Informatik Bachelor Thesis

Development of an AI based Road Damage Detection System

Begin 18.03.2024

Submit date 19.08.2024

By

Name
Tareq Abbas
Matrikelnummer: 7194491

Erstgutachter:
Dr. Martin semmann
UHH

Zweitgutachterin:
Julia Bräcker.
UHH

Abstract

Road networks are exposed to many factors that lead to the deterioration of their operational and structural conditions. The roads are annually examined and the maintenance they need is determined through a comprehensive survey that determines the conditions of the asphalt pavement layers and shows the levels of damage and the extent of their spread. We seek to develop the performance of maintenance and operation works by taking advantage of modern technologies and advanced methods to identify road damages. For this purpose we use in this paper AI algorithm You Only Look Once (YOLO) to be able to locate and identify the damage in a picture. This algorithm then will be used in online website which represents digitalized system. This system does not only allow us to detect the damage but also will allow companies to use the provided Information to study the damage case and make their Offers to repair them.

Inhaltsverzeichnis

Table of contents **4**

1 Introduction **5**

2 Literature Survey: **7**

2.1 IEEE Road Damage Detection (RDD) 7

2.1.1 Dataset 8

3 Computer Vision and RDD **12**

3.1 Image Processing 12

3.1.1 Image Classification 12

3.1.2 Image Localization 13

3.1.3 Object Detection 13

3.1.4 Image Segmentation 14

3.2 Method based on Neural Networks 14

3.2.1 Artificial Neural Network (ANN) 14

3.2.2 Convolutional Neural Network (CNN) 17

3.3 RCNN 24

3.4 YOLO 25

3.4.1 YOLO Structure 26

3.4.2 Training YOLO 26

3.4.3 How YOLO works: 27

3.5 System Overview: 30

3.5.1 Users: 30

3.5.2 System in the background: 30

3.6 Previous improvements of YOLO Algorithm versions for RDD: 33
3.7 Results 33

4 Summary 34

Ressources 35

Abbildungsverzeichnis

2.1 class count 9
2.2 wanted classes 10
2.3 road damage types 11

3.1 Image Processing: Object Classification, Localization, Segmentation and Detection 14
3.2 Feed Forward and Backward Propagation in one Output Artificial Neural Network 17
3.3 The structure diagram of Convolutional Neural Network . . . 18
3.4 Adding more conolutional Layers will lead to extract more features which will lead to better results. 19
3.5 Calculating features maps using 3 x 3 filters for an input image (the filter is used to detect X shape in our example) 19
3.6 pooling features from features maps in two different ways Max- and Average Pooling 21
3.7 converting the two-dimensional feature maps into a one-dimensional vector and passing the vector to the classification layer . . . 22
3.8 Description of working steps of Regional Convolutional Neural Network 24
3.9 Two Objects Share the same Center in the same Image . . . 29

Kapitel 1

Introduction

Due to Germany's strategic location in the middle of the western part of Europe, its large area, and its possession of a long border strip with a number of European countries, in addition to its being an industrial country and one of the best economies in the world, it was necessary for it to have transportation lines suitable for this global status. However, due to its vast area and the multiplicity of transportation methods, including sea and river navigation, and extensive railway and land transportation lines, unlike the countries surrounding it, it is difficult to quickly track the damage caused and determine its type and location. In addition to that we can also mention the time caused by the bureaucracy to repair these damages. All of that made the needed time to fix such problems increasing at an insane rate that does not fit with the urgent need for repairs in record time. Therefore, seeking a measure that would achieve this goal in a faster time was inevitable. This is what we sought to achieve in this project, as we will focus on the damage that roads may be exposed to. To achieve this goal, we have relied on the use of the YOLO algorithm, which has previously been tested for this purpose in many other countries and has proven its efficiency to achieve the desired goal. In these countries, images were collected of the types of damages that roads could be exposed to and their types were determined. Based on these images, we were able to train the YOLO algorithm. As for speeding up the procedure for detecting these damages, we have integrated the previous part of detecting damages with a website that allows the user to upload photos, which then the type of damage will be determined automatically in addition to displaying it on a map showing the address of its location. In addition, to provide more privacy to community members, we used image blurring technology to remove anything other than damage from the image. Thus, we allow everyone who is interested in this matter to have a general overview of the damage, its type, and its location, which reduces the time required to search for them. [1]

Kapitel 2

Literature Survey:

In view of the importance of the subject, the ability to identify road damage by means of deep learning and computer vision became possible by using the Yolo algorithm. This algorithm has the capability not only to recognize the damage but also to localize it exactly on a specific image.

2.1 IEEE Road Damage Detection (RDD)

The idea of developing a smartphone-based system to monitor road conditions began with a research team at the University of Tokyo in 2018. The team then hosted the IEEE BigData Cup challenge to evaluate contemporary methods working towards the same goal, which was widely adopted by a wide range of people and research groups around the world. However, it was noted that most current models are limited to road conditions in a single country. Developing a method applicable to more than one country could lead to the design of an independent system for detecting road damage worldwide. In light of this requirement, Arya et al. (2020) enhanced the Japanese dataset with road damage images from India and the Czech Republic. Global Road Damage Detection Competition (GRDDC) is an online event organized in conjunction with the IEEE International Conference on Big Data 2020. It invites models capable of efficiently detecting road damage in India, Japan, and the Czech Republic. The challenge aims to address the significant issue of road maintenance by leveraging advanced machine learning and computer vision techniques. Participants are provided with annotated datasets containing images of roads showing various types of damage, such as cracks, potholes, and other surface irregularities. These datasets include detailed labels indicating the location and type of damage, which are crucial for training and evaluating the models. Smartphones and dash cameras alone are sufficient to conduct road inspections, not just for one country but for all countries worldwide. The challenge also promotes collaboration and innovation among researchers, engineers, and practitioners from various fields. It provides a platform for showcasing cutting-edge research and advancements in computer vision and machine learning. Participants have the opportunity to exchange ideas and develop novel solutions that push the boundaries of current technology. In summary, the IEEE Road Damage Detection Challenge plays a crucial role in advancing automated road maintenance technologies, contributing to the creation of a safer and more efficient transportation infrastructure.

2.1.1 Dataset

The first critical point of this work is the choice of suitable datasets. It is important to select raw data from which it is known in advance which road damages classes we can face. So, there are some datasets in which the damages are known in advance or have been sufficiently documented. In this project we used the dataset for Global Road Damage Detection Challenge 2020 (GRRDC 2020), in addition to Road Damage Detection United States 2022(RDDUS2022). These two datasets contain images from 4 different countries: US, India, Japan, and Czech Republic.

Each Image is associated by an xml annotation file which contains some information about the damages included in that specific Image. The information is about height, width, depth, damage label and the coordinates of the bounding box of each road damage, which tells where exactly the damage in the image is. This dataset contains 10 different classes [D00, D01, D10, D11 , D20, D40, D43, D44, D50, D0W0]. As shown in figure 2.1.

