UiA University of Agder

A Deep Learning-based approach for Fault Detection of Power Lines

CHRISTER MATHISEN

SUPERVISOR
Morten Goodwin

Master's Thesis
University of Agder, 2020
Faculty of Engineering and Science
Department of Information and
Communication Technology

UiA
University of Agder
Master's thesis

Faculty of Engineering and Science
Department of Information and
Communication Technology

Abstract

A transmission network is the most crucial part of modern infrastructure. However, it requires an extensive amount of power line inspection each year to maintain, and with an increased interest in replacing large helicopters with drones for this process, the possibility of including AI is equally compelling. This thesis goes into the second part by taking a deep learning-based approach in the interest of fault detection. A literature review illustrates that earlier research has some to none understanding of the complexity required for inspection.

Due to the advancement in object detection and classification, this thesis has identified and implemented an applicable model capable of giving state-of-the-art accuracy in electrical pole and component detection by dividing the process into multiple layers. This thesis takes as well and proposes a new method that presented great result in assuring more reliable fault detection and is a way to improve the quality of images taken by drones. The pole detection layer gave 97.7 mAP, the component detection layer reached 95.6 mAP, the fault classifier delivered an accuracy of 93%, and the proposed quality classifier had an accuracy of 93% as well.

The presented approach illustrates the possibility of phasing the physical inspection out. The amount of component labeled that must be available for algorithmic training to surpass a human expert is not readily available. Nevertheless, the presented approach is a sufficient tool for assisting the inspector.

Preface

This thesis wraps up my final work for the master program information- and communication technology at the University of Agder. The quality of this thesis would never have been possible without my supervisor Morten Goodwin.

The motivation for this project was brought to light by Agder Energy. Therefore I would give a huge thanks to the company for allowing me to work with this exciting problem, and I hope they find my contribution rewarding. There are a few people in the company I want to give my gratitude, Per-Oddvar Osland and Tobias Haumann, for providing constructive criticism on my approaches, and Jarle Stokke-Olsen and Solveig Fjellbakk for providing me data and support regarding questions I had about the inspection process.

The work presented in this paper is not altered in any way to highlight the University of Agder or Agder Energy more than what is factual.

After using a significant amount of my life in school, it is a bit far-fetched that this is most likely the last work I deliver. However, the journey has been far, and there are many people that I should have acknowledged, but I believe in my heart they know who they are. Thank you all.

Christer Mathisen
June 04, 2020

Table of Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 4, 46

AP Average Precision. 11, 27

CNN Convolutional Neural Network. 6, 7, 12, 14, 18, 21, 26

COIL Columbia Object Image Library. 8

CSPNet Cross Stage Partial Network. 22

FC Fully-Connected. 16, 58

FN False Negative. 9, 51, A2

FP False Positive. 9, 51, A2

FPS Frames Per Second. 20, 21, 23, 27

GAN Generative Adversarial Network. 26

HOG Histograms of Oriented Gradients. 17, 18, 26

IBM International Business Machines. 12

IoU Intersection-Over-Union. 9, 11, 20, A2

mAP Mean Average Precision. 11, 18, 21, 22, 29, 59, A2

MNIST Modified National Institute of Standards and Technology. 8, 12

MS COCO Microsoft Common Objects in Context. 9, 11, 12, 22, 23, 41, 42, 61

PAN Path Aggregation Network. 23

R-CNN Region-based Convolutional Neural Networks. 20–23, 27, 39

ReLU Rectified Linear Unit. 14, 58

RoI Region of Interest. 9, 21, 26, 28, 30, 35, 40, 43

RPN Region Proposal Networ. 21

SIFT Scale Invariant Feature Transform. 17

SPP Spatial Pyramid Pooling. 23

SRCNN Super-Resolution Convolutional Neural Networks. 29, 39

SVM Multiclass Support Vector Machine. 16, 21

TN True Negative. 9

TP True Positive. 9, 51, A2

UAV Unmanned Aerial Vehicle. 4, 27–30, 36, 41, 45, 61, 62

VOC Visual Object Classes. 8, 11, 12, 21, 22

YOLO You Only Look Once. 20, 22, 29, 38, 39, 41, 42, 45, 46

Part I

Research Overview

Chapter 1

Introduction

Electricity has become one of the most necessary dependencies in modern society. This comes with its own challenges related to reliability, availability, and sustainability. Even small blackouts can be costly, and here the importance of power line inspection comes in – but the size of the Norwegian power lines is immense. In addition to volume, the electrical network is divided into three transmission levels covering over 130000 kilometers with power lines. Because of these levels, and the lack of early planning in regards to standards, the transmission network is highly complex.

Power line inspections are mostly done with a low-flying helicopter where at least two people are required – one to take multiple high-resolution photos of each electrical pole and one maneuvering the helicopter in the inspection direction. This process is tedious and puts human lives in a dangerous situation. An inspection done in 2018 was extremely close to becoming fatal, where a helicopter almost collided with a line crossing over the inspected line. Nevertheless, it is not only human lives that are continuously at risk. There have been situations where farm- [1] and wildlife animals [2][3] have died because of the stressful situation the helicopters put upon them. These cases illustrate the beneficial needs of replacing large helicopters with drones in regard to human- and animal safety.

1.1 Motivation

The situation presented has been challenged by different companies that have observed an opportunity to use an Unmanned Aerial Vehicle (UAV) and Artificial Intelligence (AI) together - permanently removing humans from the equation. While waiting for drone technology to improve, this thesis' motivation is to take a deep learning-based approach in interest to automated fault detection and assuring that images contain enough information to assure a proper inspection.

1.2 Thesis Definition

The primary purpose of this thesis is to take a deep learning-based approach to support power line inspection. To be able to propose a possible solution for out-phasing the potential slow time power line inspectors use, the research is divided into three goals and three hypotheses.

1.2.1 Thesis Goals

Goal 1: *Create or locate a dataset suitable for taking a deep learning-based approach for power line fault detection.*

Goal 2: *Identify and implement an applicable model capable of locating electrical poles and components under complex conditions and provide state-of-the-art accuracy.*

Goal 3: *Identify and implement an applicable model suitable to classifying components given from goal two in regards to fault detection and quality assurance.*

1.2.2 Hypotheses

Hypothesis 1: *The quality of an image can not entirely be represented based on the number of pixels wrapping a component.*

Hypothesis 2: *State-of-the-art deep-learning methods for image recognition are more suitable than earlier presented methods.*

Hypothesis 3: *A Deep Learning-based approach for fault detection is capable of phasing out an inspection performed by an expert so long the dataset contains enough information.*

1.3 Contributions

This thesis takes state-of-the-art methods and uses it in an area that has had only a hand-full of earlier experiments. With high variation in contributed results and questionable accuracy from earlier projects, this thesis has shown an improvement in detection by using methods not seen documented in power line inspection before.

In addition to better accuracy, a new method for assuring reliable image data from an inspection is proposed. From the authors' knowledge, no documented research has proposed or tested a similar method.

1.4 Thesis Outline

Chapter 2 outlines background theory relevant for the experimentation and research, by presenting standards in the field of object recognition (2.1), the fundamental architecture of CNN (2.2), and the change in object recognition (2.3).

Chapter 3 investigates current state-of-the-art in object detection and classification (3.1), and what earlier research in the area of power line inspection has accomplished (3.2).

Chapter 4 presents information in regards to the inspection data (4.1), brief overview of the labeling tool created (4.2), and a description of datasets created (4.3).

Chapter 5 introduces the proposed solution for the presented goals in (1.2), by describing the methods for object detection (5.2) and object classification (5.3). Ending the chapter with a outline of the full design (5.1).

Chapter 6 reveals and discusses the results from the experiments done in regards to the proposed solution in Chapter 5.

Chapter 7 concludes this thesis and presents possible future research.

Chapter 2

Background

Neural network is a common term used within machine learning for algorithms that mimics the way the human brain operates. Since this topic has become a wast sea of solutions and methods during the last few years, will only the most relevant network for this thesis be briefly described, namely Convolutional Neural Network (CNN).

This chapter outlines background theory relevant for the experimentation and research presented in this thesis. Section 2.1 describe standards used for measuring the quality of methods. Section 2.2 presents the fundamental architecture of CNN. Section 2.3 outlines the change in object recognition from traditional methods to neural networks.

2.1 Standards – Datasets and Formulas

In most sports where people compete, there is a set of rules everyone follows to ensure fairness. Just as in the Olympics, scientists and engineers in the field of computer vision have established standards to provide a common ground for everyone. These standards can be related to the dataset used or the definition of the mathematical statements within scientific papers or journals.

2.1.1 Datasets – computer vision and classification

To be able to determine detection and classification accuracy to find the best model, various official datasets have been put together to achieve common ground for measurement. Early datasets created for image classification contained a single image and a blank background, Modified National Institute of Standards and Technology (MNIST) [4] and Columbia Object Image Library (COIL) [5] are examples of this kind of official datasets. The transition to more realistic images came with the Caltech-101, and fast followed by Visual Object Classes (VOC) 2005 [6].

An explosion of data came with the rise of the digital age, with social media and more available technology for the average consumer increased access to information. Because of this availability, Pascal VOC went from only four classes and 1578 images in 2005 to 20 classes and 11530 images in 2011[1] [8]. The number of categories in Pascal VOC did not satisfy a team of scientists at Princeton University, which announced in 2010 ImageNet [9], a large-scale database of images divided into 1000 classes and 3.2 million images. ImageNet is extremely diverse in categories. Because of this, the number of images within each category had to be limited to an amount between 500 and 1000 images.

One of the primary goals of computer vision is to get an understanding of the content, what objects are present, and the location of each object. To achieve this goal, images should contain complexity in the number of categories presented. ImageNet and Pascal VOC have a common flaw regarding this; they have not focused on category complexity. Over 60% of their data only contain one category [10]. Because of this lack in diversity,

[1] "The 2012 dataset is the same as the 2011 dataset"[7]

a new dataset was created, named Microsoft Common Objects in Context (MS COCO) [10]. MS COCO is designed to advance the state-of-the-art within object recognition and is today the most used dataset benchmark in its field. The dataset contains 91 objects distributed across 328 thousand images, where only 10% of the images contains only one category.

2.1.2 Formulas – mathematical statements

Jaccard Index or better known as **Intersection-Over-Union (IoU)**, is an evaluation metric used to determine the accuracy of an object detector on a ground-truth bounding box (i.e., a hand-drawn box determine the location of a given object), most often defined as **Region of Interest (RoI)**. As seen in formula 2.1, two values are required to be able to decide the IoU, a ground-truth bounding box which is most often quality assured by a human and a predicted bounding box which is the box provided by the vision-based algorithm. The IoU result given is a value between 0 and 1, where 1 is a perfect overlap, and 0 is no contact.

$$IOU = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cup B|} \tag{2.1}$$

To find out how well an object detection model is doing, the result given back from a model is divided into different values based on the prediction made related to the ground-truth:

- **True Positive (TP):** A detection is satisfied with the IoU threshold and class type.
- **False Positive (FP):** A detection is not satisfied given requirements related to class or IoU.
- **False Negative (FN):** A prediction was not made where an object was present.
- **True Negative (TN):** does not apply in object detection, because that would be all possible boxes that were never given by the model.

Figure 2.1 gives a more visual view of what the different values represented. It should be stated that the IoU threshold between TP and FN is not a globally defined value, but there exist some defined values most scientists follow.

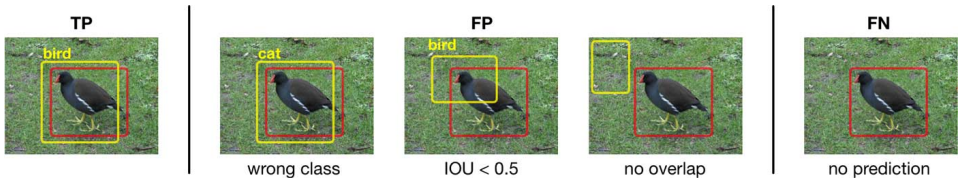


Figure 2.1: A visual illustration on how object detection determine the result between true positive, false positive, and false negative.

Precision determine how many of the predicted classes with a boundary box are relevant – also known as positive predictive value. Seen in formula 2.2 is the true positive values divided on the total amount of prediction done. The result given is a value between 0 and 1.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ detection} \tag{2.2}$$

Recall, on the other hand, determines how many of the selected items are relevant. In equation 2.3 is the true positive values divided on the total amount of relevant occurrences. The result given is a value between 0 and 1, where 1 have no failed predictions, while 0 is a situation where there was never a good enough prediction.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \tag{2.3}$$

F_1 score is a measure used to test the accuracy of a model. The score can be interpreted as a weighted average of the recall and precision, a harmonic mean between the values. Just as recall and precision, the score given by the $F_1 score$ is a value between 0 and 1. The $F_1 score$ is a better messure when seeking the balance between recall and precision and in datasets where classes have a uneven amount of object [11].

$$F_1 score = \left(\frac{2}{recall^{-1} + precision^{-1}} \right) = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.4}$$

Average Precision (AP) is perhaps the main measurement value. AP combines multiple values given and calculate an averaged value. Before the calculation is done, the pattern given is smooted to the lowest and highest value, as seen in equation 2.5. Since precision and recall always give a value between 0 and 1, AP will as well fall between this range.

$$AP = \int_0^1 p(r)dr \tag{2.5}$$

IoU and AP operate hand-in-hand when it comes to object detection. Looking away from class difference and situations with no predictions, AP always have an IoU-threshold that defines if something is true positive or false positive. The most common measurement used is IoU larger than 0.5, commonly written as $AP^{IoU=.50}$, AP_{50}, or AP@[.50]. This threshold has been considered *good enough* for object detection and was used a lot in the VOC challenges. $AP^{IoU=.75}$ or AP_{75} is called strict metrics and is as the number states a stricter requirement before something is defined as true positive. The MS COCO challenge has taken this measurement even further and takes the average over multiple different IoU threshold between 0.5 and 0.95, where the value increases with 0.05 for each step up to 0.95 (i.e., teen steps in total). This can as well be written as $AP^{IoU=.50:.05:.95}$ and a few other ways already mentioned. In some cases stands AP alone with no defined threshold mentioned. There is no globally established value of what AP-threshold is when standing alone but can, in most cases, be determined based on what type of dataset used when determining the value (i.e., VOC, or MS COCO).

