Deutsches Zentrum
DLR für Luft- und Raumfahrt

Werkzeugmaschinen
und Steuerungen
TU KAISERSLAUTERN

MASTER THESIS

Autonomous Scientific Image Segregation by Means of Deep Neural Networks

Author:	Akshay Anilkumar Girija
Matriculation number:	402426
Supervisor:	M.Sc. Tatjana Legler
Submitted to:	Prof. Dr.-Ing. Martin Ruskowski
Number of the paper:	
Date:	21. August 2020

Declaration

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources, I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Kaiserslautern, den August 21, 2020

_____

Akshay Anilkumar Girija

Acknowledgements

Foremost, I want to thank Dr.-Ing.Martin Ruskowski for providing this opportunity to write my thesis in his department of Machine Tools and Control Systems. I would also like to express my sincere gratitude to my supervisor Dr.Fabian Hampp and MSc.Tatjana Legler for their continuous support for my master thesis and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my DLR group members for their encouragement, insightful comments, and also for the support with hardware and software.

Last but not the least, I would like to thank my family for supporting me spiritually throughout my life.

Abstract

In the field of energy science especially related to combustion technologies, the impact of turbulent reacting multi-phase flows determines the efficiency and performance capability of many industrial processes. Applications such as aero engines are tested and engine operability, thermodynamic efficiency and emissions are determined by understanding the dependability of fuel spray, turbulent flow and the chemistry behind the combustion process. Insufficient atomisation of the fuel and droplet dispersion of a spray system can lead to emission fails and poor performance. For better results from the industrial spray, a deeper understanding of controlling turbulent flow and combustion chemistry are required. In this project, the physical and chemical state of the injected droplets in the latter stage are identified and detected using deep learning. Exploration of Deep learning models has made an exquisite breakthrough in the field of computer vision problems. Mask R-CNN is currently one of the states of the art algorithm in which object detection, object localization, and semantic segmentation pipelines are combined to achieve good results much faster and accurate. The main goal of this study is to implement an instance segmentation model such as Mask R-CNN on the field of combustion technology where the sprayed droplets from the fuel injector are detected and categorized according to size, shape, and solidity. The secondary goal of this study is to create a Synthetic dataset of the droplets using machine vision techniques that are similar to a real dataset. The Mask R-CNN model is used to train these datasets and the trained weights are used to evaluate on the test dataset using the evaluation metric, mean IoU. For comparison of the results, the model has been trained with different sets of synthetic image data and different hyperparameters. Finally, the trained model has been used for the statistical analysis of the real images which are in the format of im7 files.

Keywords: Combustion technology, Deep learning, Mask R-CNN, Instance segmentation, Combustion technology, Synthetic dataset

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

List of Abbreviations

NLP Natural Language Processing

AI Artificial Intelligence

CNN Convolutional Neural Network

DNN Deep Neural Network

MLP Multilayer Perceptron

MSE Mean Squared Error

SGD Stochastic Gradient Descent

Adam Adaptive Moment Estimation

RNN Recurrent Neural Network

R-CNN Region-based Convolutional Neural Network

RPN Region Proposal Networks

FCIS Fully Convolutional Instance Segmentation

ROI Region of Interest

FPN Feature Pyramid Network

FCN Fully Convolutional Network

IoU Intersection over Union

1 Introduction

Intelligence can be defined as an approach that our brain takes to solve complicated problems in real life. When we learn some skills, depending on our intelligence our brain will utilize this skill to solve the various problems. But in the early days, it was really hard for scientists to interpret how the brain functions while the whole process was deep and complex. Creating an intelligent system similar to the human brain was a challenging task during this time. But Marvin Minsky said that its possible to create an intelligent system by considering our brains as machines [MINS90]. The basic computational operations such as generating, codifying, storing, and using the information are performed by human brains. Implementation of these operations to logical machines to solve the problems is one of the pioneers for building an intelligent system. Combination of these automata theory and neuroscience had led to the proposal of artificial neurons." The perceptron" concept which was originated from these artificial neurons is considered as the first computational intelligence algorithm [PERE18]. When a computer program can learn by itself without the intervention of humans on later phases is considered as machine learning. It has been executed in bigger platforms such as data mining and big data analytics.