We removed the least 6 classes from the dataset considering them as

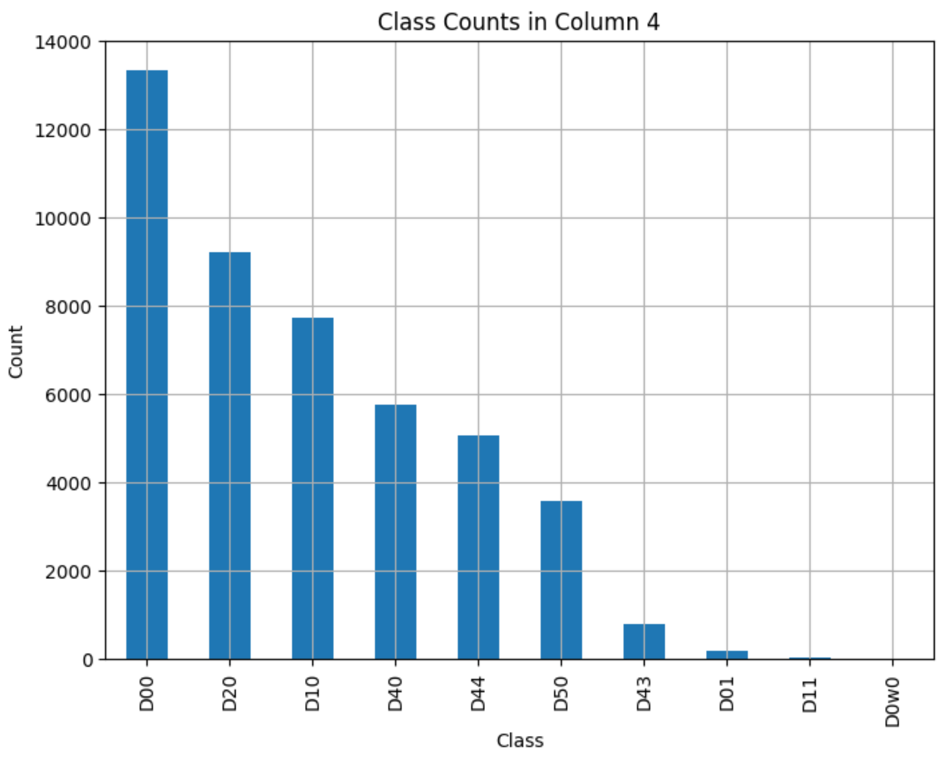


Abbildung 2.1: class count

outliers in the dataset to avoid any problem during the training such as [D11, D01, D43], and others to avoid any training mistakes [D44] (this damage represents crosswalk blur as we see in the image, but even if it has a decent amount of images, it can lead to errors if the images contain white line in them). So our dataset contains only 4 classes [D00, D10, D20, D40]. As shown in figure 2.2.

After checking the number of each class we want to collect all corresponding images and .xml files from the total dataset and store them in a new folder. Then the dataset will be divided into Train Dataset and Valid Dataset. Each folder has 2 sub folders images and labels. images folder will store all the images and labels folder will store the corresponding labels for each image. For using YOLOv9 we need to convert our annotations from (.xml) Pascal VOC format, which contains the object's bounding box coordinates (top-left corner and bottom-right corner) and its class label, to YOLO (.txt) format. In each xml file we look for details about the damage such as (xmin, ymin, xmax, ymax) of the bounding box and the 'label' of the damage. Then we need to reform it to yolo format and save them in a YOLO format.

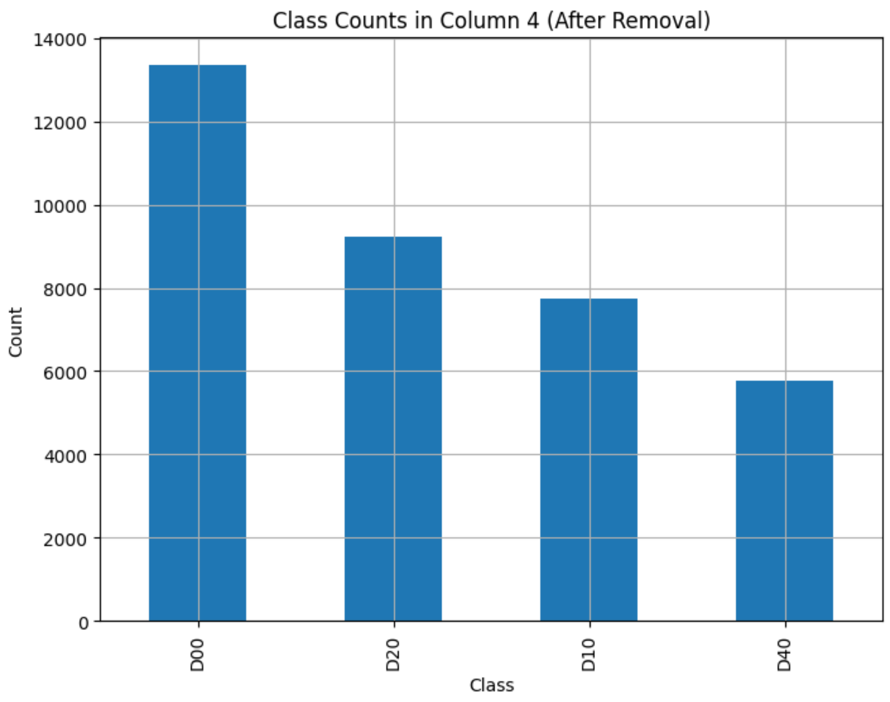


Abbildung 2.2: wanted classes

YOLO format utilizes a text file with a specific format. Each line in the text file represents a single object within the image. This line includes the class label, a confidence score indicating how certain the model is about the detection, and then four normalized values representing the bounding box location relative to the entire image size (center point of the box and its width and height as proportions of the image). The code can be found with some comments on it in $dataset - preparation.ipynb$.

The classification of cracks includes two main categories: linear cracks and alligator cracks. Linear cracks can be further divided into two subtypes: longitudinal cracks and lateral cracks. On the other hand, other types of road damage fall into a separate category, which is further subdivided into three specific subcategories: potholes and rutting, white line blur, and crosswalk blur. Figure 2.3 provides an extensive and detailed overview of the different types of damage that can occur on roadways. It includes thorough descriptions of each type of damage, outlining their specific characteristics and classifications. This comprehensive depiction aids in understanding the varied nature of roadway damages, offering detailed insights into the unique features and distinctions associated with each type of damage.