Mean Average Precision (mAP) is, in most cases, the average value of the computed AP for each class, but it is important to note that this might not be the case. For example, in the context of MS COCO, there is no distinction between mAP and AP [12]. When there is a difference between the values, mAP calculation is seen in equation 2.6, here K is the number of classes and AP_i is the AP of a given class i.

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{2.6}$$

2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) or also known as ConvNet, was first introduced by IBM in the year 1998 [13], and have similarities to standard neural networks with learnable weights and biases. What separates ConvNet architecture from standard neural networks is the explicit presumption that everything that comes into the input-layer is images; Allows ConvNet to have properties within the architecture that specializes in visual-recognition [14].

There are five types of layers used together in a *stack* to form a full ConvNet architecture. A not too technical description will be presented to give a overview of role of the different layers.

Input Layer

The input layer is the first building-block for all types of networks, defining a large part of the architecture based on the dimensional shape allowed into the model. The shape is not universally defined and depends on the situation desired. Standard datasets described in section 2.1.1, such as MS COCO and Pascal VOC, have high complexity images and require a larger input dimension even to get a perception of the objects, while MNIST, on the other hand, can work with less input information because of low image complexity.

Convolutinal Layer

The convolutional layer is the fundemental building-block for ConvNet, in this layer most of the heavy computational workload is preformed. The amount of workload is determined by multiple factors such as the input volume and the values of different hyperparameters:

1. **Depth (K)** represents the number of filters, and this value most often starts at the input layer with an amount of three because of the color spectrum computers are built toward. Still, in some cases, the value might be preferable as one (i.e., gray-images) for efficiency reasons.

Each hidden layer within the network has the potentiality to reduce image width and height, for the benefit of increasing the depth-value of the given data.

2. **stride (S)** determines how far the filter moves for each iteration. A stride value of one will move the filter by one pixel in the horizontal direction until reaching the end of the width-axis, before moving a stride value down and starting the window from left again. More substantial stride values will produce smaller output volume, but it is uncommonly to see a usage of stride value higher than two.

3. **Zero-padding (P)** is a technique where values of zero's are added around the border of an image to control the size of the output. This is a common practice when we want to preserve the spatial size of the image (i.e., the same width and height in the output layer as in the input layer).

4. **Spatial extend (F)** is a value used to determine the receptive field of neurons. This is a considerable technique used when dealing with high-dimensional inputs such as large images to reduce the number of connections each neuron requires between layers. The width and height dimension can differ, but the depth-dimension is always equal to the depth of the input value from the previous layer.

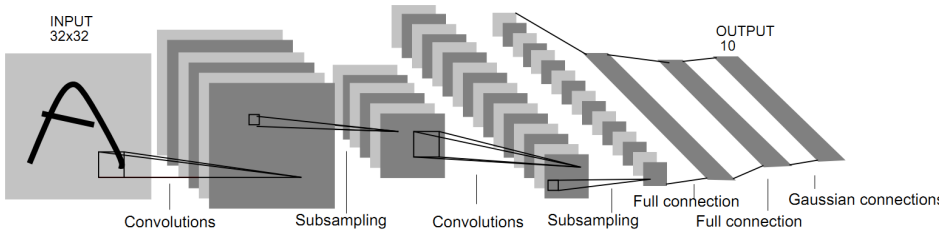


Figure 2.2: Architecture of the first convolutional neural network, LeNet-5 [13].

The convolutional layer combines the four hyperparameters and use them to determine the values of the next layer. By accepting an input volume of size $W_1 \times H_1 \times D_1$, the hyperparameters produces a output value by:

- $W_2 = \frac{W_1 - F + 2P}{S} + 1$
- $H_2 = \frac{H_1 - F + 2P}{S} + 1$
- $D_2 = K$

Figure 2.2 illustrates how the original image is transformed and given more depth through the hidden convectional layers, the architecture is from LeNet-5, which only contained five hidden layers, but the concept still applies for more modern approaches.

ReLU layer

Activation functions are not often explicitly written as a layer. However, to get a modest understanding of a CNN it is explained with a focus around ReLU activation. Rectified Linear Unit or better known as ReLU, it is a type of activation function most often found in neural networks. ReLU is linear for all positive values, where negative values become transferred to zero, defined as $\boldsymbol{y = max(0, x)}$. Nevertheless, turning all negative values to zero have a downside. A concept called *Dying ReLU* occurs if the inputs are caught at the negative side, then all outputs become zero, and once a neuron gets negative, the chance for recovery is unlikely. To counter the *dying ReLU* other variants of ReLU have been used, such as Leaky ReLU $\boldsymbol{y = max(0.1x, x)}$ or Paramteric ReLU $\boldsymbol{y = ax}$ where $\boldsymbol{a}$ is a constant value the system determine by itself when $\boldsymbol{x < 0}$.

Looking away from the problem with *Dying ReLU*, the concept of ReLU is inexpensive to compute, allowing the model to train and converges faster. It is sparsely activated since zero replacing all negative inputs which are most often desirable when working with multiple classes since the model filter away unnecessary information.

Pooling Layer

The pooling layer is a common periodically practice to progressively reduce the number of parameters and required computations, by downsampling the dimension of an input array. Pooling works by taking in a spatial extend $\boldsymbol{F}$ and a stride value $\boldsymbol{S}$. The most common form of pooling is $\boldsymbol{F} = \mathbf{2}$, and $\boldsymbol{S} = \mathbf{2}$, but $\boldsymbol{F} = \mathbf{3}$, and $\boldsymbol{S} = \mathbf{2}$ is also seen, larger dimensions are considered too destructive.

There exist different ways of preforming pooling; max pooling is the most commonly used. This practice takes the $\boldsymbol{F}$ box and only keep the highest value, before striding to the next window. Other methods exist but are not commonly used, such as average pooling and minimum pooling. Discard of the entire pooling concept has occurred, and some scientists [15] only favor a well balanced convolutional layer with variations in hyperparameters above the idea of pooling.

Fully-Connected Layer

Fully-Connected (FC) layer is as the name state, a layer where all neurons are connected fully to all activations from the previous layer. This differs a bit from what was earlier mentioned related to the convolutional layer, where the connections are within a receptive field. In regards to differences between the two layers, it is only the amount of links that differ, where FC has a global connection, while convolutional has locally based connections.

The FC layer is not only a sub-layer used within a network, it is as well the last layer within the architecture. This layer uses a preferable loss function to calculate a class score based on the provided input image. There are different functions available, but the most common ones are:

- **Multiclass Support Vector Machine (SVM)** loss is one of two commonly seen classifiers. The SVM classification seen in equation 2.7 wants a score higher than the incorrect classes by a fixed value greater than Δ (delta) to achieve a loss value of zero. When SVM reaches the defined delta value, there is no need for further improving; the system only cares about reaching the delta difference.

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta) \tag{2.7}$$

- **Softmax** is a binary logistic regression classifier, used for generalizing the score of multiple classes. The Softmax takes a vector of arbitrary real-values scores and normalizes the value to a value between zero and one, where the sum of all categories equals to one.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.8}$$

2.3 Object Recognition

Recognizing something is a way to gain a level of understanding of what the visual perception is seeing. This is something that comes naturally to humans. However, for a machine, this is entirely different and most often requires a lot of visual information.

Object recognition is a general term used to describe a collection of related computer vision tasks used in computer vision, such as classification, localization, and detection.

Recognizing different objects requires some sort of feature extraction of the visual traits, which can present a semantic and robust representation [16]. Methods used for this type of extraction is diverse, but there are a few representative ones:

- **Scale Invariant Feature Transform (SIFT) [17]**, is an approach which transforms image data into scale-invariant coordinates relative to local features. The method first initializes key points based on the maxima and minima of the different-of-Gaussian functions, then a threshold of minimum contrast is applied for reducing the number of keypoints, before finally using a threshold on a ratio of principal curvatures. An essential aspect of this approach is the vast quantity of generated features that densely cover the image.
- **Histograms of Oriented Gradients (HOG) [18]** is reminiscent of SIFT, but instead of dense grids of uniformly spaced cells and feature overlapping, HOG uses a grid of overlapping blocks. HOG first normalize the gamma and color value of the input image, before dividing the image into small connected regions and computes a histogram of gradient directions or edge orientation for pixels inside the cell. The pixels then calculate a weighted vote for an edge orientation based on the gradient element concentrated on the pixel position, the votes get accumulated into orientation bins over a local cell, then the overlapping cells are locally contrast-normalized to increase performance. Ending the HOG method by adding a detector window to the image, creating a black border around the image, which has shown a significant amount of increase in context.

To recognize different objects with this type of feature extractions, a large

number of features have been pre-extracted from similar images and added to a *feature-database*. This database provides a basis for object and scene recognition. After the increased interest in neural networks, primarily CNN, these types of feature extractions have become antiquated because the obtainable mAP is not comparable. Ross Girshick et al. [19] illustrated this jump in mAP with the introduction of R-CNN in 2013 – the method had a 61% relative improvement compared to a HOG based method.

In a CNN architecture, feature extraction is conducted in the backbone network – this network is acting as the primary feature extractor for an object detection task. The network takes an input image of a defined size and outputs a feature map of the corresponding image. The last layers use this feature map to classify it to a predicted class. [20]

Chapter 3

State-of-the-art

The inspection of power lines is a critical task to ensure stability in crucial infrastructure. With the advances in object detection and image classification in the last few years, state-of-the-art methods have quickly advanced, opening up the possibility to outsource repetitive and tedious inspection.

This chapter will present well-established state-of-the-art methods used in object detection and image classification in Section 3.1, before moving over to a literature review in the field of visual power line inspection in Section 3.2.

3.1 Object Recognition

3.1.1 Object detection

Defining the location of an object within an image is the main problem defining object detection. The pipeline used to achieve this is mainly divided into three stages: informative region selection, feature extraction, and classification [16]. The object detection systems taking advantages of this pipeline can usually be divided into two categories:

- **Two-stage Deep Object Detection** splits the pipeline into a region-proposal-stage focusing on object localization, extract the features and then moves over to the next stage which is classification. This method have delivered some of the best results in terms of object recognition accuracy with methods such as Fast R-CNN [21] and Faster R-CNN [22].

- **One-stage Deep Object Detection** does not split the pipeline into two-stages allowing much higher computational efficiency. YOLO [23] is a well known one-stage object detector that illustrates higher computational efficiency in the form of numbers of boundary boxes required and Frames Per Second (FPS) of detection time, allowing real-time object detection. The amount of FPS depends on the computational power and the backbone of the model, e.g., YOLOv3 [24] introduced three methods where there was a tradeoff between execution time and FPS in regards to localization accuracy.

Except for the number of stages in the categories, both models wants to increase the localization accuracy to provide the best object detector. The most common measurement for determining accuracy is IoU, which can work as a loss function for faster convergence and better accuracy compared to L2-loss [25]. But the tradeoff between two- and one-stage detecting have always been in the form of accuracy and speed until Alexey Bochovskity et al. [26] introduced version four of YOLO.

3.1.2 Two-stage detector – R-CNN

Object detection concept had reached their peak on modern datasets. So in 2013, Ross Girshick et al. [19] introduced Region-based Convolutional Neural Networks (R-CNN), a two-stage object detector with a mAP 7.1% higher than the previous best result on the 200-class ILSVRC2013 detection dataset. The architecture of R-CNN is first based on the selective search algorithm from [27] for object extraction, proposing approximately 2000 possible regions, then running each proposed region through a CNN, before classifying each image with SVM and determine the best fitted bounding box out of the proposed regions.

R-CNN needed days to train, required hundreds of gigabytes of storage for VOC 2007 features, and used around 47 seconds to propose objects on each test case. This was not suitable for a real use case. Hence, the introduction of Fast R-CNN [21], built on the previous work. Fast R-CNN is 9x faster in training, 146x faster in test-time, and achieves higher mAP than R-CNN. This was done by changing the method of processing from each RoI, to the entire image. It still proposes RoI, but overlapping regions are sharing computation – which is a significant change for improving the speed. The CNN is still built on the VGG16 architecture. However, the classifier is changed from SVM to softmax because of a slight improvement in mAP.

The test speed of Fast R-CNN was still considered too slow for an object detector. As a consequence of this, Faster R-CNN [22] became introduced. The architecture for detection is the same as in the previous model. However, the selective search algorithm became replaced with an integrated Region Proposal Networ (RPN). RPN is a sliding-window concept, moving n pixels at a time, and testing k number of anchors at each sliding position. This small change in the first stage of the model increased the test speed 10x compared to Fast R-CNN, reducing the test time at each image to 0.2 seconds; 5 FPS detection speed.

3.1.3 One-stage detector – YOLO

To accomplish real-time image processing, complex methods such as sliding window or generating potential boundary boxes had to be eliminated to reduce computation power. By looking at the problem as a single regression problem, YOLO can predict what objects are present and the location of the given object by only looking once, thereby the name *You only look once* [23]. YOLO divides an Image into a SxS grid, where each cell in the grid predicts B bounding boxes, confidence for those boxes, and the class probability. If an object is within a cell, that cell is responsible for detecting the object, or the confidence for the object existence should be zero.

YOLO had a variety of shortcomings relative to state-of-the-art detection systems. With a significant number of localization errors, and relatively low recall compared to region proposal-based methods. Because of this an improved model was designed, YOLOv2 [28]. YOLOv2 takes a variety of concepts to improve the older model, e.g., batch normalization, higher resolution classification, dimension clusters, direct location prediction, fine-grained features, and multi-scale training. The exact improvement each technique has on the model can be found in [28], although the total improvement was significant and put YOLOv2 in line with other state-of-the-art detectors such as Faster R-CNN. At a high resolution (i.e., 480x480, 544x544) YOLOv2 provided state-of-the-art detection on the Pascal VOC 2007+2012 dataset with a 78.6 mAP. Even with impressiv result at Pascal VOC, the methode was lagging behind on the latest developed standard dataset (i.e., MS COCO). To challange this YOLOv3 was introduced. According to Joseph Redmon et al. [24] there was not done any significant improvement to the method, but even so, they mananged to improve AP_{50} with 31.6% on the MS COCO set, passing Faster R-CNN. Even with this strong detection metric, it was not able to surpass RatinaNet in any form of defined metric.