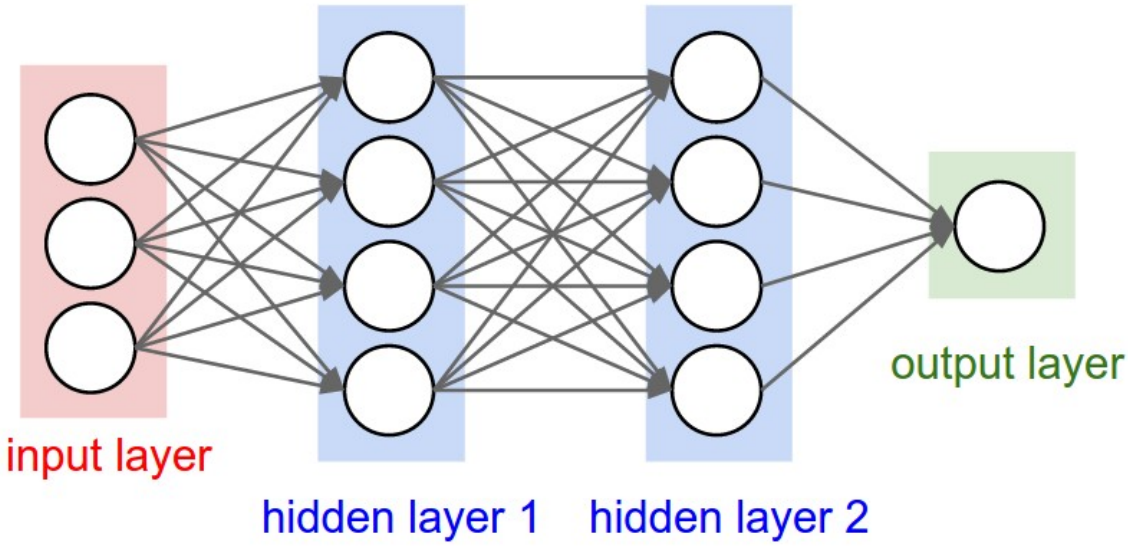

Abbildung 1.1: CNN representation consists of an input layer, two hidden layers, and an output layer.

As a different form of machine learning technique, deep learning was introduced and used to solve more complex tasks such as computer vision techniques, natural language processing (NLP), and

other computationally overhead tasks. As compared to traditional machine learning approaches, deep learning consists of neural networks which consist of many layers that are used to extract complex features from the data as data representation through its hierarchical learning process as shown in figure 2.16. As the number of layers increases, more complex data can be abstracted which makes the neural network deeper. Hence it is called a deep neural network. Therefore deep learning models can outperform traditional machine learning techniques [Wehl17]. Deep learning models have been implemented on computer vision tasks to achieve higher accuracy and for saving time. But the challenging tasks such as object detection, classification, and localization were always been a core problem respective to the application where a deep learning model has been implemented.

1.1 Motivation

Artificial intelligence (AI) has made a huge leap in the field of intelligent combustion systems. More researches have been carried out to improve AI modelling as well as algorithm optimization for these specific applications which can lead to a better economy, the safety of related combustion application [Xi19]. Advancement of convolutional neural networks (CNNs) made remarkable progress in deep learning tasks especially in the field of machine vision [Simo14]. CNN based approaches for analysing images has expanded the applicability of AI throughout many applications such as diagnosing Diabetic Retinopathy (DR) by analysing the images and classify them [Prat16] and also classification of skin lesions for the identification of skin cancer by training a CNN [Este17]. Apart from classification tasks, object localization, object detection, semantic segmentation, and instance segmentation can also be achieved using deep neural networks(DNNs). Analysing images for simultaneous detection and delineation of objects are achieved through Instance segmentation which can be produced combining object detection and semantic segmentation [Wata19]. Instance segmentation of cell nuclei from a wide range of microscopic images using Mask-RCNN performed very effectively and efficiently [John19]. Using an instance segmentation model for analysing and identifying physical and chemical states of the fuel droplets from the latter images of the industrial spraying mechanism that are obtained from the laboratory is a challenging task which can lead to further quantitative data analysis by using AI techniques in the future.

1.2 Objectives

The following are the main objectives of this work

- **Objective 1** Creation of synthetic image dataset similar to the images acquired from the laboratory using previously extracted droplet regions from the real images.