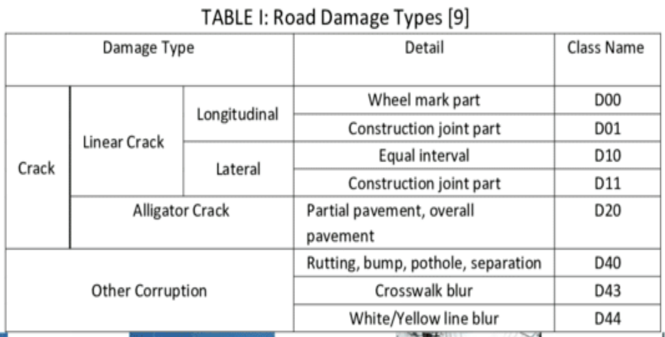
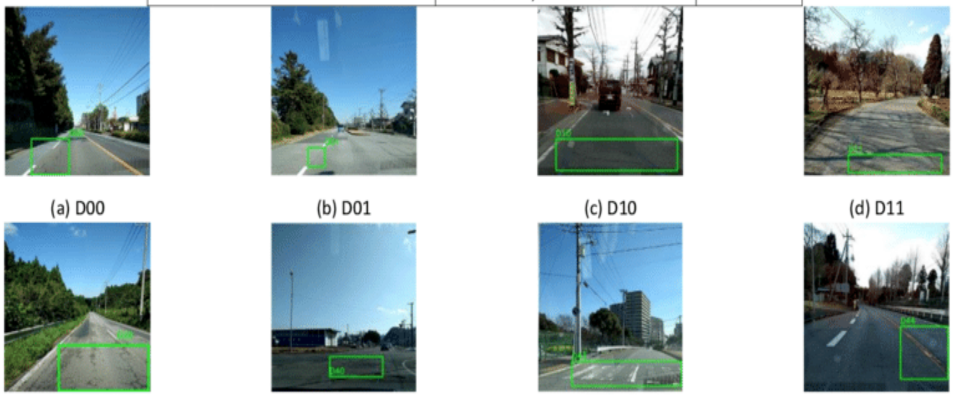


TABLE I: Road Damage Types [9]

Damage Type			Detail	Class Name
Crack	Linear Crack	Longitudinal	Wheel mark part	D00
			Construction joint part	D01
		Lateral	Equal interval	D10
			Construction joint part	D11
	Alligator Crack		Partial pavement, overall pavement	D20
Other Corruption			Rutting, bump, pothole, separation	D40
			Crosswalk blur	D43
			White/Yellow line blur	D44



(a) D00 (b) D01 (c) D10 (d) D11

(e) D20 (f) D40 (g) D43 (h) D44

Abbildung 2.3: road damage types

Kapitel 3

Computer Vision and RDD

Before starting with solving the RDD problem we need to understand a few concepts of image processing.

3.1 Image Processing

3.1.1 Image Classification

Image classification is a fundamental task in computer vision. Image classification is a process in which artificial intelligence systems analyze images and classify them into different categories. This can be achieved through various machine learning techniques, such as supervised learning, unsupervised learning, and deep learning. In supervised learning, an AI system is trained on a labeled dataset, where images are already labeled, and the system learns to recognize patterns and features that distinguish one class from another. In unsupervised learning, the system is given a large set of unlabeled images and must discover underlying patterns and structure on its own. Deep learning is a type of machine learning that uses neural networks with multiple layers to learn complex representations of images. These representations can then be used to classify images into different categories. Image classification traditionally focuses on assigning a single label to an entire image, indicating the primary object or theme present.

3.1.2 Image Localization

Object localization is a computer vision task focused on identifying and pinpointing the locations of objects within an image by predicting their bounding boxes. Unlike image classification, which assigns a single label to the entire image, object localization aims to classify objects and determine their precise positions, drawing rectangular bounding boxes around them. This process involves using machine learning techniques to extract features from the image, which are then processed to predict both the class labels and the coordinates of the bounding boxes. Training such models requires annotated datasets with labeled bounding boxes, and the process involves minimizing a combination of classification and localization losses.

3.1.3 Object Detection

It is a computer vision technology that allows us to determine the type and location of objects in an image or video. This technology can count the total number of objects in a given scene and determine their precise locations.

Object detection and image classification are sometimes confused, so I will briefly explain the difference between these two processes.

In image classification, a specific image is classified into pre-scheduled categories. Where the image represents the input of the process, and the output represents the name of the element in the image. For example, if the input is an image containing a Traffic light, the output will be the word “Traffic light,” and if the input is an image containing two Traffic lights, the output will also be “Traffic light.” From this, we can conclude that the process of classifying images, despite its importance, is limited by the results it gives us. The opposite of this is in the object detection process, where each of the detected items is identified within a frame, and in addition, the location of this item in the given scene is determined.

The importance of object detection lies in the ability of the applied algorithm to determine the type and location of a specific object in a photo or video, the speed required to do so, as well as the quantity of objects that this algorithm can detect.

3.1.4 Image Segmentation

Segmentation is an important stage that allows the extraction of qualitative information about the image, as it provides a high-level description, as each region is linked to its neighboring regions within a network of pixels. It is the process of dividing the image into interconnected and homogeneous regions according to a specific standard such as color. The union of these regions should result in a reconstruction of the original image. Image segmentation is a crucial procedure in many areas of image processing, because it allows advanced analysis and understanding of the image content.

If we turn our attention to Figure 3.1, we will find a visual representation of all the different ways in which we can process an Image.

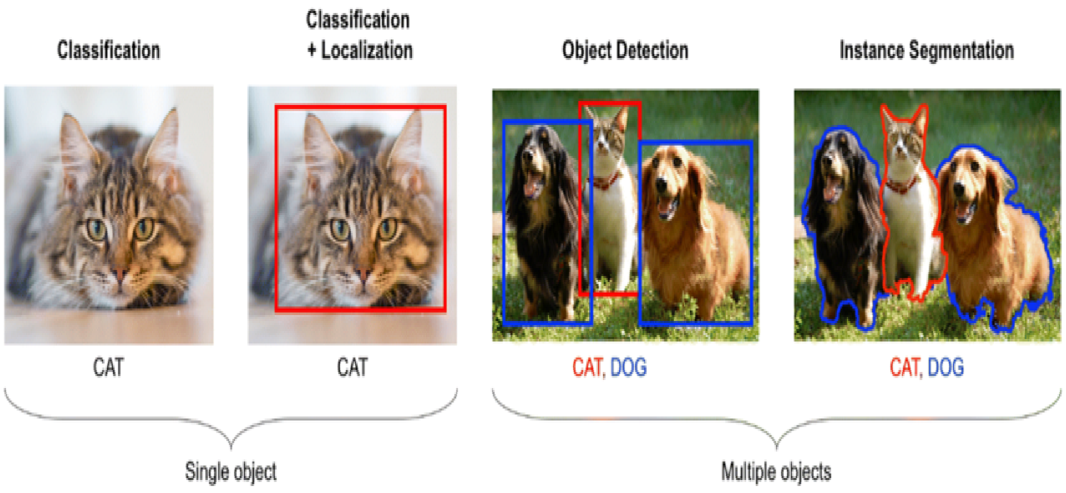


Abbildung 3.1: Image Processing: Object Classification, Localization, Segmentation and Detection

3.2 Method based on Neural Networks

3.2.1 Artificial Neural Network (ANN)

It is a simulation of human brain behavior in its pattern of thinking and learning from large amounts of data, allowing computer programs to recognize patterns and solve common problems in the fields of artificial intelligence, machine learning, and deep learning. ANN is a subcategory under the umbrella of machine learning and serves as the foundation and the main framework for algorithms that fall within deep learning. It is worth mentioning that when discussing deep learning technology, it refers to a neural network that contains many hidden layers and millions of interconnected neurons, with the term ”deepïndicating the depth of these layers. It contains an input layer, one or more hidden layers, and an output layer. The hidden layer takes information from the input layer or other hidden layers. A single neural network can contain many hidden layers, with each hidden layer analyzing and processing the outputs of the previous hidden layer before passing them to the next layer. The output layer, or more precisely the final stage in the operation of neural networks, is responsible for providing the final results for all data processed by the neural network. This layer may contain single or multiple nodes depending on the classification of the problem.