Two years later, on April 23.2020 Alexey Bochkovskity et al. [26] introduces YOLOv4. The model is using the concept from YOLOv3 in front for class prediction and object localization, and have taken an advanced concept from Chien-Yao Wang et al. [29] named Cross Stage Partial Network (CSPNet) which have been implemented with the DarkNet53 backbone to reduce not only computation cost and memory but also increasing speed and accuracy. Between YOLOv3 input-based images and the CSPDarkNet53 backbone, they have added two image processing concept named

Spatial Pyramid Pooling (SPP) [30] and Path Aggregation Network (PAN) [31]; the methods are built for improving feature extraction. The combination of these methods, with a few smaller ones, have made YOLOv4 the best available state-of-the-art detector in regards to real-time[1] accuracy on the MS COCO dataset.

3.1.4 Image classification

Teaching machines to find meaning within an image have existed for decades, already in 1998 the first CNN architecture was proposed by Yann LeCun. The architecture called LeNet-5 [4] was designed to recognition handwritten and machine-printed characters. However, the concept of using neural networks for image classification was almost forgotten because of computer power, data availability, and the lack of notice within the science community.

In 2012 a neural network known as AlexNet [32] entered the ImageNet competition and vastly outpaced the competitions. The result started a new age for computer vision; someone might consider that history began with AlexNet [33]. Two years later, in 2014, the Visual Geometry Group (VGG) network was introduced [34] – this is the same backbone the classification part of R-CNN is using. Different classification methods have been proposed afterward, although the most significant is EfficentNet.

3.1.5 EfficientNet

Network scaling is never straight forward, and the most common ways for scaling have been in one of three directions, as seen in Figure 3.1: width (b), depth (c), or image resolution (d). Mingxing Tan and Quoc V. Le [35] wanted to rethink the scaling process and looked into the effect of each component interacting with each other, finding a balance. They state that ConvNets have become increasingly more accurate by going more prominent in the number of parameters, but the size of state-of-the-art networks such as GPipe is so massive it can only be trained with a specialized pipeline. Nevertheless, they reveal that carefully balanced network parameters perform better in accuracy than earlier state-of-the-art methods [35, Tab. 2].

[1] Speed faster than 30 Frames Per Second

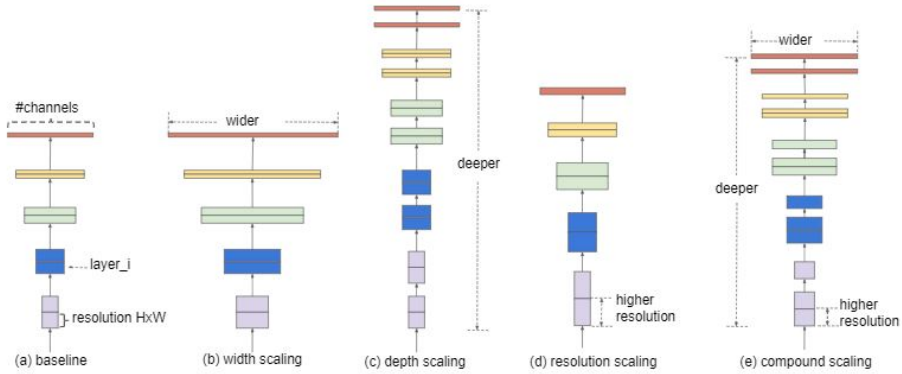


Figure 3.1: (b)-(d) are different methods for neural network scaling, (e) is the proposed solution for EfficentNet [35, Fig. 2]

The proposed parameter scaling method uses a compound coefficient ϕ on a fixed baseline network named EfficentNet-B0. The scaling of the baseline network is based on equation 3.1, where ${\alpha_1}^2$, β^3, γ are fixed constraints determined by a grid search.

$$\begin{aligned} depth &: d = \alpha_1^{\phi} \\ width &: w = \beta^{\phi} \\ resolution &: r = \gamma^{\phi} \\ constraint &: \alpha_1 \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ restriction &: \alpha_1 \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3.1}$$

Multiple observations of different scaling dimensions have shown that an increase in image resolution should also have an increase in depth to capture similar features and in width to capture more fine-grained patterns. With a small grid search based on equation [35, eq. (2)] and 3.1, have the authors found the best values for the baseline network to be: $\alpha_1 = 1.2$, $\beta = 1.1$, and $\gamma = 1.15$. Only a change in the ϕ is done to obtain EfficentNet-B1 to -B7[4].

EfficentNet is a highly effective compound scaling method to a suitable net-

[2]Have a subscript of one to not get misled with the learning-rate value α
[3]The exponential decay rate, uses an index of two β_2, and is not the same as β
[4]EfficentNet-B8, is later introduced as a larger alternative [36]

work for a given task. Additional work has improved the EfficentNet model further, such as using adversarial examples for improving image recognition [36], semi-supervised learning for training a second model [37], and fixing the train-test resolution [38]. An increase in accuracy at ImageNet top-1 is approximately 4.1% (from 84.4% to 88.5%) in difference between the original EfficentNet-B7 [35] and the best improved solution [38].

3.2 Literature Review

Powergrid inspection has not been a topic for more than six to seven years – the lack of available data and promising methods are two main factors. There has been done research in transmission quality [39][40][41] line detection [42][43][44] and the advances of Generative Adversarial Network (GAN) have illustrated the possibilities for segmenting out components [45]. However, these types of inspections are not of value for this thesis. Further, we are going to present research done in regards to visual object detection and classification.

3.2.1 An early approach to powergrid detection

A paper by Carlos Sampedro et al. [46] did already in 2014 look at the possibility of using supervised learning to detect components in regards to power line inspection. They did state early on in the paper that computer vision is a crucial technique for automating the inspection process of power lines. It is a very challenging task because of the heterogeneous and complex infrastructure power lines possess.

Another essential factor that must be taken into consideration when trying to automate the power line inspection is the quality of images. This is a challenging problem since it varies a lot depending on the type of inspection conducted and the vehicle used.

The experiment conducted is based on tower location-detection and classification of pole types. The system was divided into two stages. The first part uses a sliding window across an image and tries to determine if the cropped out window contains a tower while the second part takes the RoI from the first part to classify the type of pool. The result obtained illustrated a tower detection accuracy of 96% and type classification accuracy between 92% and 98%. The result demonstrates that automation is a possible feature for power line detection. Achieving 96% without using modern concepts such as CNN is impressive comparing it to the result presented in Subsection 3.2.3. Still, images presented in the report illustrate from the authors' point of view a *perfect condition*, which should even be easy for a HOG based approach to distinguish between a white pole and a black background.

3.2.2 Pole and crossarm detection based on pixel value

Pedro B. Castellucci et al. [47] raise a concern about how power outages may cause structural and financial losses, and that inspections done in a traditional way only covers a small part of the total infrastructure. A UAV can potentially work more efficiently, but it requires a way to guide itself.

The focus of this paper is to detect pole and cross-arms, which can be used for a drone to track the electrical poles. Looking away from the colormap experiment done, they had a system connected with multiple steps, building on the usage of Artificial Neural Network and statistical approaches. First, they classified if an image contained a pole before continuing; this is the Artificial part. Then they start to change pixel colors based on their value, and if enough percentages of white pixels fit within a fixed template, the pixels kept its color, or else they are turned black. The detection result depends on the color map used, but for simplicity, they managed at best to achieve an accuracy of 72%.

3.2.3 Transmission tower inspection with R-CNN and YOLO

Jiang Bian et al. [48] testes different methods for electrical pole detection, and proposes a modified version of Faster R-CNN named Tower R-CNN. They also introduce a drone for the inspection job and methods for detecting power lines, but this is not relevant for this thesis.

Their proposed model has a reduced amount of convolutional layers since electrical towers have low-level of edge features and do not require to be described by deeper abstract features. They also reduce the number of anchor boxes to one, which means that only boxes with an aspect ratio of 2:1 are selected as proposals, and according to them, this has increases the proposal quality and obtains higher detection accuracy. The result presented is as followed in regard to AP and FPS: Faster R-CNN 89.8% with 0.8 FPS, YOLOv2 86.8% with 5.6 FPS, and Tower R-CNN 89.8% with 5 FPS.

Result presented is better in accuracy than [47] and presumably faster than [46]. However, two aspects regarding the result need to be criticized. (1) The dataset contains 1300 images that they state are collected from inspection videos, but with a link in the paper to the applied dataset, it is not the case.

Most images appear to be scrapped from the Internet, with watermarks from multiple stock photo sellers, and at least three images contain a person attempting suicide. (2) With a restriction to only 2:1 ratio anchor boxes, it should not be possible to get better accuracy on the provided dataset, images taken from a high angle would be close to 1:1 in ratio and a low angle 3:1. Nonetheless, most images are taken from the ground, making it hard for a UAV to understand pole-top features.

3.2.4 Insulator fault detection in aerial images

Jiaming Han et al. [45] focuses on insulator fault detection on high-voltage transmission; more specifically, they state that the current methods often suffer from lack of accuracy and robustness. Developed methods only distinguish between fault and not fault, but not the amount of fault present in the insulator.

Their model structure for insulator detection uses ResNet50, with a small change in the last convolutional layer to fit the purpose of the project. Then they apply a feature pyramid [49] to detect an object in three image scales, and on top of the entire structure, the header layer of YOLOv2 is attached. This model is named ResnetV2. Additional smaller changes are produced to increase the performance of the model, but when it comes to the detection accuracy, they are below YOLOv3 and defend this with 14.5% less memory usage.

With no mention of anchor boxes being transformed in the YOLOv2 header, the paper states that the best ratio for detection is 1:5 in height/width threshold – reaching a precision of 96.3%. Their method for fault detection is similar to an earlier mentioned paper [47], where the RoI is transformed into a black and white color scheme – ending the process by highlighting possible areas with faults (i.e., the white parts in the image).

3.2.5 Electrical component detection with YOLOv3

Heipeng Chen et al. [50] performed research on recognition methods related to electrical components. With no dataset publicly available for insulator and shockproof detection, they used UAV images from a specific area of China. Because of blurriness in a lot of the available images, a pre-processing

method for image sharpening named SRCNN [51] became utilized. After blurred images had been pre-processed, they got resized down to 416x416 – matching the input-layer of YOLOv3. In addition to YOLOv3, Faster R-CNN and SSD were tested in accuracy and mAP. However, YOLOv3 had the best performance with a test accuracy of 96.45% and a mAP of 93.6%.

The paper illustrates good accuracy in detecting the objects. Nevertheless, the author has to question how useful this system is in a real situation since the focus is automatic detection with a UAV, but they manipulate bad UAV images beforehand to increase accuracy and precision. With the usage of SRCNN, they also smoothen out possible component faults.

3.2.6 Advancing automatic power line inspection

The most comprehensive documented resource in the field of power line inspection is possibly done by Van Nhan Nguyen [52], which has contributed three years of his life in researching and testing different methods. He identifies six main challenges in regards to deep learning vision-based inspection: The lack of training data, class imbalance, the detection of small power line components and defects, the detection of power lines in a cluttered background, the detection of previously unseen power line components and defects, and the lack of metrics for evaluation inspection performance. The different challenges are divided into four research papers. The last two papers focuses on line inspection [44], and a few-shot learning method [52], not relevant for further discussion.

The first paper [53] goes into the current status and the potential role of deep learning in regards to vision-based inspection, and just as previously mentioned papers, he expresses the lack of publicly training data for power line components and continuing by declaring a new dataset with 30000 manually tagged images distributed over 54 classes. With an average of 8 components in each image, a class imbalance problem is presented, where a brown insulator contains 43275 examples while missing top cap only contains 210 examples. Even with a considerable amount of examples, the paper states that well-known object detectors such as YOLO would never perform well because of the small scale of the component. However, the paper continues by saying that the result can be increased by first cropping out the mast.

The second paper [54] goes into the proposed inspection system for compo-

nent detection. They use two custom build UAVs, with different components to acquire images of power masts. The acquired images are combined with images provided by two power companies, creating a dataset containing 28674 unique images, which is a bit less than the dataset presented in the first paper. However, the number of component classes are the same. To increase the dataset for component detection, the RoI containing a mast is augmented twelve times its size by sliding the window (left, right, up, down) and flipping the image. Other augmentation techniques are used on the imbalanced classes, but the methods are not presented. The result shows a lot of different precision in regards to components and methods used. The paper continues by stating that the multi-stage pipeline has no problem addressing the fault-detection challenge. However, there is no mention of the first step in the pipeline; mast detection.

Part II

Contributions

Chapter 4

Environments

Computer vision is a popular research area for challenging a machine to understand something human takes for granted. Where humans can with only a handful of examples get a clear understanding of an object, the "eyes" of a computer require hundreds or even thousands of examples repeatedly shown for a defined number of iterations. In this chapter a short presentation of the inspection data (4.1) is presented, before defining the labeling tool used (4.2), ending with a description of the available datasets (4.3) needed for fulfilling the first goal.

4.1 Inspection Data

Working with restricted information often requires access to a closed system. The only way to access images taken of the critical power line infrastructure is by a controlled laptop from Agder Energy – limiting the possibility to self-install tools needed.

The inspection data goes into two categories: Maintenance inspection and top inspection. A maintenance inspection is a fast inspection with fair image resolution (i.e., 2481x3509, or 4864x3232), few images of each mast, and low precision in image quality. Top inspection, on the other hand, is a more delicate inspection process with high-resolution images (i.e., 6000x4000), more images of each mast, and most often have a higher precision in the photos.

Photos from both categories are inspected by hand. If a fault is found, different labels are then added to the image. The number of labels can be above 40 columns, so a deep dive into the possibilities will not be discussed. Nevertheless, the labels give information about mast-type, material in pole and cross-arm, type of fault, and age. Information that later can be used for categorizational tasks – what the columns do not contain are boundary boxes of components. With limited access to install an application on the Agder Energy's environment, a swift labeling application was developed to work in the environment (4.2).

4.2 Labeling Tool

Image labeling is a tedious process, building a tool in addition to that is something entirely different. Restriction to other labeling applications was perhaps the leading reason for the development, but there was also a belief that a specialized program could create an interest in image labeling – removing the barrier for unavailable data. The rest of this subsection is a brief description of the tool.

