.649	0.291	0.560

The results confirmed that changes in the test dataset improved results. The biggest improvement is on BT and WF classes.

There are other improvements, which could be done and can significantly improve dataset and trained neural network. There are:

- Add missing labels for test and train dataset
- Change bounding box of actual labels

10 FUTURE WORK

In this chapter the author of the thesis shortly describes possible improvements of tested object detection systems and quality of dataset.

10.1 Scaled-YOLO v4

Scaled YOLOv4 P6 showed good results in this work. Thus, for future work, the author of the thesis recommends mainly more experiments with Scaled-YOLOv4 P6 and try Scaled-YOLOv4 P7 training with more GPUs with bigger RAM. The Bigger RAM memory will increase the advantage of training bigger batches with batch normalization and more GPUs will increase the speed of the training. With this hardware, the power of the bigger model Scaled-YOLOv4 P7 using a bigger image resolution over P6 can be seen. In addition, the author of the thesis recommends playing with hyperparameter (available from SCALED-YOLOv4 GIT), which optimizes anchor sizes and ratios base on the pest's dataset.

10.2 Faster R-CNN

As the result showed, the main problem of Faster R-CNN is the detection of small objects. Thus, the author of the thesis recommends trying to use a bigger image resolution (max size approx.1200), which will also need better hardware for better training as was mentioned before. In addition, the author of the thesis also recommends adding techniques, which improve detections of small objects. The main techniques are as follows:

- Replace IOU with GIOU - GIOU is better for predictions of small objects and widely used in recent state-of-the-art networks, including Scaled YOLO
- Soft Non-max suppression - Soft NMS is used to improve the accuracy of Faster R-CNN on small objects
- Improvements for better context recognition

In addition, the author of the thesis recommends trying other new state-of-the-art networks as Swin-L, Cascade Eff-B7 NAS-FPN, and others.

10.3 Dataset quality

Regarding the dataset quality, the author of the thesis recommends changing the wrong ground-truth bounding boxes and add missing ones. Then the training might be more stable and yield better overall results.

CONCLUSION

This research aimed to identify and test fast accurate object detection systems for pest detection on tomato plants. Based on a quantitative and qualitative analysis of models indicators, it can be concluded that Scaled YOLOv4 CSP Large is a good choice for fast pest detection and Faster R-CNN Resnet152 and Scaled YOLOv4 P6 Large is the best in terms of accuracy.

Scaled YOLOv4 P6 Large was compared with results of different models from benchmark paper. The results were similar in terms of accuracy and better in terms of speed. The outcome also indicates the impact of basic data augmentations is approximately a 10% accuracy increase.

Based on these conclusions, further research should focus on more advanced augmentations in the case of Faster R-CNN and try Scaled YOLOv4 P7 Large on better hardware. In addition, the author of the thesis recommends correcting wrong labeled ground-truth bounding boxes in the dataset for future work.

This work solved the pest detection problem with state-of-the-art object detection models and the results confirm the theoretical power of the models and the importantness of data augmentation.

BIBLIOGRAPHY

[1] LeCun, Yann & Bengio, Y. & Hinton, Geoffrey. (2015). Deep Learning. Nature. 521. 436-44. 10.1038/nature14539.

[2] Szu, Harold & Rogers, G.. (1992). Generalized McCullouch-Pitts neuron model with threshold dynamics. 535 - 540 vol.3. 10.1109/IJCNN.1992.227119.

[3] A. K. Palit and D. Popovic. Computational Intelligence in Time Series Forecasting : Theory and Engineering Applications. Advances in Industrial Control. Springer-Verlag London Limited, 2005.

[4] Robby Henkelmann: A Deep Learning based Approach for Automotive Spare Part Demand Forecasting Otto-von-Guericke-Universität Magdeburg, 2018.

[5] Cheney, Ward (2001). "The Chain Rule and Mean Value Theorems". Analysis for Applied Mathematics. New York: Springer. pp. 121–125. ISBN 0-387-95279-9.

[6] Rumelhart, G. Hinton, and R. Williams: Learning Representations by Back-propagating Errors D Nature 323 (6088): 533--536 (1986).

[7] Gluck, Mark A.; Mercado, Eduardo; Myers, Catherine E. (2011). Learning and memory: from the brain to behavior (2nd ed.). New York: Worth Publishers. p. 209. ISBN 9781429240147.

[8] PRINCE, Simon J. D. Computer vision: models, learning, and inference. New York: Cambridge University Press, 2012. ISBN 978-1-107-01179-3.