How does it work?

Since not all neurons activate simultaneously, the neurons receiving inputs from the left multiplied with the weights then they pass through the hidden layers. When the accumulated inputs from each neuron exceed a certain threshold, the previously inactive neurons will be activated. The learning method for artificial neural networks involves training from mistakes and reinforcing correct actions, known as backpropagation. Artificial neural networks use backpropagation to discern right from wrong, representing deep learning using artificial intelligence. Once the output from the final layer of the neural network is generated, the loss function, which compares inputs against outputs, is calculated, and backpropagation is performed

What is Backpropagation?

Backpropagation is a fundamental mechanism used to train neural networks, serving as a method to adjust the weights of a neural network based on the error rate produced in the previous iteration. It acts like a mediator, informing the model whether the network made an error after making a prediction. In neural networks, backpropagation involves transmitting information and associating this information with the error produced by the model when making a guess. This method aims to reduce the error, referred to as the loss function. Initially, when designing a neural network, random values are assigned as weights, and it is uncertain whether these weights are correct or suitable for the model. Consequently, the model outputs a value that differs from the actual or expected outputs, producing an error value. To achieve appropriate outputs with minimal error, the model must be trained on numerous relevant data and parameters, monitoring its progress with each prediction. The neural network's relationship with the error means that as parameters change, the error also changes. this can be achieved according to next steps:

1. Initial Weights Assignment: values of W initially assigned randomly.
2. X [x1, .., xn] value arrives to the input layer
3. Forward Pass: The output for each neuron is calculated through a forward pass where each element of the value X will be multiplied by the weight of each neuron, involving the input layer, hidden layers, and output layer. We want to learn from the input and update the weight values W because we want the weight values W to allow the model to generalize well from the training data to new, unseen data, improving its performance. This way, even if we give the model new input X2, it will multiply the new X2 values by the weight W it learned during the training phase, and thus correctly classify or predict the value of the input X2.
4. finding the error between the output layer and the expected value Y
5. using the backpropagation through the output and hidden layers, so that the weights are adjusted to reduce the error.
6. Another forward pass is performed to calculate the new output and error. If the error is minimized sufficiently, the process ends; otherwise, backpropagation is repeated to adjust the weight values.

7. This cycle continues until the error is reduced to a minimum and the desired outputs are achieved.

8. This helps adjust the weights of the neural network so that the result gets closer and closer to the known target. The goal of the backpropagation algorithm is to optimize the weights so that the neural network can learn to correctly map arbitrary inputs to outputs.

Previous steps are shown in figure 3.2

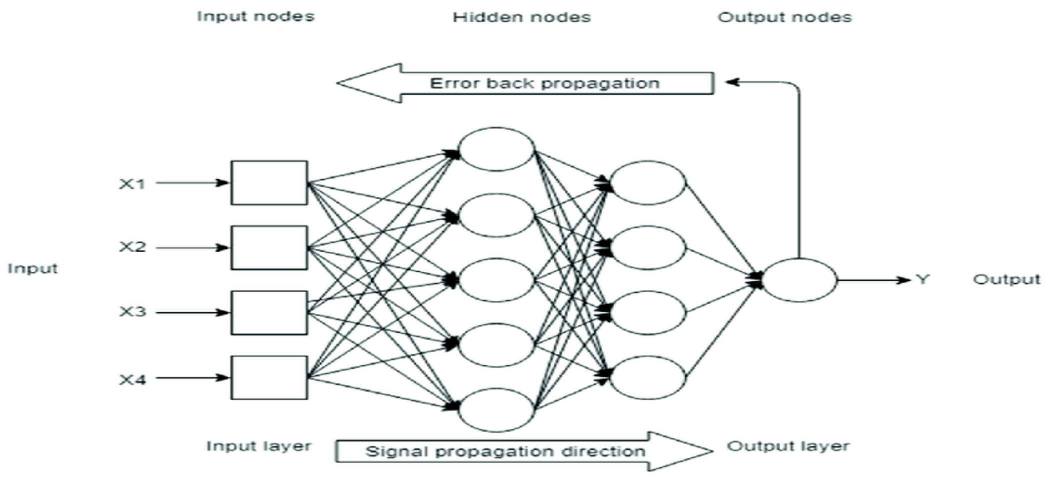


Abbildung 3.2: Feed Forward and Backward Propagation in one Output Artificial Neural Network

3.2.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) [2] are critical to deep learning and enable diverse use cases across industries such as in image analysis and computer vision, capable of accurately identifying and classifying shapes and patterns in images. Additionally, these deep neural networks rely on operational algorithms that allow them to extract distinctive features from data, enhancing machine learning performance. These neural networks are known for having fewer parameters than other networks, but they are relatively slow and complex, making them difficult to design and adjust. Their name derives from the type of hidden layers they comprise. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, and fully connected layers See Figure 3.3.

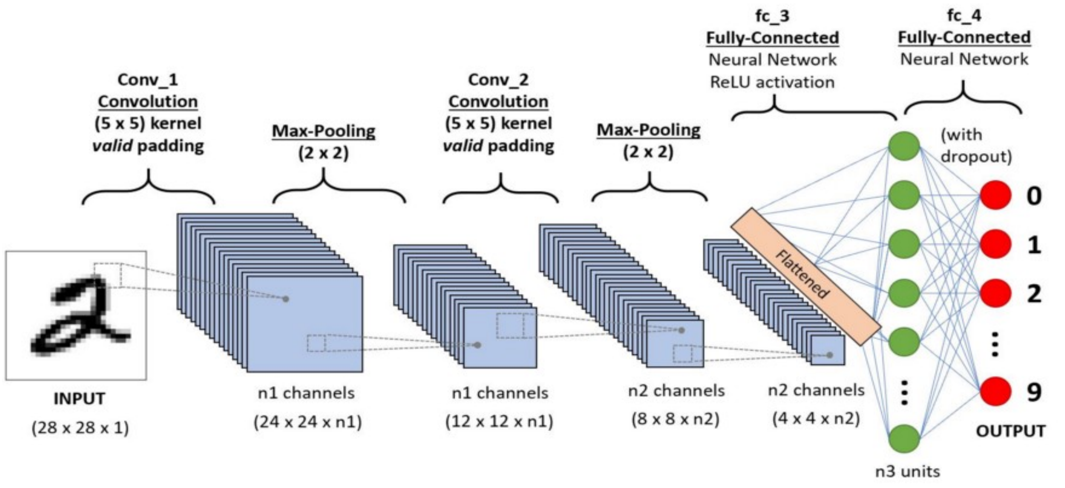


Abbildung 3.3: The structure diagram of Convolutional Neural Network

This means that instead of using standard activation functions, convolution and pooling functions are used as activation functions. Each layer serves a different purpose, performs a task on ingested data, and learns increasing amounts of complexity. To understand CNNs in detail, one must grasp the concepts of convolution and pooling, which are borrowed from the field of computer vision and defined as follows:

Additional convolutional layer

By adding more convolutional layers we can get more can lead to exploring more complex features which will lead recognising more details and shapes that can help to detect more complex objects.