The labeling tool is built on the programming language Python 3.x and uses the Kivy library[1] for interface interaction. To get access to images of

[1] `https://kivy.org/`

electrical poles and their corresponding labels, an excel file generated from an early 21st-century inspection program is required. The file contains a unique path link to a restricted network drive containing the needed data. The image file is copied into the labeling program and is now ready for boundary box labeling. Too secure flexibility, the boundary box coordinates use float values between 0 and 1, allowing the image to get stretched or compressed without altering the foundation of the boundary box.

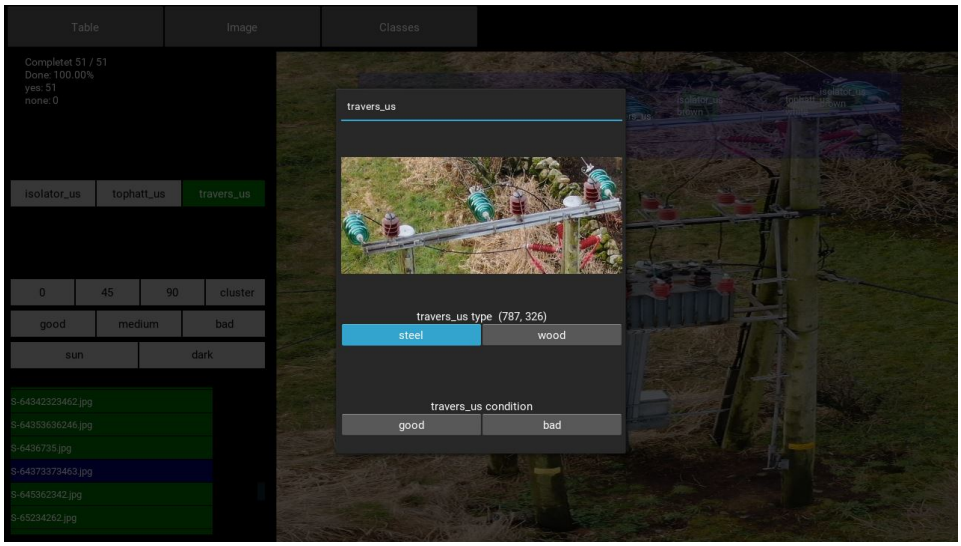


Figure 4.1: A snapshot of the self-created labeling tool

Because of the immense nature around most pole images, the program is dividing the labeling into two layers. The first layer is pole detection. This layer is only built for marking all poles contained within each image. The second layer zooms into the RoI selected from the first layer, making it easier to label small components with high accuracy. Figure 4.1 is a snapshot taken from the second layer in the application, where the component "traver"[2] is selected with a boundary box. The middle image presents the selected area and the boxes under present information relevant to the selected component. Component "insulator" had labels in regards to insulator type and the quality of the image presented, while component "top-cap" had other conditions specialised for the planned task. All datasets presented in Table 4.1 are generated by this application, fulfilling the first goal of this project.

[2]Named "travers", later defined as pole-top

4.3 Dataset

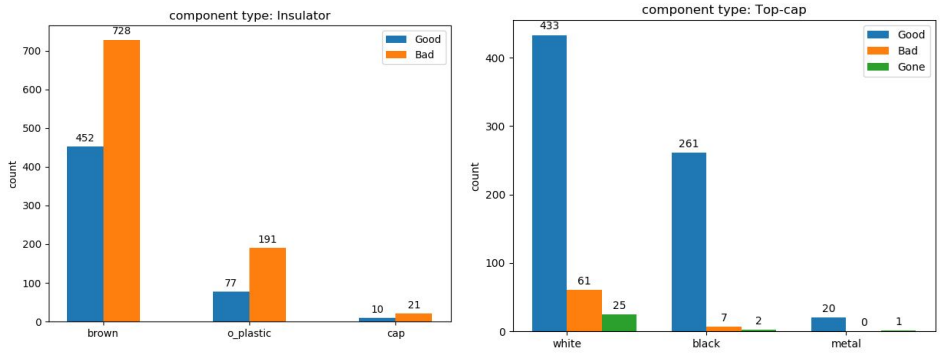
Goal 1 states: *Create or locate a dataset suitable for taking a deep learning-based approach for power line fault detection.*

Papers focusing on similar topics in regards to power line inspection are all stating the same; there are no publicly available datasets good enough for adequate training of a model. Despite, unique datasets have been produced. Carlos Sampedro et al. [46] created a dataset consisting of 1600 background and 1600 tower labeled images, with four tower classes. Pedro B. et al. [47] acquired 700 high-resolution images (i.e., 4000x3000) with a UAV. Van Nhan Nguyen [54] made a dataset consisting of 28674 extremely high-resolution images (i.e., 6048x4032) containing the position of electrical poles and the 54 most common power line components with their corresponding class label.

This paper is no different in regards to producing a working result; tedious hours have been used to create a dataset suitable. Opposite from what many of the earlier projects have done in this field, Agder Energy supplied this thesis with raw images. A file containing links to 16000 helicopter images from the year 2018 and 2019 maintenance inspection was used. Almost 5000 images were physically looked at, and marked with the location of the electrical poles. After clearing out not suitable images, the final pole location dataset consisted of 3562 images, where the split was: 2500 training and 1062 test images. The images only contain one class; electrical pole. The planned categorization was firstly based on pole-type, but the available information was considered inaccurate and unnecessary – this labeling concept got dropped, although the labels still exist within the dataset.

To continue the detection pipeline, an additional dataset was created for component detection. Figure 4.2a consist of insulator count, split between four classes. Figure 4.2b is top-cap count, consisting of three defined classes. Both components have sup-classes under each class, but the meaning is entirely different for the experiment. Insulator sub-classes are defining the available information (see figure 4.3), i.e., how much information is physically seen in the image. Top-cap sub-classes are defining the amount of fault in the component, where “good” is a perfect top-cap, while “gone” is a missing top-cap and “bad” is something in between.

After the component location detection experiment, the location accuracy for component detection was not considered suitable, and because of this, a



(a) Insulator count **(b)** Top-cap count

Figure 4.2: Component location- and class count

third dataset got created. This dataset focused on top-pole location, which should not be confused with cross-arm detection. The reason behind this is that the author did not want to disclude the detection of straight poles with no cross-arm, since they can as well contain a component fault. The top-pole dataset contains 547 images, which are the same photographs used in regards to component detection. From the *pole-top-location* dataset, was the information from component-detection re-shaped into the location of the top-pole position in regards to previous float coordinates and class type.

(a) An insulator classified *bad* **(b)** An insulator classified *good*

Figure 4.3: Insulator classification

In addition to four detection datasets, two classification datasets got cropped out from the component boundary-boxes in the second dataset. To prevent feature-loss, cropping was performed only on the original image size. The first classification-dataset uses the information found in Figure 4.2a under class-type *brown*; splitting the dataset into two classes: “good”, and “bad”.

The second classification-dataset takes information found in Figure 4.2b, splitting the dataset into four possible classes: “white_good”, “white_bad”, “black_good”, and “gone”. Because of the lack of available information in regards to “black_bad” this class has been excluded.

Table 4.1: All datasets for training, testing, and validation developed

Dataset	Size	Classes	Training	Testing
Pole-location	3562	1	2500	1062
Component-detection	547	2	426	121
Pole-top-location	547	1	350	150
Pole-top-components	547	1	426	121
Insulator classification	1180	2	944	236
Top-cap classification	780	4	390	390

Table 4.1 is a summary of all dataset used for achieving the result fount in chapter 6. All numbers refer to unique images containing multiple different components. The author is not trying to compensate for something (e.g., [52]) by flipping and shifting images to give a vision of a larger dataset than it is – YOLO already has this type of functions integrated.

Chapter 5

Proposed Solutions

In Chapter 2 we have looked into the background of object recognition, Chapter 3 presents relevant models and papers for defining a proposed solution, and Chapter 4 introduces the datasets. Related work presented in Section 3.2, mostly seems to use high-quality images to present their result – one paper [50] even goes so far as using SRCNN to fabricated higher quality - a method useless when trying to detect a crack the size of a few pixels.

Almost all related work presented in the literature only focuses on a single problem, such as direct component detection [47][50] or fault classification [45]. Most state-of-the-art methods for detection such as YOLO and R-CNN works well for what it is designed for, but detecting a fault on a component $^{1}/_{1000}$ in size of the total image would never yield any good accuracy. To compensate for this, the author believes a similar pipeline as Nhan Nguyen [52, Fig. 5.1] has presented is the best alternative to go forward with when working with uncommonly large image sizes.

5.1 The Proposed Design

The proposed design is a multi-stage pipeline, divided into two possible detection directions, and two classification methods. The presented pipeline in Figure 5.1 is designed to reduce background noise by cropping out all RoI presented in each part of the pipeline.

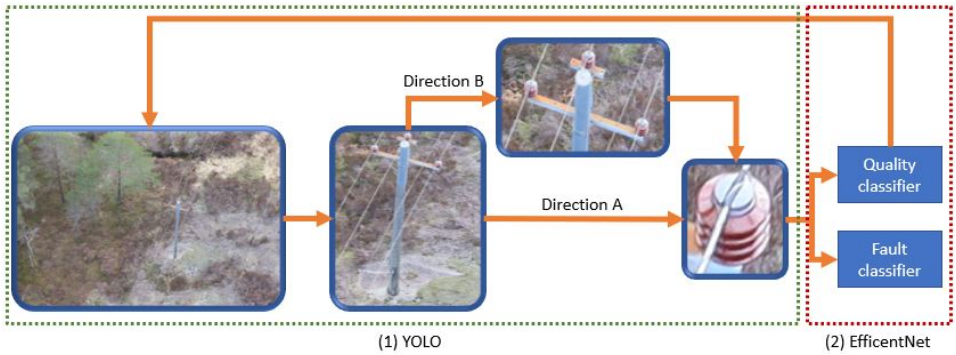


Figure 5.1: The proposed multi-stage power line inspection system for detection, classification and assuring image quality.

The first stage of the pipeline is the detection of electrical poles. The pole-location dataset is build to include a small part of the ground under not to disclude the possibility for faults at the lower part of the pole. This is something Agder Energy has stated essential to include, even though no previous work has stated something similar, but this part will not have any more further focus in this thesis. There can be multiple poles located within the field of vision because of crossing power lines or different levels of transmission networks. All poles RoI are taken to the next step since it is at the current moment, no method to determine which pole is part of the planned inspection.

In the second part of the pipeline two directions can be taken, **Direction A**, a direct detection of component, or **Direction B**, first detect the pole-top to get higher feature extraction and then detect the components. Even if the first direction is faster for the pipeline, there is more details to get from adding this extra layer. Even if the Result 6.2 illustrate something different.

When components are detected, a split occurs. Components chosen for

quality classification are classified by a given label “good” or “bad”. All other known components become transferred to fault classification. There can be various classifiers within this field, each specialized for a defined component. In this paper, only top-caps illustrate the possibility of fault detection, but with more data, the amount of component used for fault classification is extensive.

For a UAV this entire detection process is too heavy because of the limited processing power most UAVs possess. However, theUAV images can be transmitted to a more powerful device that can, as illustrated in the figure, report back the image quality – allowing the UAV to take action based on the response. However, this is more in the category of future work (7.2).

5.2 Object Detection

Goal 2 states: *Identify and implement an applicable model capable of locating electrical poles and components under complex conditions and provide state-of-the-art accuracy.*

The theoretical work presented in Chapter 3 present one- and two-stage state-of-the-art detection models. From this obtained information, the detection of power lines and component types are considered to not require the deepest of models, fine-tuned to obtain the best detection result in the MS COCO. Since power line detection is relevant both for post- and during-detection, a one-stage object detection method is the best choice, because of the real-time detection capabilities compared to existing two-stage methods.

The most suitable one-stage detection model documented is the YOLO architecture. Alexey et al. [26] presented YOLOv4 in the last month of this project, offering a state-of-the-art detector that is faster and more accurate on the MS COCO than the previously best one-stage detector; YOLOv3. YOLOv3 was considered the best choice early in the project, though available time made it possible to include YOLOv4. As mentioned in Subsection 3.1.3, YOLOv3, and YOLOv4 use the same head and similar backbone for feature extraction, which makes the setup for the experiments very similar.

5.2.1 Model setup

Before presenting the different stages in the detection part of the pipeline, a short explanation of the YOLO hyperparameters must be explained. Firstly, discovered during experimentation, the batch size of 64 needed a subdivision increase to 32 for YOLOv3 and 64 for YOLOv4, because of the limited video memory. The amount of iteration (i) for a fully trained model used the proposed calculation $\boldsymbol{i = max\_steps(4000, classes \cdot 2000)}$. Because of the policy value (i.e., "steps"), the max iteration value would affect the "steps"[1] variable. The "steps" variable used is the recommended formula: $\boldsymbol{steps = (max\_steps \cdot 0.8, max\_steps \cdot 0.9)}$. In short, these are the values used in regards to learning rate. When a defined amount of iterations reached in the "steps" variable, the rate of learning is multiplied with 0.1, meaning that the start learning rate of $1e^{-3}$ is after 80% completion changed to $1e^{-4}$, and the last 90% uses $1e^{-5}$. Explanation of other available hyperparameters and its defined value:

- **angle=0**, randomly rotates the image during training up to given amount.
- **saturation=1.5**, randomly changes the image saturation.
- **exposure=1.5**, randomly changes the image exposure.
- **jitter=0.3**, randomly changes image size and its aspect ratio from $x(1 - 2 \cdot jitter)$ to $x(1 + 2 \cdot jitter)$.
- **random=1**, randomly resizes the network size every 10 epoch.

Additional hyperparameters included in YOLOv4 are CutMix, Mixup, Mosaic, and Blur – these are described in [26]. The author of YOLOv4 [26] tested multiple combinations of these parameters and concluded that the combination of CutMix and Mosaic yielded the best accuracy in MS COCO, however because of the thesis problem, only Mosaic is activated. CutMix is considered too destructive for the given problem.

Before the YOLO model can be used, two parameters are required to be changed in each output layer of the pyramid. Firstly, the variable "classes"

[1] policy value "steps", and the "steps" variable is not the same

is changed to the number of classes, one for pole location detection, and two for component detection. Secondly, in the last convolutional layer, filter depth must be adjusted to $\boldsymbol{filters = (classes + 5) \cdot 3}$ (i.e., 18, and 21).