- **Objective 2** Training the Mask R-CNN model using the synthetic dataset by tuning different hyperparameters.
- **Objective 3** Evaluating the model on test dataset by using the evaluation metrics mean iou.
- **Objective 4** Statistical analysis of the real dataset

1.3 Methodology

The research method elaborates on the approaches regarding the evaluation and analysis of the droplets from the image acquired from the droplet generator. Generation of synthetic dataset and training and evaluation on this dataset will be conducted and the trained weights are used for real data analysis. Depending on performance during analysis, the hypothesis is defined whether the model should be trained on a different synthetic dataset which consists of much more complex features and with different hyperparameter tuning. One of the major problems is the dataset preparation while manual annotation of raw images takes more time which leads to a potential need for an alternative. The challenging task is to create a synthetic image similar to a real image and has to be annotated which needs an innovative approach. In the end, the trained model has been evaluated and data is formulated. The analysis of the real data by using the trained deep learning model is visualized.

2 Background

This section covers the information for understanding the concepts in the field of Artificial intelligence and its subfields such as machine learning and deep learning. An inclusion of introduction regarding the field of machine vision approaches and techniques and also about choosing a deep neural network for training is presented. The concepts of machine vision approaches are vital for understanding synthetic image generation.

2.1 Artificial intelligence

AI or machine intelligence is the development of a computer program that makes a machine intelligent similar to human beings which can be used to perform and solve complex tasks. AI focuses on building and analysing of an intelligent entity which has been designed for specific purposes. The general subfields of AI include visual perception, speech recognition, natural language processing, and logical reasoning [Nils96].

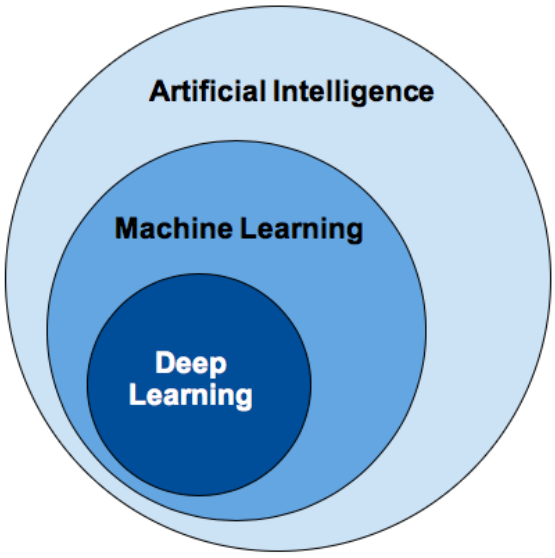

Abbildung 2.1: Relation between AI, Machine learning, and Deep learning in a glance.

2.2 Machine learning

As a subset of artificial intelligence, machine learning involves programming which makes the machine capable of learning the data automatically and performs according to each situation without explicit programming for each instance. For understanding the anomalies and patterns within the data, most of the industries are using machine learning algorithms and models. Machine learning algorithms help the model to learn from the train data and improves the learning process to achieve better outcomes while testing. There are different types of machine learning approaches which can be used depending on the data type and the quantity of the data available. They are supervised learning, unsupervised learning, and reinforcement learning [HURW18].

2.2.1 Supervised learning

In supervised learning [HURW18], the dataset contains labelled data that serve as both input and output for the mathematical model. The model is trained so that the mathematical model maps these inputs and outputs which consist of patterns or other features in the data that can be used for the further analytic process by taking input and returns a corresponding output. Classification and regression tasks are an example of supervised learning. In the classification model, the model takes input and predicts the class. An example of a classification task is to predict the class of a random fruit image from a dataset that consists of different images of fruit. Regression analysis helps to forecast the predictions by learning from historical data. For example, weather forecasting, predicting recession period can be predicted by teaching the model from the previous data.

2.2.2 Unsupervised learning

When the dataset contains a huge amount of unlabeled data, then unsupervised learning approach is considered to solve this problem. Unlike supervised learning, corresponding outputs are not available for the input data during training. Therefore, the unsupervised learning algorithms are used to find patterns, structures, or commonalities and segment the data accordingly. In this manner, when there are a huge amount of data then unsupervised learning can resolve the outcome. Some examples of unsupervised learning are spam detection in email, where the unsupervised learning algorithm could find patterns and commonalities in these emails and group the data which is accomplished by the clustering technique [HURW18].