Figure 3.4 can give a good understanding of its importance.

Convolution layer:

Convolution operates on two signals : one can be considered as the ïnputßignal (or image) and the other (called the kernel) as the "filteräpplied to the input image, resulting in an output image. The convolution process involves sliding the filter across the entire image, performing a dot product operation, and calculating new values. After this operation, we obtain an activation map, which is sometimes referred to as a feature map. When we apply a 3x3 kernel to a M x M input matrix, the resulting matrix dimensions become (M-2)

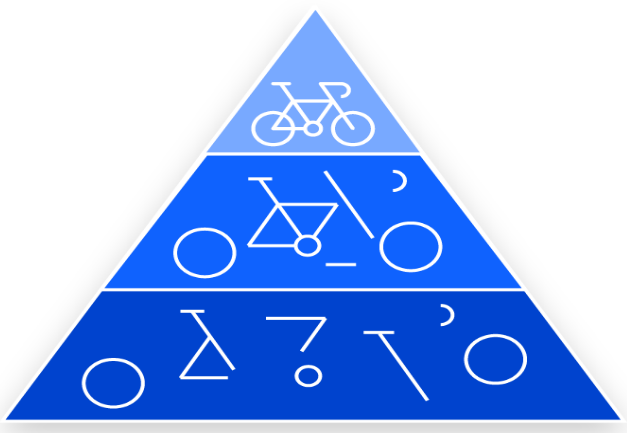


Abbildung 3.4: Adding more conolutional Layers will lead to extract more features which will lead to better results.

x (M-2). This means the matrix shrinks in size each time we perform a convolution operation. Therefore, the number of times we can apply convolution is limited. See figure 3.5 .

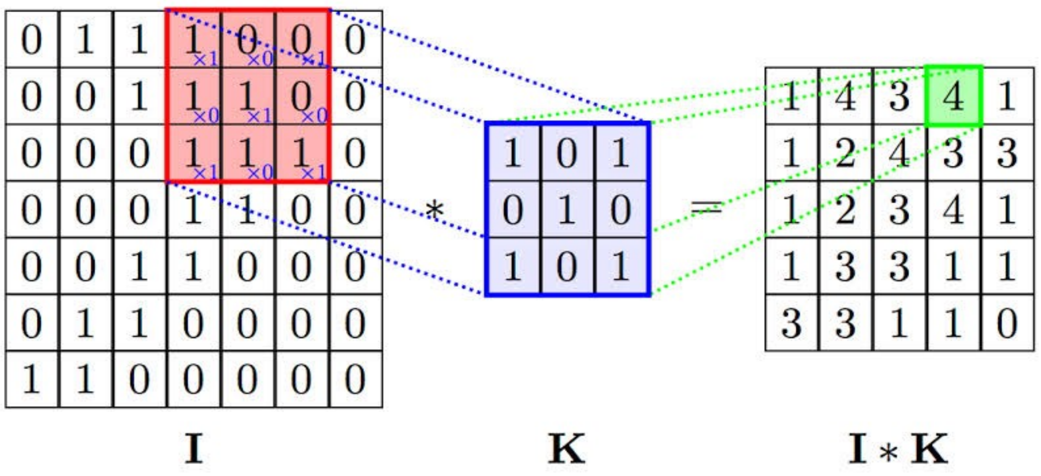


Abbildung 3.5: Calculating features maps using 3 x 3 filters for an input image (the filter is used to detect X shape in our example)

The output of each filter used in this layer represents a set of important features for image recognition (each application of the filter to a part of the image is considered a feature). The learning algorithm in the deep neural network selects the most relevant features for the reference image. It is important to note that the convolution process can identify the local dependencies of pixels (the spatial relationship in the image from the perspective of the pixels). CNNs learn the values of these filters by themselves during the training process (considering that we need to predefine some parameters before training the model, such as the number of filters, filter size, network architecture, etc.). The more filters we have, the more features are extracted from the input image, resulting in a better network for identifying patterns in new images that the network has not seen before. We pass each filter over the input image, initially performing the dot product of the matrix elements (representing the image) corresponding to the filter dimensions (in our example, 3x3). This gives us the first output value of the filter (the first feature). By repeating the previous process and sliding the filter across the entire input matrix, we obtain the final output for this filter. After completing the operations with the first filter, we repeat the same steps for all the filters in this layer (we apply multiple filters in each layer).

Activation layer:

The second step is the activation layer: We will choose the ReLU activation function, which is defined by the following relationship:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

It preserves positive values as they are and sets negative values to zero. This is the result of applying the activation function to each element of the input matrix. After the convolution process is complete, the feature map is fed into the activation layer, where the activation function is applied to each neuron, equivalent to each element in the feature map. This process considers the threshold of the neuron's output and the non-linearity in the activation process, as the previous convolution operations were linear (addition and multiplication). The most commonly used activation function in this process is ReLU, which has proven to be more effective compared to other functions like sigmoid and hyperbolic tangent, which have several drawbacks, the most notable being the vanishing gradient problem during backpropagation training. This issue arises because the derivative of these functions becomes zero at large values (in absolute terms), slowing down the training process and potentially stopping it before completion. Therefore, the ReLU function is considered the best for the training process, and we will use this function as the activation function. The ReLU function is defined as follows:

$$F(x) = \text{max}(0, x)$$

Pooling Layer :

In the third step, we use the pooling function to further adjust and reduce the output. The pooling function replaces the output of the network at a certain location with a statistical summary of the output. An example of this is Max Pooling, where each window (a set of adjacent elements) is matched with a single element representing the highest value within that window. There are also other famous pooling operations such as: Average Pooling (using the average function) and Max Pooling Max pooling is the process of finding the maximum pixel value from the region of the image covered by the kernel. Max pooling also acts as a noise suppressor. 3.6 .