5.2.2 Detection stages

Detection of electrical mast is the most crucial step in the entire process; the pipeline depends on it. Similar work has presented good accuracy in pole detection [46][47][48]. However, as stated in Section 3.2, none of this result could be applied in a real situation. To work with complex environments, the model must get complex features. The dataset *Pole-location* presented in Section 4.3 is built to include this complexity, with 3562 raw images from multiple classified positions across Southern Norway, high diversity in pole types, light conditions, pole placement, pole distance, and image quality - Appendix B illustrates image types even further. Because of the immense resolution in the original images, the trained dataset is resized to 608x608, which is the same size as the input size of the model.

Detection of components is the second step. In this step the pole should already have been detected, and RoI provided. Related work for component detection have gone for a more direct detection [45][50], except [54], which uses a similar approach. What all approaches have in common is the lack of the entire pole. To cope with this, a **Direction B** is proposed to be included in the pipeline to achieve higher feature extraction of components by first detecting the pole top and present a new RoI. However, a direct detection is included in the experiment as well, to speed up the pipeline and to see what alternative is preferable. The size of input images is in the original size from the cropped out images; the difference in accuracy is immense when more features are kept (further discussed in Section 6.2).

5.3 Object Classification

Goal 3 states: *Identify and implement an applicable model suitable to classifying components given from goal two in regards to fault detection and quality assurance.*

The theoretical work compassed in Chapter 3 displays the EfficentNet family as the best state-of-the-art classifier capable of running on a normal computer. The EfficentNet family is a flexible model where a small change in the ϕ value would increase depth, width, and input resolution. Although, this ϕ value is already fine-tuned by the authors of the model and spread across eight baselines; B0 to B7. The paper does not specify how the values change from the original B0 model up to the B7. However, these values are defined in the source code[2]. The code demonstrates how the width and depth coefficient values change from 1.0 ϕ at both dimensions in B0 to 2.0 in width ϕ and 3.1 in depth ϕ in B7 - discluding the even larger B8 and L2 backbone from this comparison.

Table 5.1: EfficentNet-B0 baseline network [35, Tab 1]

Stage i	**Operator** $\hat{\mathcal{F}}_i$	**Resolution** $\hat{H}_i \times \hat{W}_i$	**Channels** $\hat{C}_i$	**Layers** $\hat{L}_i$
1	Conv3x3	224 x 224	32	1
2	MBConv1, k3x3	112 x 112	16	1
3	MBConv6, k3x3	112 x 112	24	2
4	MBConv6, k5x5	56 x 56	40	2
5	MBConv6, k3x3	28 x 28	80	3
6	MBConv6, k5x5	28 x 28	112	3
7	MBConv6, k5x5	7 x 7	192	4
8	MBConv6, k3x3	7 x 7	320	1
9	Conv1x1 & Pooling & FC	7 x 7	1280	1

Nonetheless, the third value is perhaps the most significant for the given situation to determine what baseline works best for the given classification problem; input resolution. When width and depth of EfficentNet increases,

[2] `https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/efficientnet_builder.py#L28`

the recommended resolution size also expand. B0 has a recommended value of 224x224, while B7 recommends 600x600 - three-layer color channels are invariably implied. The dimension of component images from a raw out-cropped state is small. Thus, the input dimension for EfficentNet was decided to be small, which meant that a large baseline network should not be needed for a satisfactory result.

To give a clearer understanding of the baseline model, Table 5.1 describes the B0 backbone, and this is the building-block for the entire EfficentNet family. The result related to EfficentNet goes further into the structure placed around the network, but before that, a few key statements related to each classification task must be laid out.

5.3.1 Quality classification

Image quality classification is from the author's knowledge never proposed before in the field of power line inspection to ensure a sufficient amount of information within an image. With the increase of UAVs, this is a technique that can feedback information to a drone to determine if it should retake an image before it is to far gone or to later control the quality of data provided by a third-party supplier.

Since EfficentNet is the best in what it does, it was not found rational to use YOLO for handling the classification part directly. This adds an additional layer to the pipeline but increases the whole quality of the process. Even so, by discluding the object detection part, fine-tuning can quickly be performed for classification, which was shown necessary because of the limited data.

Agder Energy defined the split between "good" and "bad" to be the visibility of the *benzel coil* at insulators, for other power grid companies this might not apply for them. However, overlooking a critical fault because of lousy image quality can be more expensive than labeling a few images to determine the quality of the data. It should be mentioned that an expert has not surveyed the quality dataset, the author had to improvise to establish a line between the classes, and because of this, we can not contradict that a few images are wrongly classified.

5.3.2 Fault classification

Classifying objects from each other is the most fundamental part of visual AI. The classifier's primary objective in the thesis situation is to determine if a component contains a possible fault that can be devastating over time for a given transmission network if not detected. Compared to a standard dataset, determining the difference between a dog and a cat, this type of data requires most often an expert in the field of power line inspection to find the most subtle faults. Since the author has narrowed experience in what to see after, and top-cap is a simple and highly used, it was a functional component for proving a concept. Figure 5.2 visual illustrates the difference between the three defined fault levels, discarding the different types of top-cap from the illustration, such as black and metal-based top-caps. It should be mentioned that the category "bad" is not a standard used by Agder Energy, although the author did not want to classify a damaged top-cap as "good".

Just as Subsection 5.3.1, YOLO is not directly used because of the limited data available, and the possibility to tune a model to fit a defined problem would never be surpassed by including the problem directly in the YOLO detection task. This allows the possibility for an endless amount of classification tasks to be done on the detected components - allowing the pipeline to be fitted for the needs of a given company.

Figure 5.2: Top-cap classification categories

Part III

Experiments and Results

Chapter 6

Result and Discussion

This chapter presents the experimental results from the proposed solution in chapter 5 and discussion related to the findings – the sections are numerically ordered by the presented steps in the proposed design from Section 5.1.

Section 6.1, 6.2, and 6.3 presents the result for detecting objects with YOLOv3 and YOLOv4. More specifically, Section 6.1 demonstrates pole detection as a classification problem. Section 6.2 continues with classification of the specific component without pole-top zoom, and Section 6.3 expands upon this with pole-top first detected. Section 6.4 and 6.5 present results related to classification of objects with EfficentNet, Where section 6.4 focuses on testing the proposed image quality detection method and uses student's t-test to control that the result can not be directly calculated, while 6.4 focuses on fault detection of top-caps. The last section (6.6) takes the relevant results and sums it up.

6.1 Electrical pole detection

Pole detection shows excellent results in the standard AP_{50}, and YOLOv4 lifted the strict-detection AP_{75} with 7.1% from the previous YOLOv3 network, seen in Table 6.1.

Both methods followed the recommended amount of iteration regarding batch size and amount of classes before stopping the training process. Because of limited video memory, the subdivision variable was 32 for YOLOv3 and 64 for YOLOv4, which means that the amount of information loaded into video memory is $\frac{Batchsize}{subdivisions}$.

As explained in Section 4.3, the applied dataset contained 3562 images, where 2500 images are for training and 1062 for testing. A minimal change regarding better feature extraction was done when moving over to YOLOv4. The image size used for YOLOv3 training was 512x512 – a fitting match in size since it must be dividable with 32. However, since YOLOv3 and YOLOv4 use 608x608 as input, the image size got resized, not from 512x512, but from the original raw image.

Table 6.1: Result for electrical pole detection

Method	**Backbone**	$\boldsymbol{AP_{30}}$	$\boldsymbol{AP_{50}}$	$\boldsymbol{AP_{75}}$
YOLOv3	DarkNet53	-	96.1	84.8
YOLOv4	CSPDarkNet53	98.2	97.7	90.0

There have been mentioned a few papers with different results in form of precision in regards to detection of pole: 89.8% [48], and 96% [46]. The YOLOv4 method has outperformed them all (as seen in Table 6.1) without picking out or fix unexceptional bad examples within the test data, demonstrated in Appendix B.

6.2 Component detection – Direction A

After the pole detection, the information outside the boundary box is cropped away, leaving the system with a defined area to focus on for component detection. Presented papers [45][50] have gone for a more direct discovery of insulators and other components. However, illustrated in Table 6.2 and based on documented literature in Chapter 3, there should be no reason to discuss further which approach gives the best precision. Appendix A gives additional information in regards to FP, FN, and TP on the result, and Appendix C present visual images of the result.

Table 6.2: Result for component detection with direct discovery

Method	**Component**	$\boldsymbol{AP_{30}}$	$\boldsymbol{AP_{50}}$	$\boldsymbol{AP_{75}}$
YOLOv3	Insulator	98.09	97.51	46.68
YOLOv4	Insulator	91.15	90.14	47.09
YOLOv3	Top-cap	94.80	93.70	30.40
YOLOv4	Top-cap	84.54	84.14	30.03

The images used for training and testing in Table 6.2 were the raw image size after cropping. The impact of using the original size compared to input size is significant. An earlier test done with YOLOv3 using image size preprocessed before training to 512x512 gave a completely different score not suitable for automation. At $\boldsymbol{AP_{50}}$ the insulator scored **69.54%** and top-cap had **83.23%**.

The author was bewildered by the extreme differences between YOLOv3 and YOLOv4 shown in Table 6.2, and it should be noted that they both used the same dataset for training and testing. However, without performing more experiments, the author believes the Mosaic hyperparameter might be responsible. This function makes the input image mutch smaller and includes multiple other images side by side, making the already small component mutch smaller.

6.3 Component detection – Direction B

The pole-detection dataset is labeled for detecting an entire pole if possible. However, the size of the pole is immensely large compared to most components located on the pole. Making it, in theory, harder for an object detection method to extract features based on the components, so by locating the pole-top beforehand, gives better feature extraction compared to direct detection.

Table 6.3: Result for component detection after pole-top is discovered

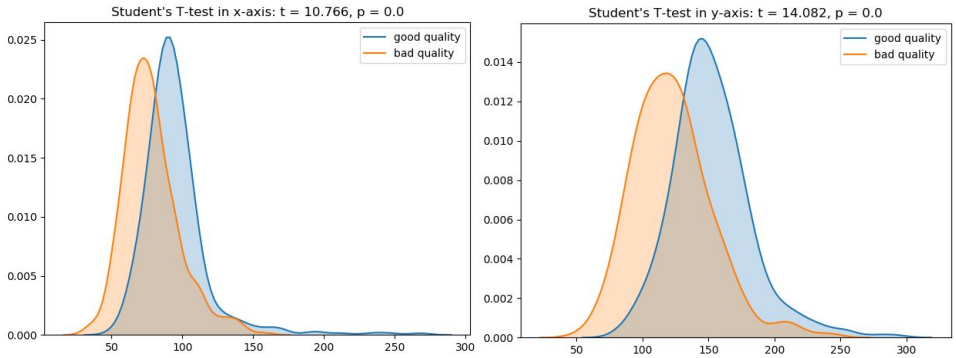
Method	**Component**	$\boldsymbol{AP_{30}}$	$\boldsymbol{AP_{50}}$	$\boldsymbol{AP_{75}}$
YOLOv3	Insulator	98.01	96.94	48.13
YOLOv4	Insulator	97.95	97.64	52.61
YOLOv3	Top-cap	89.73	88.51	21.98
YOLOv4	Top-cap	91.23	89.27	30.57

Table 6.3 present the result for this method. Where YOLOv4 have increased the insulator detection with barley 0.13% accuracy at $\boldsymbol{AP_{50}}$ compared to YOLOv3 in the direct detection test, but cannot keep up on top-cap detection. Appendix A gives a full overview of additional result, and Appendix D present visual images of the result given from the detector.

6.4 Component quality detection

6.4.1 Student's t-test

One hypotheses for this thesis was: *The quality of an image can not entirely be represented based on the number of pixels wrapping a component.* For testing this statement, a student's t-test has been used. This test only works with one-dimensional data, so images presented illustrate the pixel value in the x- and y-axis independently. Where the graph's x-axis is the pixel value, while y-axis is the density in percentage of point distributed.

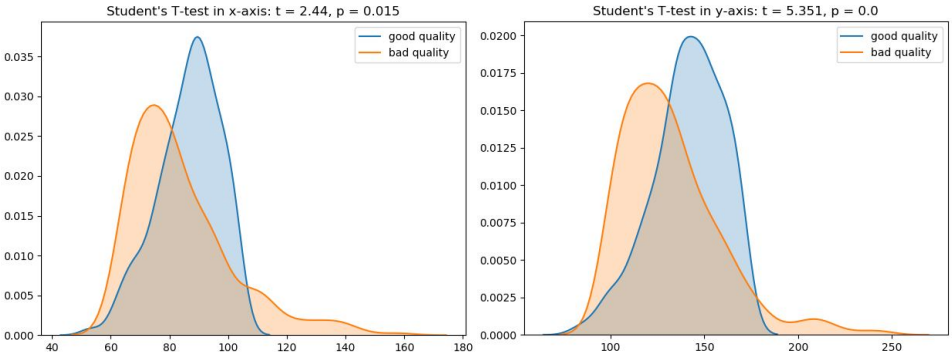


(a) Normal distribution in X-axis **(b)** Normal distribution in Y-axis

Figure 6.1: A normal distribution graph with equal variance, 0% dominant edge removal

Figure 6.1 uses 400 data points from each classified group – setting the degree of freedom to 798. By calculating the t-value separately for each axis, the t-score for the x-axis is 10.77, and for the y-axis, it is 14.08. To set a meaningful confidence level for a null hypothesis, the highest value must be less than 3.3, which both axes have surpassed.