[9] The Neural Network Zoo - The Asimov Institute. The Asimov Institute [online]. [cit. 2021-05-10]. Available from: http://www.asimovinstitute.org/neural-network-zoo/

[10] CHOLLET, François. Deep learning with Python. 2018. Shelter Island, NY: Manning, 2018, xxi, 24 cm. ISBN 978-1-61729-443-3

[11] Applied Deep Learning - Part 4: Convolutional Neural Networks. Towards Data Science [online]. [cit. 2021-05-10]. Available from: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural- -networks-584bc134c1e2

[12] Jordan Moore. An Introduction to Sliding Window Algorithms [online] [cit. 16.05.2021]. Available from: https://levelup.gitconnected.com/an-introduction-to-sliding-window-algorithms-5533c4fe1cc7

[13] E.H. Andelson and C.H. Anderson and J.R. Bergen and P.J. Burt and J.M. Ogden. "Pyramid methods in image processing". 1984.

[14] Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. arXiv 2017, arXiv:1612.03144.

[15] Adrian Rosebrock, Deep Learning for Computer Vision with Python: Starter Bundle Autor. PyImageSearch, 2017

[16] GitHub - rafaelpadilla/Object-Detection-Metrics: Most popular metrics used to evaluate object detection algorithms.. GitHub: Where the world builds software · GitHub [online]. Copyright © 2021 GitHub, Inc. [cit. 16.05.2021]. Available from: https://github.com/rafaelpadilla/Object-Detection-Metrics

[17] Scikit-learn: Precision-Recall [online]. [cit. 2021-5-14]. Available from: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#precision-recall

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in International Conference on Learning Representations (ICLR), 2014.

[19] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, et al. PP-YOLO: An effective and efficient implementation of object detector. arXiv preprint arXiv:2007.12099, 2020.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015

[21] Alexey Bochkovskiy, Chien-Yao Wang, and HongYuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv preprint arXiv: 2012.07177 2021

[24] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, Barret Zoph. Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. arXiv preprint arXiv:2004.10934 2020

[25] Mingxing Tan, Ruoming Pang, Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. arXiv preprint arXiv:1911.09070 2020

[26] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, Xiaodan Song. SpineNet: Learning Scale-Permuted Backbone for Recognition and Localization. arXiv preprint arXiv:1912.05027 2020

[27] Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling Cross Stage Partial Network. arXiv preprint arXiv:2011.08036 2021

[28] Github: Scaled-YOLOv4 [online]. 2020 [cit. 2021-5-14]. Available from: https://github.com/WongKinYiu/ScaledYOLOv4

[29] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. arXiv preprint arXiv: arXiv:1911.11929 2019

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014

[31] R. Girshick, "Fast R-CNN," in IEEE International Conference on Computer Vision (ICCV), 2015.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv preprint arXiv: 1506.01497 2016

[33] Aitor Gutierrez, Ander Ansuategi, Loreto Susperregi, Carlos Tub-o, Ivan RankiT, Libor LenDa. A Benchmarking of Learning Strategies for Pest Detection and Identification on Tomato Plants for Autonomous Scouting Robots Using Internal Databases. Hindawi Journal of Sensors Volume 2019

[34] Albumentations: fast and flexible image augmentations. Albumentations: fast and flexible image augmentations [online]. Available from: https://albumentations.ai/

[35] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector. arXiv preprint arXiv:1512.02325 2016

[36] TorchVision Object Detection Finetuning Tutorial — PyTorch Tutorials 1.8.1+cu102 documentation. [online]. Copyright © Copyright 2021, PyTorch. [cit. 16.05.2021]. Dostupné z: https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

LIST OF ABBREVIATIONS

AP	Average Precision
AR	Average Recall
BT	Bemisia tabaci
CNN	Convolutional Neural Network
FC	Fully connected
mAP	mean Average Precision
ML	Machine Learning
WF	Trialeurodes vaporariorum
TA	Tuta absoluta