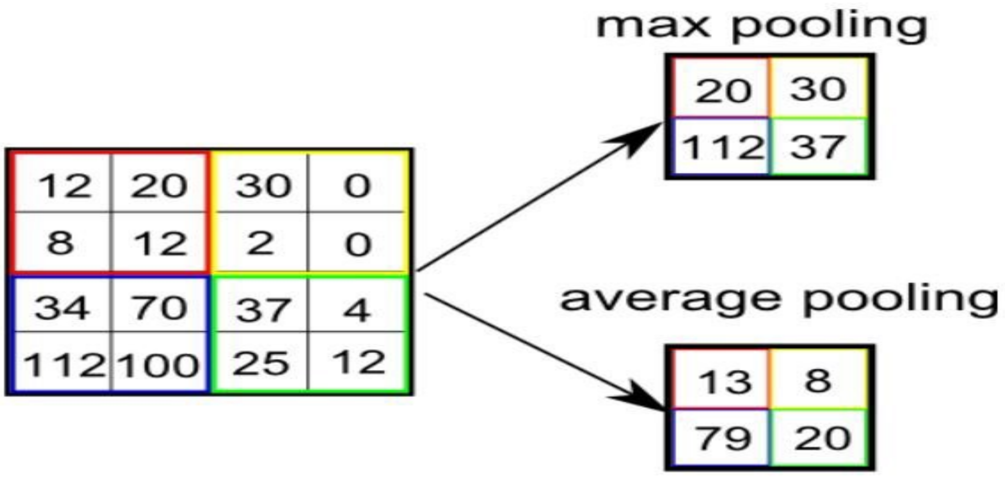


Abbildung 3.6: pooling features from features maps in two different ways Max- and Average Pooling

The output of the pooling operation is a feature map with the same depth but different width and height. Pooling has several benefits, including:

1. Reducing the dimensions of the feature maps: Pooling helps reduce the spatial dimensions of the feature maps.

2. Reducing the number of parameters and computations: This aids in controlling overfitting, which occurs when the network performs well during training but poorly during testing. By reducing the number of parameters, pooling helps prevent overfitting.

3. Making the network robust to slight translations or distortions in the input: Pooling helps the network generalize better to variations in the input data, such as noise or small distortions.

4. Useful for detecting whether a feature is present: Pooling focuses on the presence of features rather than their exact locations, which can be beneficial in certain tasks.

Flatten Layer :

After passing through the previous layers (and possibly several layers), we flatten the output of these layers into a vector that fits into the neural network's input layer to be fed into the final stage of the CNN algorithm. Flattening essentially converts the two-dimensional or three-dimensional feature maps into a one-dimensional vector so that it can be fed into a fully connected layer or any other type of layer that expects a one-dimensional input Figure 3.7 .

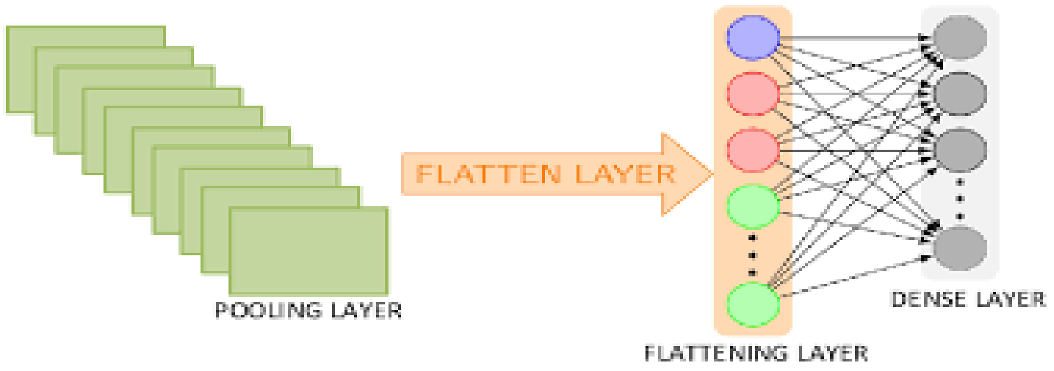


Abbildung 3.7: converting the two-dimensional feature maps into a one-dimensional vector and passing the vector to the classification layer

This flattening step is crucial before passing the data to the fully connected layers for the final classification or regression tasks.

Fully Connected Layer:

The function of the previous stages, particularly pooling, is to extract feature vectors. Initially, the network learns to detect simple features like edges, for example. These edges are then used in the next layer to detect simple shapes, and these shapes are further used to detect higher-level features in the higher layers. As the number of convolutional layers increases, the level of features learned by the network also increases. It’s not necessary that the feature maps are interpretable by humans, but for the network, they represent patterns specific to a certain class (which is determined within this layer). After feature extraction, a classifier is used to classify these features. This classifier is typically a feedforward neural network, where its input is a flattened vector composed of the feature maps after the pooling stage, and its output is a vector representing the class to which the feature map belongs.

Training the Convolutional Neural Network (CNN) involves two types of parameters:

1. Manually adjusted parameters (hyperparameters): These parameters include the size and number of filters used in each convolutional layer, as well as the stride and padding. All of these directly affect the size of the feature maps.

2. Learnable parameters updated during the training process: These parameters include the filter weights, which are the values of the filters, and the parameters used in the classification layer, such as the weights of the fully connected neural network and the biases.

During training, the network learns these parameters through the optimization process to minimize the loss function and improve its performance on the given task. The training process itself is similar as explained before in ANN.

3.3 RCNN

RCNN [3] is considered a pioneering approach that combines CNNs with region proposal mechanisms for object detection. It involves three main steps: region proposal, feature extraction, and object classification. During the region proposal stage,The regions that might not contain an object are ignored. The process of selecting regions that might contain an object is called "Region Proposals". Several algorithms have been proposed to identify important regions in the image (Region of Interest, ROI) as we see in figure 3.8. The best of these algorithms is SSelective Search (SS)."

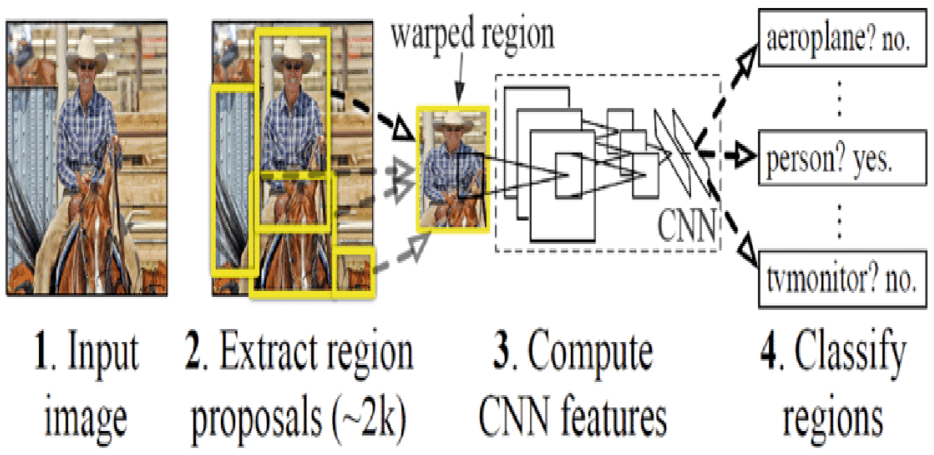


Abbildung 3.8: Description of working steps of Regional Convolutional Neural Network

This algorithm proposes about 2000 regions from the image, knowing that the regions proposed by the algorithm are not random, but each proposed region has a high probability of containing an object. It can be said that R-CNN is a region-based convolutional neural network. Steps for implementing Mask R-CNN programmatically:

1. Let's assume we have an image.
2. Identify all important regions in the image (ROI) using the Selective Search algorithm.
3. For each region, perform the following steps:

- Apply a Convolutional Neural Network (CNN) to each of the extracted regions.
- Extract the distinctive features of the image from the CNN.
- Input the features extracted from the CNN into a fully connected convolutional neural network. This will produce two outputs:
  (a) The first output predicts the bounding box for each object using a regression algorithm.
  (b) The second output classifies the object within the predicted bounding box (e.g., human, cat, car, etc.) using a Support Vector Machine (SVM) algorithm. SVM is a supervised machine learning algorithm used for binary classification and multi-class classification.