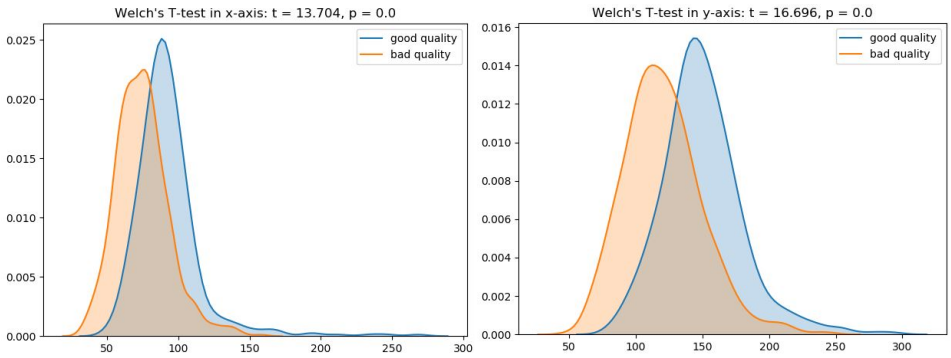
Figure 6.2 discludes the most dominant points from the dataset used in the previous figure by stripping away the 20% lowest points from "bad quality" and the 20% highest points from "good quality", this change the degree of freedom down from 798 to 638. The difference is still too significant in the y-axis with a t-value equal to 5.351. However, on the x-axis, the t-value is equal to 2.44. This value gives 1.5% confidence that there is no significant difference between the classes.



(a) Normal distribution in X-axis **(b)** Normal distribution in Y-axis

Figure 6.2: A normal distribution graph with equal variance, 20% dominant edge removal

Because of the unequal variance in the two populations, a Welch's t-test was tested. All available data points from the class brown insulator were applied, making the degree of freedom 1178 (i.e., 452 good, 728 bad). Nevertheless, including more results just expanded the difference between the two populations.



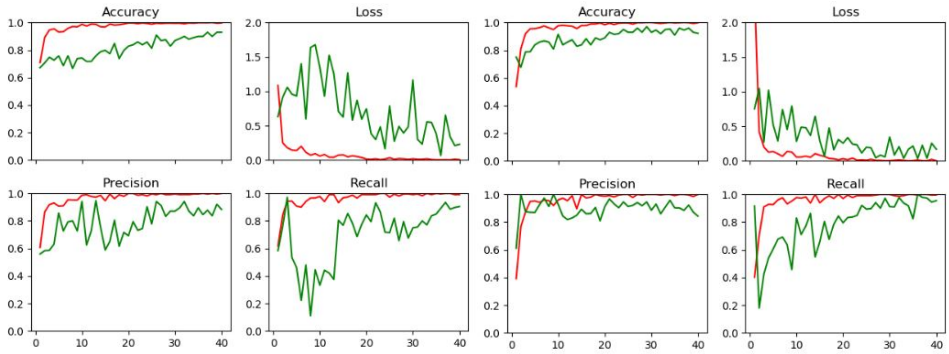
(a) Normal distribution in X-axis **(b)** Normal distribution in Y-axis

Figure 6.3: A normal distribution graph with unequal variance

Even if there is a significant difference between the sets discluding the null hypothesis, there is still an overlap between what is determined "good" and "bad", as seen in Figure 6.1, 6.2 and 6.3. This indicates that it is not always possible to count pixels – stating that the hypothesis was partially correct; an alternative hypothesis $\boldsymbol{H_a}$.

6.4.2 Result – EfficentNet

EfficentNet has produced excellent results in accuracy $\mathbf{0.93 \pm 0.02}$, precision $\mathbf{0.89 \pm 0.03}$, and recall $\mathbf{0.95 \pm 0.02}$. However, the result in Figure 6.4b was not straightforward – a dozen different tests were required to find a perfect fit.



(a) Amsgrad inactive **(b)** Amsgrad active

Figure 6.4: Classifying image quality with EfficentNet-B0 – the red lines represent training data, while the green lines represent validation data.

Without going too much into details of the process, a few structural changes are necessary to grasp. The method used was EfficentNet-B0; more extensive baselines did easily overfit the classifier and required a smaller batch size – which was the major contributor to high precision. By using AMSGrad [55], the model converged faster and gave more stable loss (seen in Figure 6.4a and 6.4b). There were added two FC-layers at the end of the model container 1280 nodes each (Table 6.4), by changing the activation function from ReLU to TanH did not possess much difference in accuracy and precision, but gave a small increase in recall and a decrease in loss. Even if there was no sign of a reasonable convergence, the learning-rate used for optimization was the recommended $\boldsymbol{\alpha}$ value $\mathbf{1}e^{-3}$ [56]. The $\boldsymbol{\alpha}$ value was decreased every fifth epoch with a multiplication of $\mathbf{0.85}^{\frac{epoch\_number}{5}}$ to fine-tune the weights[1]. Tuning the exponential decay rate $\boldsymbol{\beta_2}$, showed promising results in developing the accuracy by increasing the value with $\mathbf{2}e^{-4}$ – the change was so minimal that it is not included in the results.

Small changes to the backbone and the optimizer can increase the score

[1]No source, just recommended in a forum and gave promising results

even further, although the essential factor for a more robust model rests on the dataset. Using an image data generator to increase the number of images has been a contributing factor. However, the current dataset relies on brown insulators, which is a decreasing factor in modern power lines.

Table 6.4: Architecture – Image Quality Assurance

Stage i	**Operator** $\hat{\mathcal{F}}_i$	**Channels** $\hat{C}_i$	**Layers** $\hat{L}_i$
1	Input Layer	128x128	128x128
2	EfficentNet-B0: Model	128x128	18
3	FC & TanH	1280	2
4	FC & Sigmoid	1	1

6.5 Component fault detection

Having an imbalance in the class sizes is a significant problem – if the balance gets too big, it can be preferable for the model only to choose the more substantial alternative. Seen in Figure 4.2b the difference between the classes "good", "bad", and "gone" are so imbalanced the trained model did not surpass 85% accuracy, which is close to the exact number for guessing "good". The usage of the performance metric $\boldsymbol{F_1 score}$ (equation 2.4) would be a good alternative, except when working with softmax – precision and recall is outputting the same as accuracy making the information unreliable.

A survey on data collection [57] states that the amount of data is the major bottleneck in machine learning, and another paper [58] demonstrates that an increase in the batch size is often better for a model than a decrease in the learning rate. Limited time makes the first alternative unwise, and because of available video memory, the batch size cannot increase any further without cutting down on image size.

Working with extreme imbalance classes are nothing new, multiple papers [59][60][61] have proposed possible solutions such as oversampling, cost-sensitive learning, and performance metrics. Weighting classes have been experimented with in this project, but without access to useful performance metrics in Keras (i.e., precision and recall), this alternative was not considered any further – if someone wants to recreate the experiment, the fine-tuned weights would be close to useless for them. Oversampling was also considered, but Keras already has an oversampling method integrated when training and validating. In a worst-case scenario, this would create the same data at the validation side, making the result seem more appealing. Instead of oversampling the data, we can go in the other direction and undersample it.

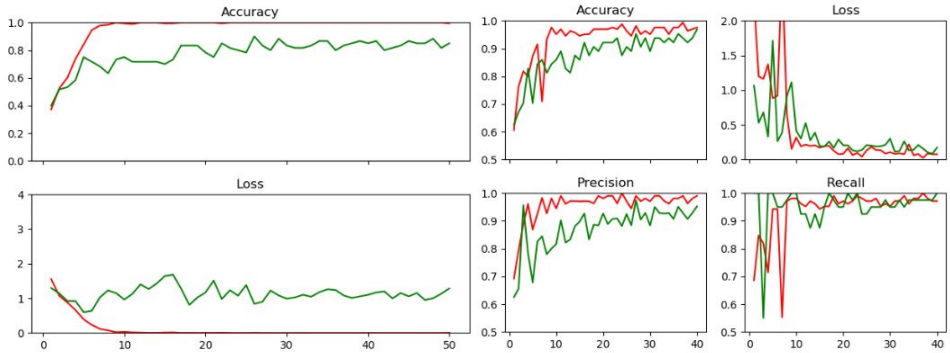
Multiple undersampling algorithms can be used [60] – divided into random and informative. Working with images makes it hard to make an informative distinguishing. Alternatives such as pixel density, image darkness, and noise could be eliminating factors. However, this is considered too much manipulation of the end-result, whereby "random" was considered the best choice. To balance the data, the class "good" was cut with almost 400 images from white top-caps and all 227 examples of black top-caps.

The result viewed in Figure 6.5a consists of three classes similar in size,

containing an extremely minimal dataset of approximately 35 images in each class. With this minimalistic dataset, the accuracy achieved is $\mathbf{0.78 \pm 0.02}$.

Table 6.5: Architecture – Fault Detection

Stage i	Operator $\hat{\mathcal{F}}_i$	Channels $\hat{C}_i$	Layers $\hat{L}_i$
1	Input Layer	128x128	1
2	EfficentNet-B1: Model	1280	18
3	FC & ReLU & Dropout 0.5	640	2
4	FC & Softmax	3	1



(a) Three classes **(b)** Two classes

Figure 6.5: Classifying top-cap fault with EfficentNet-B1 – the red lines represent training data, while the green lines represent validation data.

The model from insulator classification in Subsection 6.4.2, was initially used. However, with a dataset only $^{1}/_{10}$ in size, overfitting occurred fast. To handle this, the architecture was changed to include a dropout layer after each FC layer with a value of 0.5, the activation function was changed back to ReLU since it gave a more stable validation accuracy, the number of weights in the FC layers was changed down to 640, the backbone EfficentNet-B0 was increased to B1, and the split between validation and training was increased from 20% to 50% to provide better and more stable validation result. Table 6.4 and 6.5 displays the different choices made in the architecture.

"Bad" top-caps have a high similarity to "good" top-caps, and from similar work [52][54], this type of class has never been considered. Because of this

high similarity and lack of examples, the author believes the accuracy of $\mathbf{0.78 \pm 0.02}$ could be even higher if we only considered "gone" and "good" as the only two alternatives. Figure 6.5b illustrate a situation where there are only two classes – this gives following result: Accuracy $\mathbf{0.93 \pm 0.01}$, recall $\mathbf{0.93 \pm 0.01}$, and precision $\mathbf{0.91 \pm 0.01}$. The architecture used is similar to Table 6.5, although the last stage is changed to a binary classifier; sigmoid activation.

6.6 Summary

There have been presented a lot of results in this chapter. Before concluding this thesis, a short summary of the result must be addressed.

The proposed design in Section 5.1 is a multi-layer pipeline with two possible directions tested to find the best approach in regards to mAP. A deeper presentation of the result can be found in Appendix A, and more visual results are displayed in Appendix B, C, and D. To sum up the pipeline in regards to best result:

1. **Pole detection** – 97.7 mAP with YOLOv4
2. **Component detection** – 95.6 mAP with YOLOv3
3. **Quality classification** – 93% accuracy with EfficentNet-B0
4. **Fault detection** – 93% accuracy with EfficentNet-B1

Since direction B (6.3) has less mAP than direction A (6.2) it is not included into the list since it is meant to sum up the best possible approach.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has taken a deep-learning based approach for fault detection of power lines and proposed a new model for improving and automating the inspection process. A one-stage objection detector is concluded to be the most suitable model to implement in the detection structure because of its capability to work post and during detection. YOLOv3 and YOLOv4 have proven good result, even if YOLOv4 is a newer model and have a better result at MS COCO, the result YOLOv3 present overall is the best model for the given task. Nevertheless, both models have surpassed earlier documented results, and thereby presenting state-of-the-art power line detection.

The proposed model also includes a quality assurance detector that can assist the UAV or provide a score of a post-inspection dataset. The quality of an image is a critical factor for fault detection, and the result shows that it is a working concept that is more suitable than calculating the abundance of features based on pixels. However, the t-test illustrates that the quality of an image can be represented based on the number of pixels wrapping a component but can not entirely be based on it.

Undersampling the fault detection dataset took a significant part of the available data. Nonetheless, the fine-tuned model managed to produce a

promising result - It illustrates the possibility for a deep learning-based approach to be used for out phasing a physical inspection as long the dataset gets sufficient. However, with the presented result, this approach will be safer as a supporting tool until an expert can no longer exceed the detector.

7.2 Future Work

This thesis has presented a new way to work with power line inspection images, and presented results are promising. However, the approach has only worked with truth images, i.e., images predefined by a human. To move forward, the author wants to state three areas for future work.

A full pipeline experiment would be an interesting approach to move forward with to see the start-to-end result. With a fully functional pipeline the last step in Section 5.1 could be tested on UAV images. However, this requires a computer with a bit of processing power to give a high-speed response.

Further improving the datasets with more components and examples would most likely increase the result even further and make the detector more robust. The pole detection and component detection have excellent results, but the focus should be on improving the classification data before adding more components. A pipeline expert at top-cap faults is more beneficial than a pipeline that finds a modest amount of faults across multiple components.

The proposed image quality detector has only been theoretically tested, and all used images are based on the author's opinion in regards to "bad" and "good". This concept has shown excellent results, but a future dataset should be developed by an expert in the field, and should contain other components for more adaptability.

References

[1] S. Corr, *Farmer wins £42,000 compensation after mod chopper scares cows to death*, Available: https://www.belfastlive.co.uk/news/farmer-wins-42000-compensation-after-14252648. Accessed on: 07.05.2020.

[2] Agderposten, *Vettskremte rådyr løp ut på e18: Ett ble påkjørt, flere måtte avlives*, Available: https://www.agderposten.no/nyheter/vettskremte-radyr-lop-ut-pa-e18-ett-ble-pakjort-flere-matte-avlives/. Accessed on: 07.05.2020.

[3] O. K. Stokka, *Her blir elg jaget i døden*, Available: https://www.aftenbladet.no/lokalt/i/BmKLg/her-blir-elg-jaget-i-dden. Accessed on: 07.05.2020.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[5] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-100)," 1996. [Online]. Available: http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php.

[6] M. Everingham, C. Williams, A. Zisserman, L. Van Gool, and K. Leuven, "The 2005 pascal visual object classes challenge," Jan. 2006.

[7] PASCAL, *Visual object classes challenge 2012 (voc2012)*, Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/. Accessed on: 18.03.2020, 2012.

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

[9] J. Deng, K. Li, M. Do, H. Su, and L. Fei-Fei, "Construction and Analysis of a Large Scale Image Ontology," Vision Sciences Society, 2009.

[10] T. L. et al., "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[11] K. P. Shung, *Accuracy, precision, recall or f1?* Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Accessed on: 28.03.2020.

[12] Cocodataset, *Detection evaluation*, Available: http://cocodataset.org/#detection-eval. Accessed on: 13.03.2020.

[13] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds., Springer, 1998.

[14] F.-F. L. et al., *Convolutional neural networks (cnns / convnets)*, Available: http://cs231n.github.io/convolutional-networks/. Accessed on: 12.03.2020.