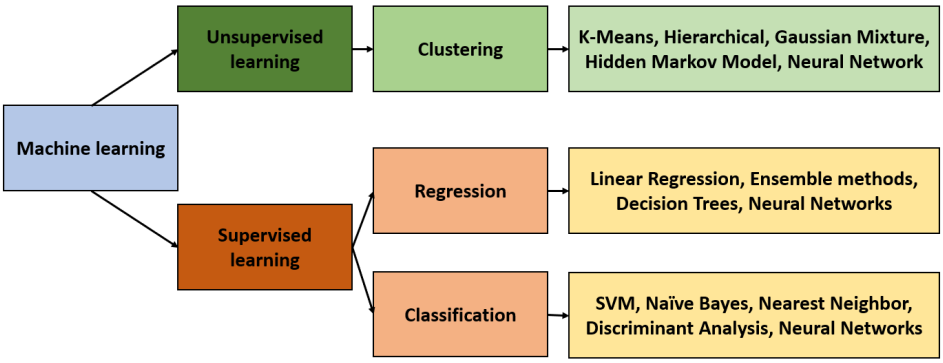

Abbildung 2.2: Supervised and unsupervised learning.

2.2.3 Reinforcement learning

In reinforcement learning, the agent should take action depends on the potentially complex environment where it is implemented. During this time the agent starts to learn the behaviour through trial and error interactions. Therefore it is also called the behavioural learning model. One of the main differences compared to supervised learning is that labelled training data is not available. There is a reward or penalty which is available depending on the action or the prediction the agent makes. The agent is allowed to learn by itself with corresponding actions it takes. Because of this, the learning and evaluation of the system take place concurrently. The environment works with the Markov decision process. The agent makes the next decision depends on the current state and

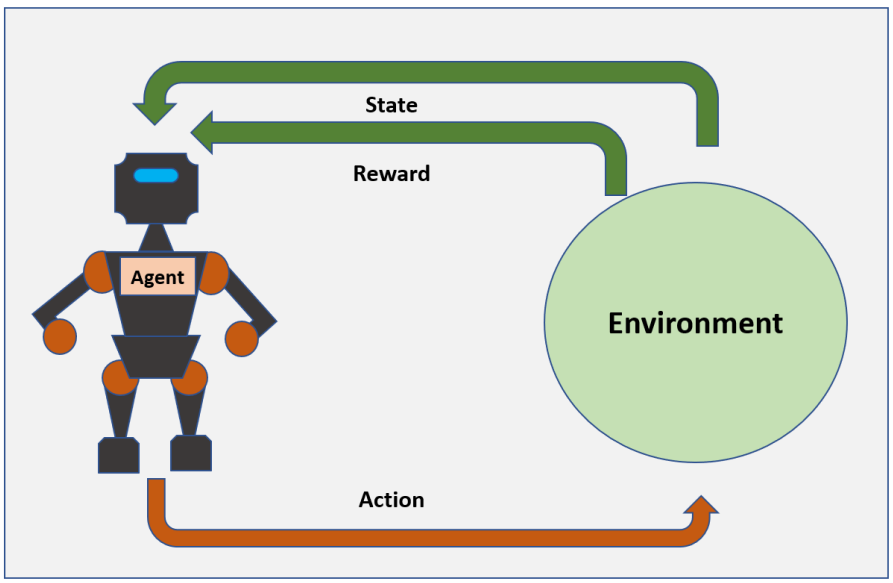

Abbildung 2.3: Reinforcement learning.

not on the previous state. When the agent makes the right decision, the rewards are added up. The goal is to maximise the rewards. Some of the applications of reinforcement learning are in

the field of robotics, industrial automation, game playing, autonomous vehicles. This shows the AI model can achieve more experience similar to human beings with the help of reinforcement learning [KAEL96].

2.2.4 Overfitting and Underfitting

A trained neural network model is expected to perform well in real-time cases. It should generalize well for the application it was intended to perform. But some times the performance of the model will be absurd. These problems occur depending on how good the model has been trained. The terms overfitting and underfitting describe the generalization ability of the trained model. Considering an example related to linear regression. Image **??** shows the model trained to have neither overfitted nor underfitted. The line shows good predictions when a new input is introduced. The distance between the points and line are not minimal.