LIST OF FIGURES

Figure 1: Deep learning and machine learning as parts of artificial intelligence [1]..11
Figure 2: Simple neuron with input weights and one output [2] 12
Figure 3: Step transfer function, which represents a simple neuron.....12
Figure 4: The example of a complex network with input, output and hidden layers [3] 13
Figure 5: Schema of the neural network with output error.....13
Figure 6: Examples of over-fitting, under-fitting, and appropriate fitting [7].....15
Figure 7: Validation and training loss through the training.....15
Figure 8: Two objects with shifted blue parts. Convolution Neural Network (CNN) can recognize these objects as objects of the same class. 17
Figure 9: Example of CNN [9] 17
Figure 10: Input, CNN kernel, and output feature map [11] 18
Figure 11: Input and output after max pooling with 2x2 window with stride 2 [11] .18
Figure 12: Starting and ending position of sliding window technique. 19
Figure 13: Examples of two image pyramid methods. The left one with images at expanded scales and the right one with pyramids at reduced scales [13] 20
Figure 14: Visual comparison of the image pyramid and feature pyramid methods [14] 20
Figure 15: Example of predicted bounding box and ground-truth box 21
Figure 16: Visualization of Intersection over Union [15].....21
Figure 17: The example with more overlapping predictions for one object. Non-max suppression should be used to get the best prediction for the object 22
Figure 18: Example of overlapping objects of different aspect ratios and different scales 23
Figure 19: Precision x Recall curve for one class [17] 24
Figure 20: OverFeat detection system with information flow from input to output [18] 26
Figure 21: Visualization of localization of objects at different scales using Multi-Scale Classification [18] 26
Figure 22: The final prediction of the network [18] 27
Figure 23: Comparison of a few state-of-the-art networks based on speed and mean Average Precision on COCO benchmark dataset [19].....28

Figure 24: Three steps of the YOLO pipeline: resize the image, run the convolutional network, and non-max suppression [20] 29
Figure 25: An input image on the left, locations with predictions in the middle and final predictions in the right part of the example [20].....29
Figure 26: Detection network with 24 CNN layers and 2 FC layers [20] 30
Figure 27: Comparison of accuracy and latency for different YOLOs, SpineNet and EfficientDet (GPU V100 for batch size = 1) [27] 31
Figure 28: The simplest example of a CSP connection (on the left is vanilla ResneXt network, on the right is a CSP network with ResneXt) [29].....32
Figure 29: An example of a CSP connection in Scaled YOLOv4-CSP / P5 / P6 / P7 (on the left are reversed dark layers with SPP, on the right is reversed CSP dark layers with SPP) [29].....33
Figure 30: Main modules of R-CNN. Namely generator of region proposals, feature extractor, classification network, and bounding box regressor [30] 34
Figure 31: Visualization of modules of Fast R-CNN. Namely generator of region proposals, feature extractor, ROI Pooling layer, classification network, and bounding box regressor. 35
Figure 32: Comparison of mean Average Precision and Average Recall versus number of object proposals [31].....36
Figure 33: The latest incarnation of the R-CNN family, Faster R-CNN, introduces a Region Proposal Network (RPN) that bakes region proposal directly in the architecture, alleviating the need for the Selective Search algorithm [32] 37
Figure 34: The example of missing ground-truth bounding boxes.....41
Figure 35: The example of bigger ground-truth bounding boxes than real bounding boxes¨ 42
Figure 36: Example of bounding boxes with groups of pests.....42
Figure 37: Comparison of training losses with and without Cosines Annealing scheduler.....45
Figure 38: Precision and Recall versus speed in frames per second on GPU Tesla V100 47
Figure 39: Average Precision, Precision, and Recall during training of Scaled YOLOv4 P6. Different colors represent different training started from the last saved point.....49

Figure 40: Precision and Recall versus speed in frames per second on GPU Tesla V100 51

LIST OF TABLES

Table 1: Speed and accuracy comparison of large Scaled-YOLO networks on COCO benchmark dataset (mAPS = Average Precision on small objects, APM = Average Precision on medium objects, APL = Average Precision on large objects) [28].....32
Table 2: Comparison of Google Colab and Google Colab Pro 40
Table 3: Comparison of Faster R-CNN training with different backbones. 45
Table 4: The test set precision and recall results of Faster R-CNN models with different backbone networks and training setups (AP = Average Precision, AR = Average Recall, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci) 46
Table 5: Comparison of training of different Scaled YOLO. 49
Table 6: The test set precision and recall results of different Scaled-YOLOv4 models and training setups (AP = Average Precision, AR = Average Recall, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci) 50
Table 7: Overall comparison of precision, recall, and FPS for the best networks (AP = Average Precision, AR = Average Recall, FPS = Frame per Second, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci, FPS = Frames per Second).....52
Table 8: Comparison of my result of Scaled YOLOv4 P6 (bolt text) with adjusted prediction confidence threshold and results from benchmark paper. The stated frame per second values are expected values for each model; the authors of benchmark paper do not discuss the model speed (AP = Average Precision, AR = Average Recall, FPS = Frame per Second, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci, FPS = Frames per Second).....53
Table 9: Comparison of results on the test dataset and improved test dataset (AP = Average Precision, AR = Average Recall, FPS = Frame per Second, WF = pest Trialeurodes vaporariorum, BT = pest Bemisia tabaci) 54