[15] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, *Striving for simplicity: The all convolutional net*, 2014. arXiv: 1412.6806 [cs.LG].

[16] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *CoRR*, vol. abs/1807.05511, 2018. arXiv: 1807.05511. [Online]. Available: http://arxiv.org/abs/1807.05511.

[17] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, Nov. 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, vol. 2, Jun. 2005.

[19] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. arXiv: 1311.2524. [Online]. Available: http://arxiv.org/abs/1311.2524.

[20] L. J. et al., "A survey of deep learning-based object detection," *CoRR*, vol. abs/1907.09408, 2019. arXiv: 1907.09408. [Online]. Available: http://arxiv.org/abs/1907.09408.

[21] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. arXiv: 1504.08083. [Online]. Available: http://arxiv.org/abs/1504.08083.

[22] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497.

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

[24] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[25] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. S. Huang, "Unitbox: An advanced object detection network," *CoRR*, vol. abs/1608.01471, 2016. arXiv: 1608.01471. [Online]. Available: http://arxiv.org/abs/1608.01471.

[26] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934.

[27] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, Sep. 2013. DOI: 10.1007/s11263-013-0620-5.

[28] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016. arXiv: 1612.08242 [cs.CV].

[29] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, *Cspnet: A new backbone that can enhance learning capability of cnn*, 2019. arXiv: 1911.11929 [cs.CV].

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014. arXiv: 1406.4729. [Online]. Available: http://arxiv.org/abs/1406.4729.

[31] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *CoRR*, vol. abs/1803.01534, 2018. arXiv: 1803.01534. [Online]. Available: http://arxiv.org/abs/1803.01534.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[33] M. Z. A. et al., *The history began from alexnet: A comprehensive survey on deep learning approaches*, 2018. arXiv: 1803.01164 [cs.CV].

[34] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. arXiv: 1409.1556 [cs.CV].

[35] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. arXiv: 1905.11946. [Online]. Available: http://arxiv.org/abs/1905.11946.

[36] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, and Q. V. Le, *Adversarial examples improve image recognition*, 2019. arXiv: 1911.09665 [cs.CV].

[37] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, *Self-training with noisy student improves imagenet classification*, 2019. arXiv: 1911.04252 [cs.LG].

[38] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, *Fixing the train-test resolution discrepancy: Fixefficientnet*, 2020. arXiv: 2003.08237 [cs.CV].

[39] M. Jamil, S. Sharma, and R. Singh, "Fault detection and classification in electrical power transmission system using artificial neural network," *SpringerPlus*, vol. 4, p. 334, Jul. 2015. DOI: 10.1186/s40064-015-1080-x.

[40] S. Das, S. P. Singh, and B. K. Panigrahi, "Transmission line fault detection and location using wide area measurements," *Electric Power Systems Research*, vol. 151, pp. 96–105, 2017, ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2017.05.025. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378779617302237.

[41] M. Jamil, S. Sharma, and R. Singh, "Fault detection and classification in electrical power transmission system using artificial neural network," *SpringerPlus*, vol. 4, p. 334, Jul. 2015. DOI: 10.1186/s40064-015-1080-x.

[42] M. Gerke and P. Seibold, "Visual inspection of power lines by u.a.s.," in *2014 International Conference and Exposition on Electrical and Power Engineering (EPE)*, 2014, pp. 1077–1082.

[43] O. Menendez, M. Perez, and F. auat cheein, "Vision based inspection of transmission lines using unmanned aerial vehicles," Sep. 2016, pp. 412–417. DOI: 10.1109/MFI.2016.7849523.

[44] V. N. Nguyen, R. Jenssen, and D. Roverso, *Ls-net: Fast single-shot line-segment detector*, 2019. arXiv: 1912.09532 [cs.CV].

[45] J. e. a. Han, "A method of insulator faults detection in aerial images for high-voltage transmission lines inspection," *Applied Sciences*, vol. 9, p. 2009, May 2019. DOI: 10.3390/app9102009.

[46] C. Sampedro Pérez, C. Martinez, A. Chauhan, and P. Campoy, "A supervised approach to electric tower detection and classification for power line inspection," Jul. 2014. DOI: 10.1109/IJCNN.2014.6889836.

[47] P. B. C. et al., "Pole and crossarm identification in distribution power line images," in *2013 Latin American Robotics Symposium and Competition*, 2013, pp. 2–7.

[48] J. Bian, X. Hui, X. Zhao, and M. Tan, "A monocular vision–based perception approach for unmanned aerial vehicle close proximity transmission tower inspection," *International Journal of Advanced Robotic Systems*, vol. 16, 2019.

[49] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. arXiv: 1612.03144. [Online]. Available: http://arxiv.org/abs/1612.03144.

[50] H. Chen, Z. He, B. Shi, and T. Zhong, "Research on recognition method of electrical components based on yolo v3," *IEEE Access*, vol. 7, pp. 157 818–157 829, 2019.

[51] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *CoRR*, vol. abs/1501.00092, 2015. arXiv: 1501.00092. [Online]. Available: http://arxiv.org/abs/1501.00092.

[52] N. Van Nhan, "Advancing deep learning for automatic autonomous vision-based power line inspection," PhD thesis, University of Tromsø, 2019. [Online]. Available: https://hdl.handle.net/10037/16815.

[53] V. N. Nguyen, R. Jenssen, and D. Roverso, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *International Journal of Electrical Power & Energy Systems*, vol. 99, pp. 107–120, 2018, ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2017.12.016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061517324444.

[54] V. Nguyen, R. Jenssen, and D. Roverso, "Intelligent monitoring and inspection of power line components powered by uavs and deep learning," *IEEE Power and Energy Technology Systems Journal*, vol. PP, pp. 1–1, Jan. 2019. DOI: 10.1109/JPETS.2018.2881429.

[55] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *CoRR*, vol. abs/1904.09237, 2019. arXiv: 1904.09237. [Online]. Available: http://arxiv.org/abs/1904.09237.

[56] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. arXiv: 1412.6980 [cs.LG].

[57] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data - AI integration perspective," *CoRR*, vol. abs/1811.03402, 2018. arXiv: 1811.03402. [Online]. Available: http://arxiv.org/abs/1811.03402.

[58] S. L. Smith, P. Kindermans, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *CoRR*, vol. abs/1711.00489, 2017. arXiv: 1711.00489. [Online]. Available: http://arxiv.org/abs/1711.00489.

[59] C. Veni, "On the classification of imbalanced data sets," Nov. 2011. DOI: 10.13140/RG.2.2.14964.24961.

[60] M. A. Arefeen, S. T. Nimi, and M. S. Rahman, *Neural network based undersampling techniques*, 2019. arXiv: 1908.06487 [cs.LG].

[61] H. He and E. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, pp. 1263–1284, Oct. 2009. DOI: 10.1109/TKDE.2008.239.

Appendices

Appendix A

Results – Detection Outputs

Appendix A presents all out-put values provided from the object detectors in Table A.1 and A.2. Most numbers are represented on the same dataset, although YOLOv3 (layer 1) pole detection had 64 images less to test on and 500 images less to train on compared to the YOLOv4 experiments (at layer 1).

Table A.1: Overall results numerical sorted based on layer. (1) pole detection, (2) component detection at Direction A, (3) component detection at Direction B

Layer	Method	IoU	mAP	Precision	Recall	TP	FP	FN
1	YOLOv3	50	96.12	98	93	985	24	69
1	YOLOv3	75	84.79	88	85	899	124	155
1	YOLOv4	30	98.20	97	97	1083	38	29
1	YOLOv4	50	97.38	96	96	1066	43	46
1	YOLOv4	75	90.01	90	90	1004	117	108
2	YOLOv3	30	96.44	97	96	534	18	24
2	YOLOv3	50	95.61	97	96	535	17	23
2	YOLOv3	75	38.54	59	59	327	225	231
2	YOLOv4	30	87.84	43	93	521	694	37
2	YOLOv4	50	87.14	43	93	520	695	38
2	YOLOv4	75	38.56	28	61	343	872	215
3	YOLOv3	30	93.87	93	93	514	37	36
3	YOLOv3	50	92.73	93	93	510	41	40
3	YOLOv3	75	35.05	58	59	322	229	228
3	YOLOv4	30	94.59	88	97	532	74	18
3	YOLOv4	50	93.45	87	96	530	76	20
3	YOLOv4	75	41.59	58	64	353	253	197

Table A.2: Full information of component results grouped by direction. Direction A represents the entire pole, and Direction B represents the pole-top.

Direction	Method	Class	IoU	AP	TP	FP
A	YOLOv3	Insulator	30	98.09	407	9
A	YOLOv4	Insulator	30	91.15	394	517
A	YOLOv3	Insulator	50	97.51	408	8
A	YOLOv4	Insulator	50	90.14	394	517
A	YOLOv3	Insulator	75	46.68	262	154
A	YOLOv4	Insulator	75	47.09	275	636
A	YOLOv3	Top-cap	30	94.80	127	9
A	YOLOv4	Top-cap	30	84.54	127	177
A	YOLOv3	Top-cap	50	93.70	127	9
A	YOLOv4	Top-cap	50	84.14	126	178
A	YOLOv3	Top-cap	75	30.40	65	71
A	YOLOv4	Top-cap	75	30.03	68	236
B	YOLOv3	Insulator	30	98.01	402	8
B	YOLOv4	Insulator	30	97.95	410	41
B	YOLOv3	Insulator	50	96.94	399	11
B	YOLOv4	Insulator	50	97.64	409	42
B	YOLOv3	Insulator	75	48.13	270	140
B	YOLOv4	Insulator	75	52.61	286	165
B	YOLOv3	Top-cap	30	89.73	112	29
B	YOLOv4	Top-cap	30	91.23	112	33
B	YOLOv3	Top-cap	50	88.51	111	30
B	YOLOv4	Top-cap	50	89.27	121	34
B	YOLOv3	Top-cap	75	21.98	52	89
B	YOLOv4	Top-cap	75	30.57	67	88

Appendix B

Results – Pole Detection

Appendix B includes images from the test dataset for pole detection. The method presented is the one with the most significant result in regards to pole detection, YOLOv4. The images have never been seen previously by the system and include situations that have not been given a good result. Figure B.1 contains images where a pole is located but the detector can not see it. Figure B.2 illustrates a situation where it has outperformed the dataset – it detects poles not labeled, and one image where the detector disagrees with the pole size.

Figure B.1: Pole detection results with no detection of visible poles

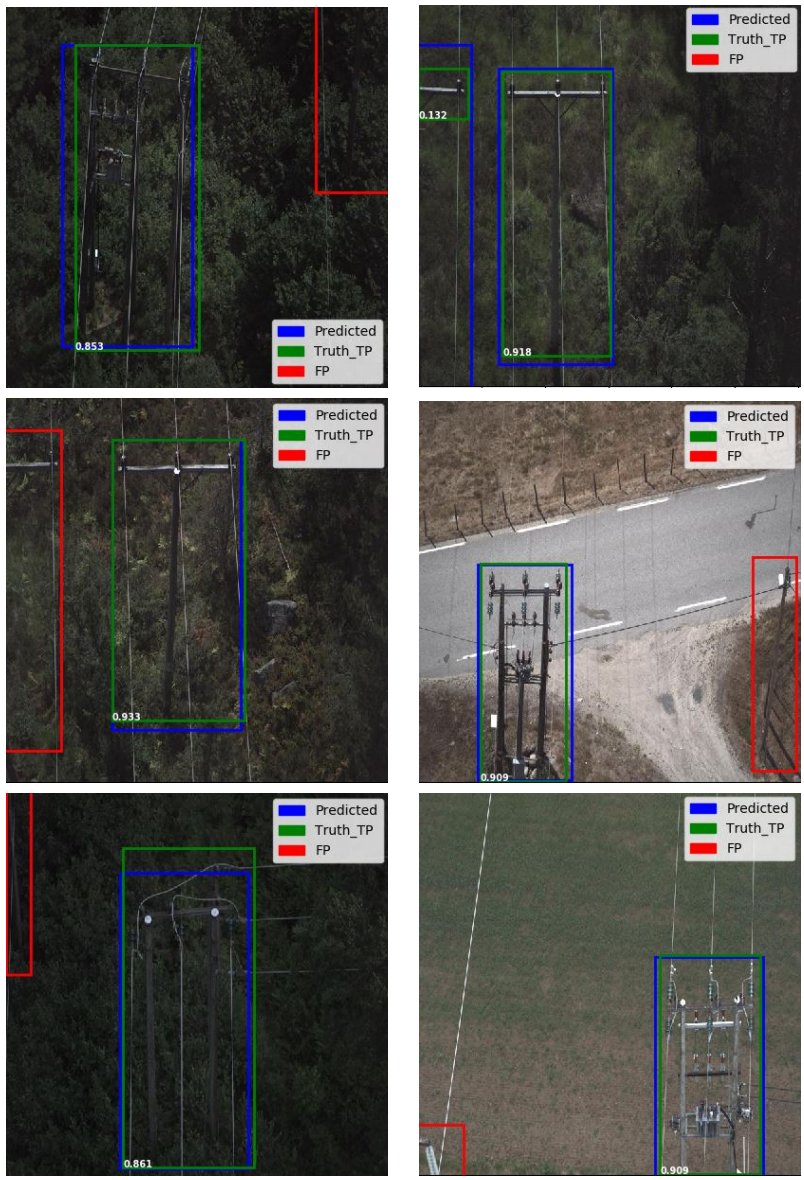


Figure B.2: Detecting existing poles, but not marked as truth

Appendix C

Results – Direction A

Appendix C includes images from the test dataset for component detection with a direct approach (Direction A). The method presented is the one with the most significant result in regards to component detection, YOLOv3. The images have never previously been seen by the system and illustrate situations where something was not correct, or it failed to predict the labeled component. Figure C.1 display components the detector were unable to detect, while Figure C.2 illustrates situations where a wrong prediction was made. This can be a situation where it predicts the wrong class, or it predicts the correct class, but the dataset did not contain it.