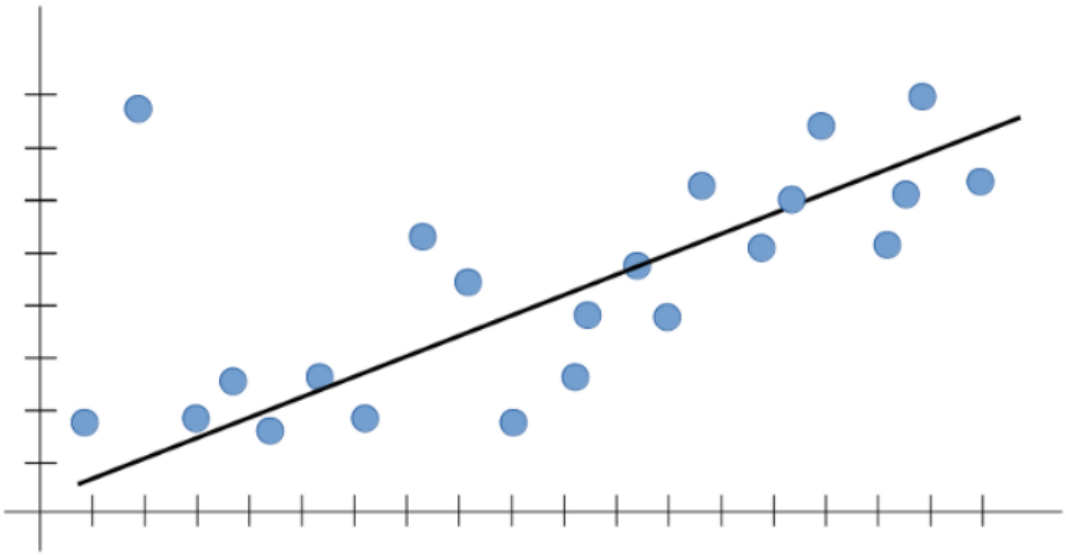

Abbildung 2.4: Generalized fitting [AL-M19].

In the case of overfitting, the line fits through all the points as shown in figure 2.5. The distance between the points and line becomes smaller and smaller after a few iterations while training. In this scenario, the model learns too much were features of both the signal and noise are learned deeply. Therefore the overall cost becomes minimum as possible. But during testing with a new dataset which lies beyond the other data points, the predicted results will be absurd. In other words, the accuracy of the training data is higher and test data is very less.

Overfitting can be avoided by:

- Using more number of data
- Removing the features which are not necessary (eg: noise).

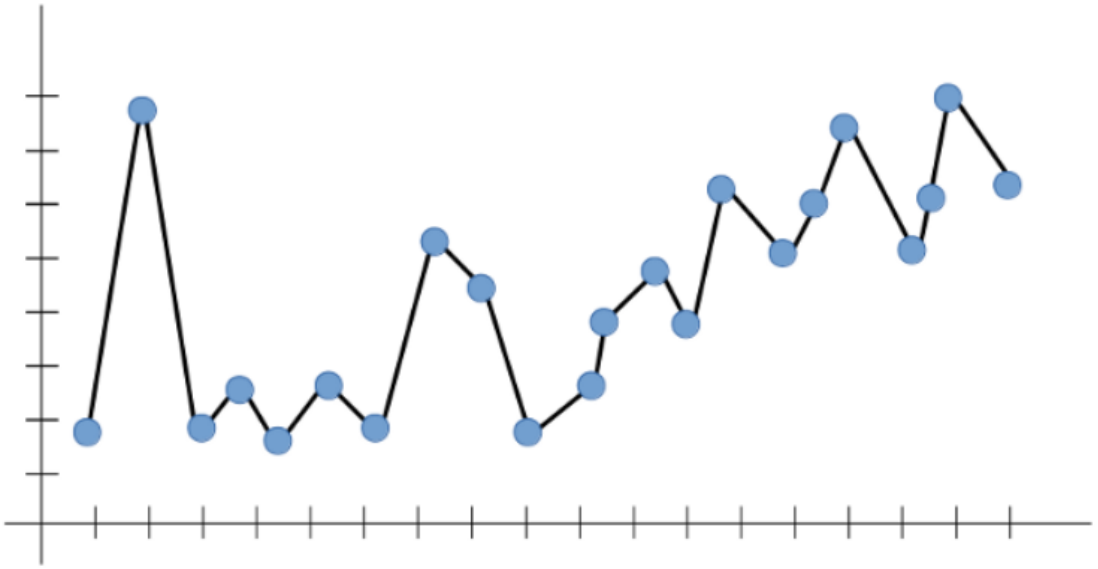

Abbildung 2.5: Overfitted Line [AL-M19].