3.4 YOLO

YOLO or you only look once is a state-of-the-art (sota) algorithm used for object detection. YOLO is one of many techniques used for object detection like Single Shot MultiBox Detector or simply SSD (both YOLO and SSD are one step object detection techniques), Region-Based (R-CNN), faster RCNN, Mask RCNN, and Cascade RCNN (which are considered as two steps object detection techniques). As with many new technologies, YOLO has its own benefits and drawbacks.YOLO excels with its 'single-shot' approach. This translates to real-time performance, making it ideal for applications where swiftness is crucial, like self-driving cars or security systems that need to react instantaneously. However, this emphasis on speed comes with a slight accuracy trade-off. While YOLO boasts good overall accuracy, some slower object detection models might achieve a higher degree of precision. It's a matter of prioritizing speed for real-time scenarios where a slight dip in accuracy is acceptable. This balanced approach makes YOLO a powerful tool, and its scalability allows for further refinement. Instead of making predictions on many regions of an image, YOLO passes the entire image at once into a CNN that predicts the labels, bounding boxes, and confidence probabilities for objects in the image.

3.4.1 YOLO Structure

YOLO is considered to be an algorithm for object detection (locating the object in the image and classifying the class of that object object). When we achieve both steps correctly we can say we can detect the object we are looking for in the image. Here we need to go through many steps. Starting stage Backbone. The backbone is the “base” classification model that the object detection model is based on, where features from the input will extracted. Some of these features are Edges, Textures and object shapes, which will be used in the second stage of the algorithm The Neck. The neck neural network represents a series of layers to mix and combine image features and to pass them forward to the prediction. The last stage ’The Head’ is used to predict the objects in the input and the bounding boxes (which will be drawn around a predicted object) as well as the class of the object (in our case the type of the road damage).

3.4.2 Training YOLO

The image is divided into an S × S grid, and each cell in the grid is responsible for all objects placed in it and must predict the offset of the bounding boxes and their corresponding class probabilities for objects (if any are found).

Each cell in the grid can predict multiple bounding boxes and a confidence score for each of those boxes, and each cell in the grid predicts only one probability for each class. Therefore, the total number of classes C means C different probabilities. For each of the B bounding boxes, you have 5 values: 4 coordinates that can be the center of the object, width and height, and a confidence score for each bounding box, which equals 5, then you have C class probabilities for each cell in the grid, and in this way you get the output layer before applying non-maximum suppression.

$$\text{Total Outputs} = S \times S \times B \times 5 + C$$

Therefore, each bounding box gives 4 coordinates x, y, w, h; where x and y represent the coordinates of the center of the object relative to the grid cell and w and h are the width and height of the object relative to the entire image, and the confidence score given for each bounding box reflects how confident the model is that the bounding box contains an object and also how accurate the boxes are. Therefore, you can consider confidence to be the probability of the existence of an object, if the object's bounding box falls within the cell, because you do not yet know to which class it belongs, so it is only

$$P(\text{class } i) \times IoU(\text{Ground Truth \& Pred})$$

. Therefore, confidence takes into account both of these quantities.

3.4.3 How YOLO works:

One of the most common problems in object detection algorithms is that instead of detecting an object just once, it may be detected multiple times. For example, cars may be recognized more than once in an image. The Non-Max Suppression (NMS) technique addresses this issue by ensuring only one detection per object. It works by first examining the probabilities associated with each detection and retaining the highest probability. Then, it looks in the same region for lower probabilities and suppresses them. This process is repeated for other detected objects.

However, what if there are multiple objects in a single cell? This is a common scenario, which leads us to the concept of anchor boxes.

Downsampling, Upsampling and Multi-Scale Prediction:

YOLO (You Only Look Once) operates on downsampled versions of the input image to achieve faster processing speeds. This downsampling significantly enhances computational efficiency by reducing the amount of data that needs to be processed. However, this comes at the cost of a reduction in the spatial resolution of the output, potentially impacting the accuracy and detail of object detection. To address this issue and ensure accurate detection across various object sizes, YOLO employs a sophisticated multi-scale prediction strategy. This involves predicting objects at different scales within the feature maps, typically using different stride values. By doing so, YOLO can effectively detect objects of various sizes and dimensions within the same image, maintaining a high level of accuracy despite the initial downsampling.

Another technique used in YOLO is Upsampling involves increasing the resolution of the downscaled feature maps to restore some of the lost detail, thus improving the accuracy of object localization. This is achieved through methods like interpolation or transposed convolutions, which help to enhance the spatial resolution of the feature maps. Furthermore, anchor box allocation is a crucial aspect of YOLO's architecture. Each cell in the feature map is assigned a set of anchor boxes, which serve as reference points for predicting object boundaries. The number of anchor boxes per cell can vary depending on the version of YOLO being used. These anchor boxes allow the network to predict multiple objects within each cell, further enhancing its capability to detect and localize objects accurately. This combination of downsampling for efficiency, multi-scale prediction for size variability, and anchor box allocation for precise localization demonstrates the advanced methodologies YOLO employs to balance speed and precision in real-time object detection.

Anchor Boxes: The Foundation of Object Localization:

At the core of YOLO's object localization mechanism lies the concept of anchor boxes. These are predefined bounding boxes with distinct dimensions and aspect ratios that serve as reference points for predicting the actual bounding boxes of objects within an image. Each grid cell within the feature map is assigned a specific set of anchor boxes, the quantity of which can vary based on the YOLO architecture. For each anchor box within a grid cell, YOLO predicts two primary components:

1. Objectness Score: A confidence value indicating the likelihood of an object being present within the corresponding grid cell, considering the assigned anchor box.
2. Bounding Box Offsets: Rather than directly predicting absolute bounding box coordinates, YOLO calculates offsets relative to the predefined anchor box. This approach simplifies the prediction process and enhances training stability.

Additionally, YOLO assigns a probability to each object class for every anchor box within a grid cell. This allows the network to classify detected objects into predefined categories.