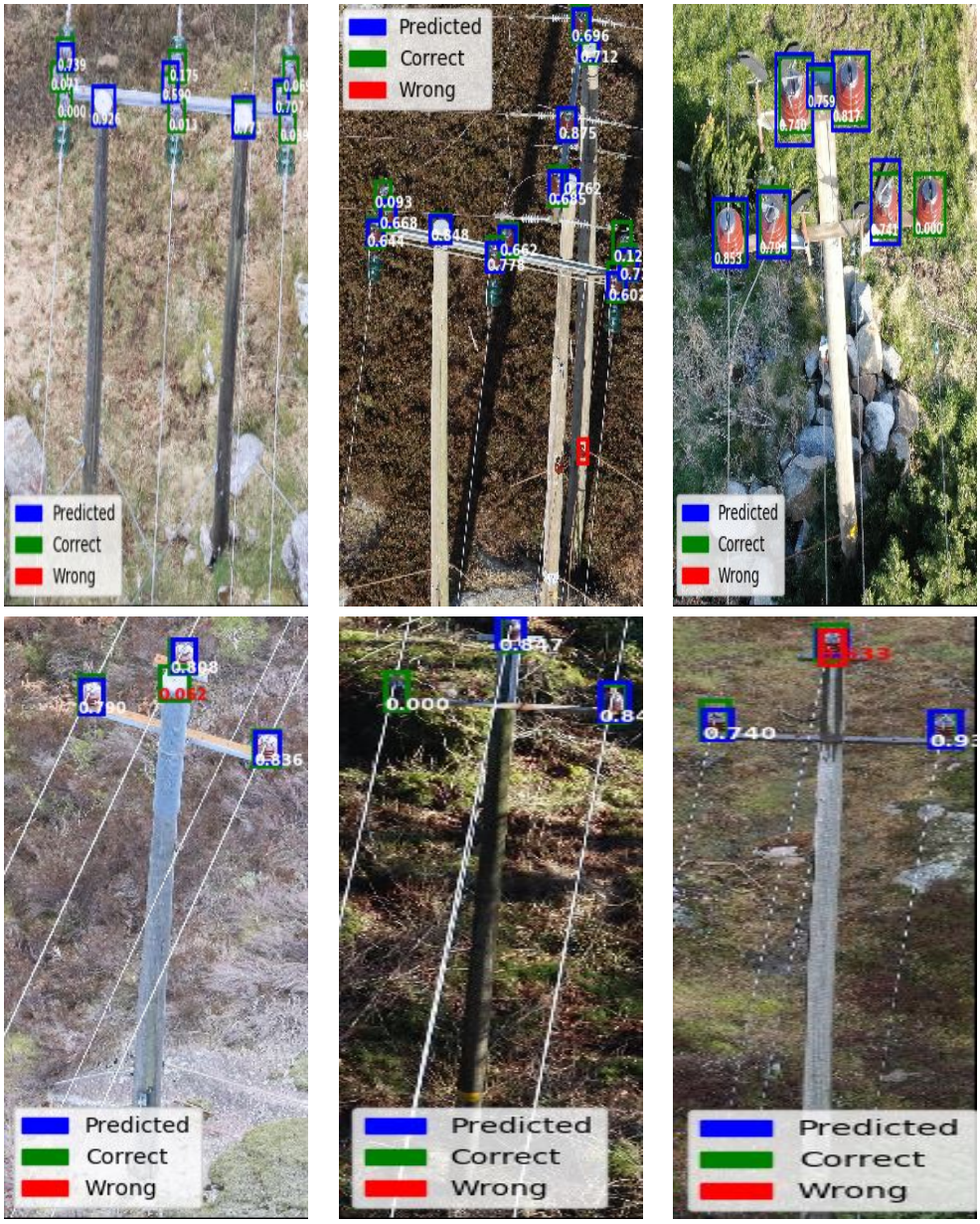


Figure C.1: Direction A – missed prediction; FN

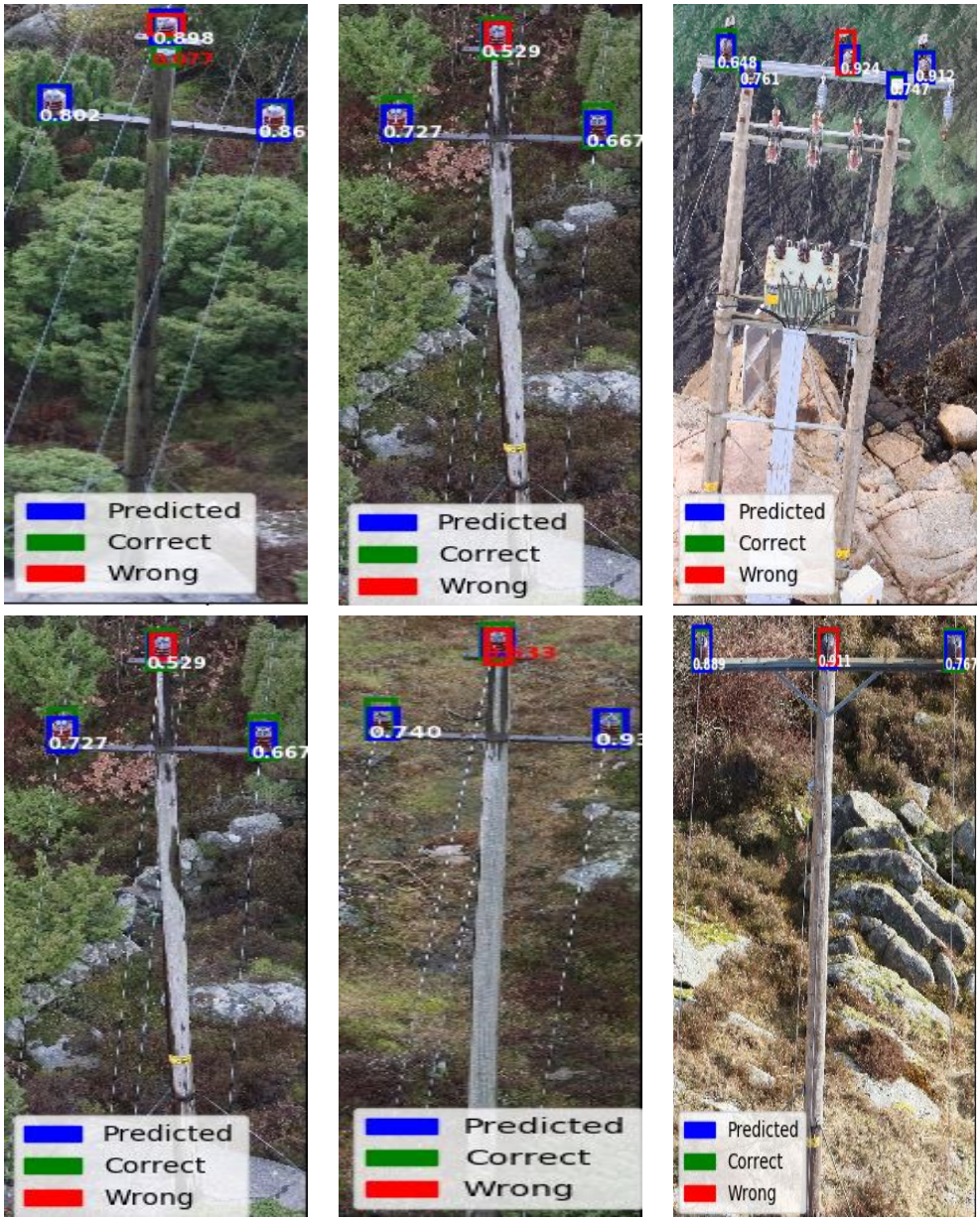


Figure C.2: Direction A – wrong prediction; FN or FP.

Appendix D

Results – Direction B

Appendix D includes images from the test dataset for component detection with the top-pole approach (Direction B). The method presented is the one with the most significant result in regards to component detection, YOLOv4. The images have never previously been seen by the system and illustrate situations where something was not correct, or it failed to predict the component. Figure D.1 display components the detector were unable to detect, while Figure D.2 illustrates situations where a wrong prediction was made. This can be a situation where it predicts the wrong class, or it predicts the correct class, but the dataset did not contain it.

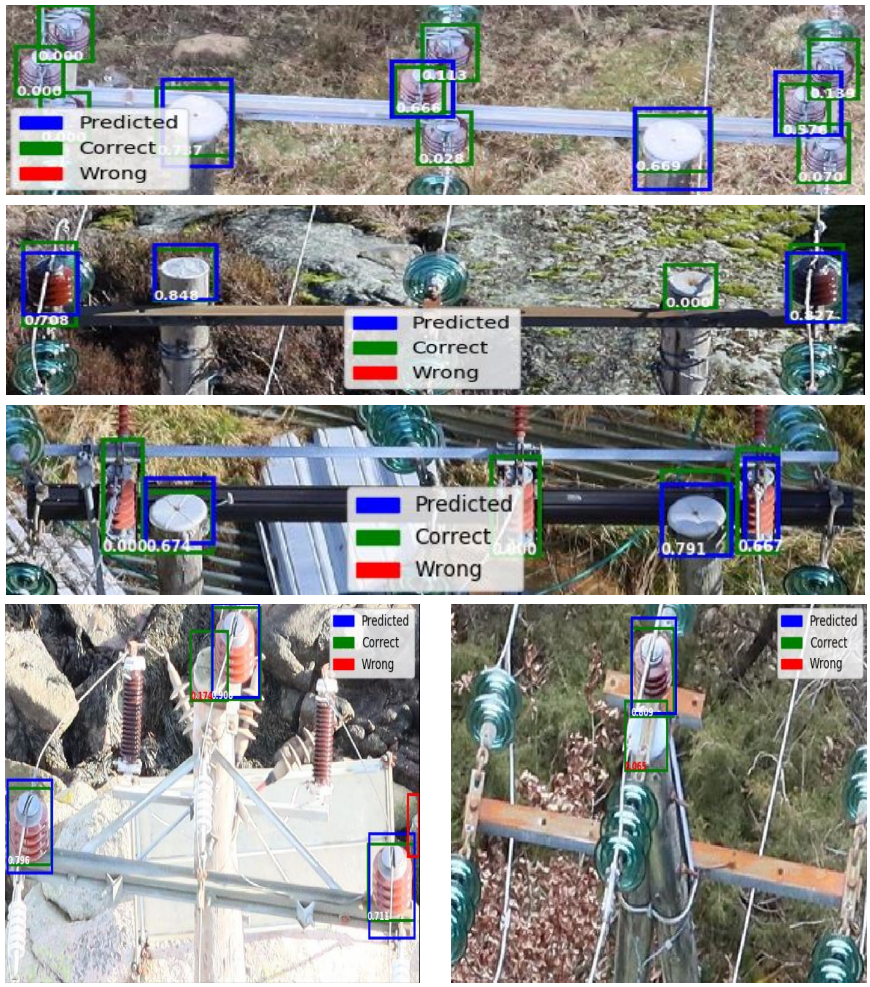


Figure D.1: Direction B – Missed prediction; FN.

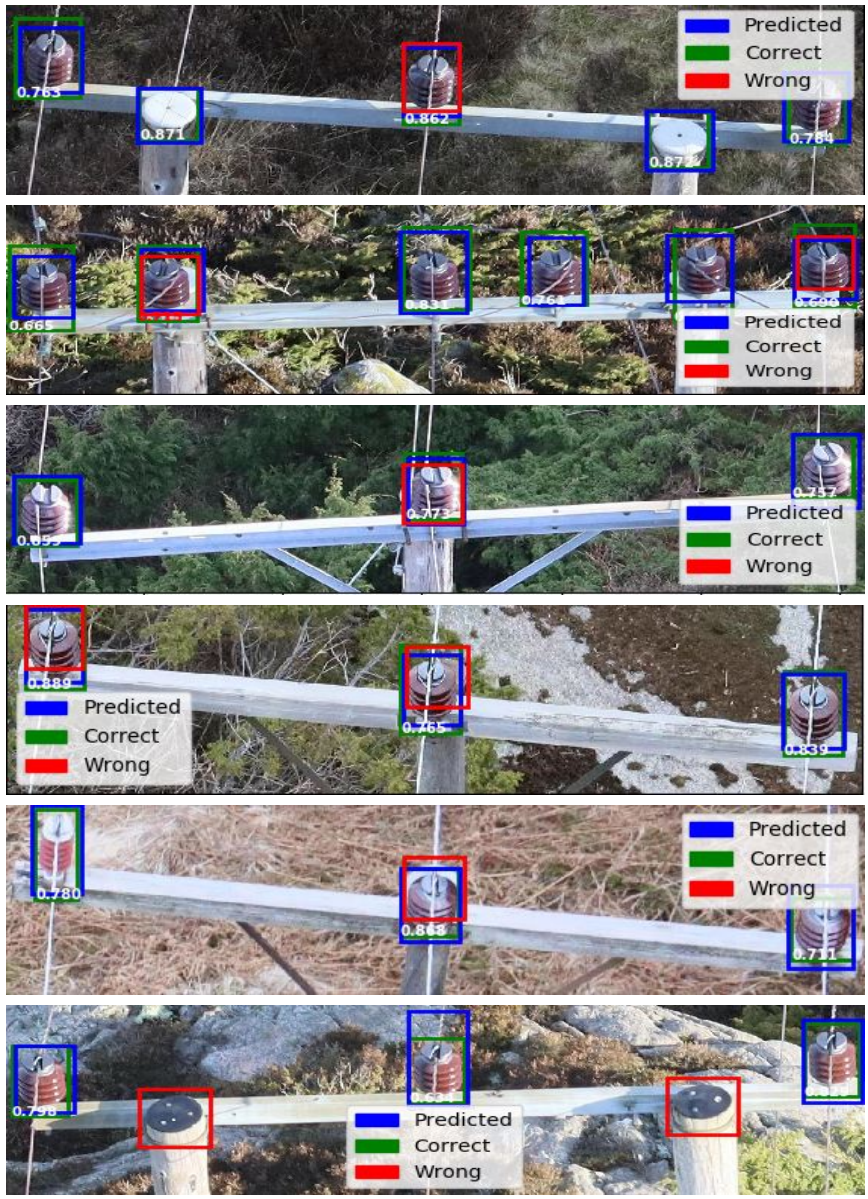


Figure D.2: Direction B – wrong prediction; FN or FP.

UiA University of Agder
Master's thesis
Faculty of Engineering and Science
Department of Information and
Communication Technology