- Bias-variance trade-off

Underfitting is a scenario where the model couldn't learn enough from the training dataset which leads to unreliable predictions. In figure 2.6, the line doesn't fit correctly across the points [AL-M19] [JABB15].

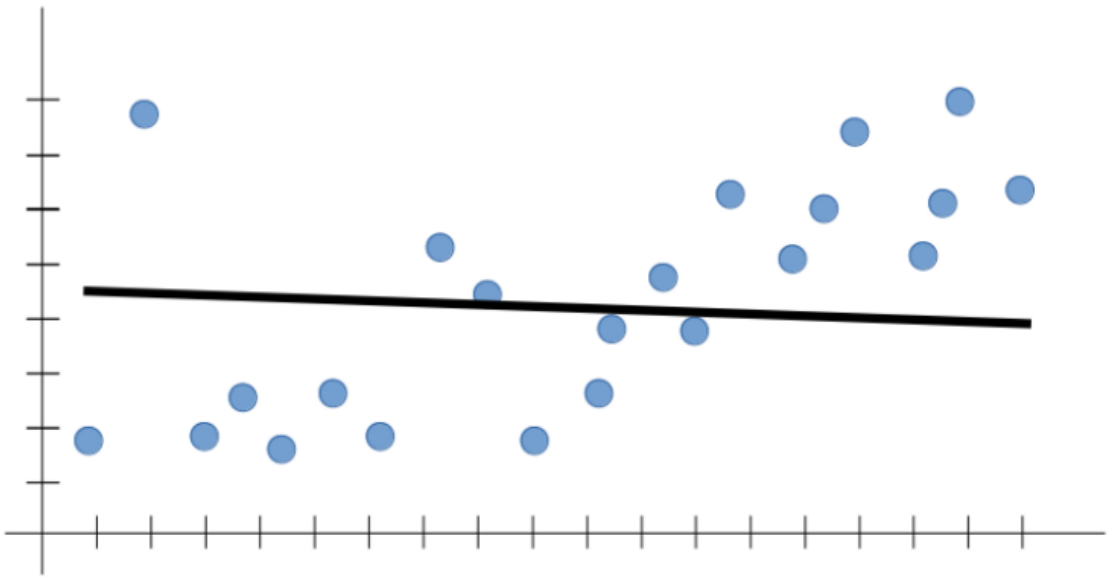

Abbildung 2.6: Underfitted Line [AL-M19].

2.3 Deep learning

Deep learning is a sub-discipline of machine learning where neural networks are introduced. The neural network consists of consecutive layers of neurons. This helps the model to learn complex features from the data in an iterative manner. To learn more complex decision boundaries from unstructured data, more layers can be included in a neural network. This makes the model deeper which includes more hidden layers for extraction of features for better learning from complex data [HURW18]. Deep learning models can learn complex information from the unlabelled and unstructured dataset by using supervised, unsupervised, and semi-supervised algorithms. The adaptive technique while training a neural network shows that it readjusts its internal structure based on the dataset [LIU17].