The Advantage of Anchor Boxes:

By utilizing a diverse set of anchor boxes with varying sizes and aspect ratios, YOLO exhibits exceptional capability in detecting objects of different shapes and scales within a single image. This flexibility is crucial for achieving robust object detection performance across a wide range of real-world scenarios. In addition to that YOLO can predict more than one object even if they share the same center. See Figure 3.9

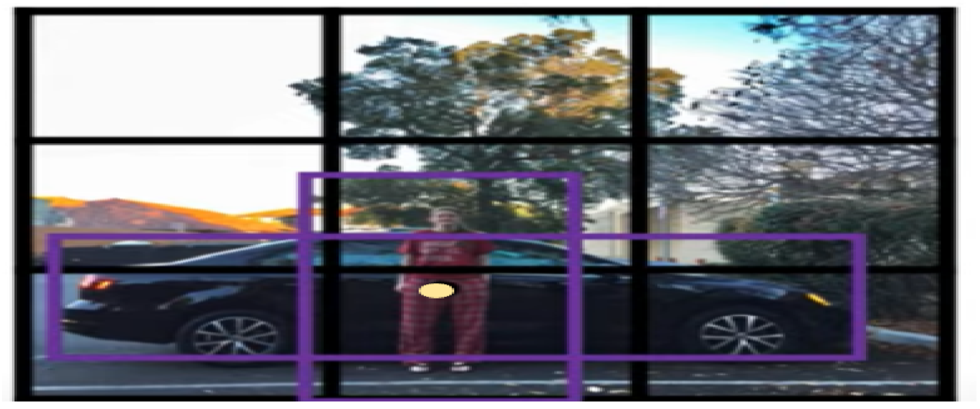


Abbildung 3.9: Two Objects Share the same Center in the same Image

3.5 System Overview:

In an effort to implement the desired goal of detecting road damages in an automated manner, in addition to allowing the relevant companies to view information about these damages and to submit their offers to carry out the necessary repairs, we began designing a website that represents this system. The system is a web application designed to provide users with robust and user-friendly tools. Operating entirely in the cloud, users can access its features from any device with an internet connection, without the need for software installations or worrying about storage limitations. The application features a clean and intuitive interface designed to be accessible to users of all skill levels, providing easy access to tools and functionalities for uploading images and checking the final results seamlessly. Users start by uploading their images selecting files from their device, with support for a wide range of image formats.

3.5.1 Users:

3.5.2 System in the background:

In this section we will try to focus on the backend design only, which will describe almost all the functionality of the system.

The system is not just about a user-friendly interface; it boasts a powerful backend that applies cutting-edge image processing techniques. This ensures the system delivers the precision and efficiency we seek and need to effortlessly achieve professional-grade results. At the heart of its blurring capabilities lies the concept of Gaussian kernels. Inspired by the natural distribution of light, these kernels create smooth and seamless blurring effects. The system doesn't stop there, though. It incorporates the YOLO algorithm, a real-time object detection powerhouse. YOLO identifies and creates bounding boxes around specific areas within the image. Such areas will describe the detected damages -if any was found- in the uploaded images. This allows for a level of control beyond simple blurring. By leveraging YOLO's object detection, the system can selectively blur areas outside the designated boxes. This intelligent approach offers several advantages: it enhances focus by directing attention to key elements, protects privacy by anonymizing background details, and reduces distractions by eliminating extraneous visual clutter. In essence, our web application combines advanced algorithms like Gaussian kernels and YOLO object detection with a user-friendly design.

What is Blurring?

Blur in image processing is a technique used to intentionally reduce the sharpness and clarity of an image, essentially softening its visual impact. This reduction in sharpness is achieved by dispersing or averaging the color or intensity values of individual pixels across a designated neighborhood within the image, which diminishes fine details and minimizes noise. The primary goal of blurring is often to make the image appear softer and smoother, reducing the prominence of sharp edges and fine textures.

Blurring techniques are particularly valuable for tasks such as Reduction of Fine Details effectively smoothing out the intricate details in an image to create a more uniform appearance, concealing imperfections and defects to improve visual quality, and decreasing noise that can detract from the main content. There are various methods to achieve blur, ranging from simple filters that average the values of neighboring pixels to more advanced approaches like convolutional method discussed before but using Gaussian kernels.

How do we achieve Blurring?

Gaussian blur is a widely used technique in which pixel values are spread according to a Gaussian function. This method produces a smooth, natural gradient that appears more organic compared to other blurring techniques. The effect of Gaussian blur is a softening of the image, reducing its sharpness and making it appear less defined. This is particularly useful in applications such as photo editing, medical imaging, and computer vision, where the reduction of sharpness can help in focusing attention on certain parts of the image or in preprocessing images for further analysis. The overall result is an image that is aesthetically more pleasing and visually less harsh than the original.

Why do we need to blur images before saving them?

As the system's purpose is to be used by people and give them the ability to take part of this process it will get harder to supervise and check all the images to see if they cover some privacy and security issues which can be faced. Some of these issues can be summarized in:

- **Privacy Protection:**
  Blurring helps protect the privacy of individuals by obscuring faces and license plates captured in Street View imagery. This helps to mitigate the risk of people being identified or tracked through our system.
- **Security Concerns:**
  Blurring can be used to obscure sensitive information, such as security features of buildings or sensitive infrastructure. This helps to reduce the potential for misuse of the information available on the system, such as Identifying Weak Points or planning crimes where blurring some details can make it harder for criminals to identify potential weaknesses in a building's security system by analyzing its layout or visible security features.
- **Balancing Transparency:**
  While blurring protects privacy, we strive for transparency. We typically blur all things and identifiable elements, which are not aimed to be used in the system.

3.6 Previous improvements of YOLO Algorithm versions for RDD:

3.7 Results

different ways to evaluate results F1-score and Mean Average Precision(mAP) Intersection over Union. To compute the mAP, we use Intersection over Union (IoU) metric. It is defined as the intersection between the predicted and ground truth bounding boxes divided by their union. The IoU values vary from 0 to 1. The high score of IoU, means more overlapping between the predicted and ground truth bounding boxes B1 and B2, respectively.

Precision: represents the proportion of positive predictions that were correctly identified.

Kapitel 4

Summary

In this thesis, we studied object detection methods and, based on the study results, we relied on the use of deep learning. Consequently, we moved on to studying deep learning and chose convolutional neural networks based on results and scientific studies. This network is distinguished by the highest classification results when dealing with matrix input. This thesis provides sufficient theoretical and practical study on convolutional neural networks, which were used in this work for object detection.

After relying on the CNN classifier, we presented a study on 2 methods for detecting images using deep learning YOLO and RCNN. Both algorithms can achieve our purpose and showed how YOLO is more effective because it can detect objects in only one step comparing with RCNN which needs 2 steps. We then presented YOLO algorithm as a solution for Road Damage Detection problem, by comparing results from different versions.

After completing the traditional method, we presented our System, which relies on best results (YOLOv9). The process begins by allowing users to upload images to the system and the system will be able to detect damages in them, and blur the background area for security and privacy reasons. Damages will be shown on a map and as detailed images with information about damages types.

Ressources

[1] M. Muster, *Beispieltitel*, ISBN: 420-69.

[2] K. O’Shea und R. Nash, „An Introduction to Convolutional Neural Networks“, 26 Nov 2015. Adresse: `https://arxiv.org/pdf/1511.08458`.

[3] R. Girshick, J. Donahue, T. Darrell und J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation“, 11 Nov 2013. Adresse: `https://arxiv.org/pdf/1311.2524v5`.