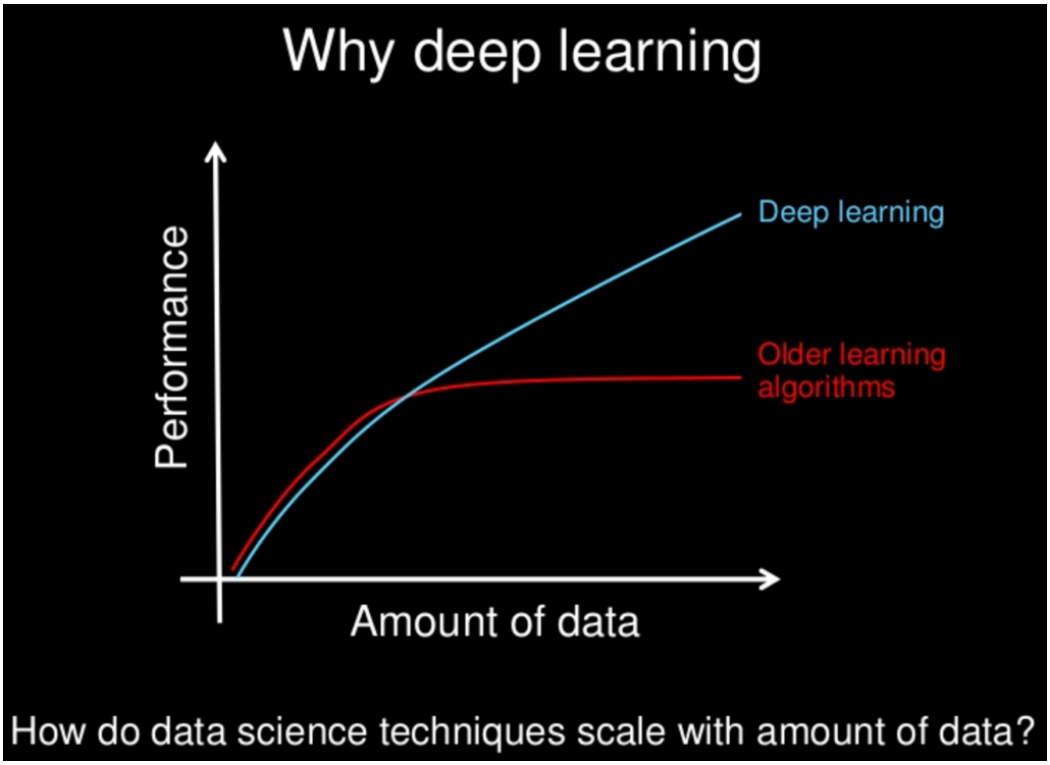

Abbildung 2.7: Performance analysis between deep learning and traditional learning algorithms for large datasets [BROW19].

2.3.1 Working of deep neural network

DNN iteratively learns from the data by adjusting the internal structure by adjusting the parameter which is called weights. Weights are numbers that are used to control the communication between the neurons. These weights are adjusted each time during training the neural network. The strength of the signal which is received at the receiver neuron is determined by weights between the giver and the receiver neurons. Depending on the strength of the signal, it can be represented as stimulating or inhibiting. Figure 2.8 represents a sample deep neural network. It consists of an

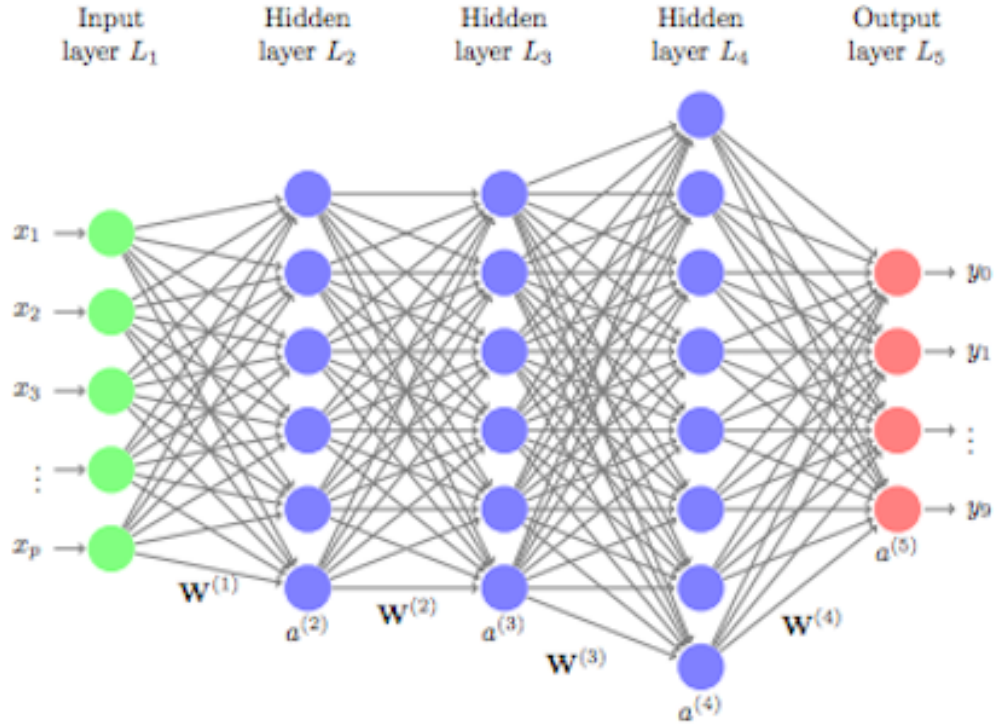

Abbildung 2.8: Sample DNN architecture [BOEH18].

input layer which is represented as Layer L_1, three hidden layers as layer L_2, L_3 and L_4, and an output layer as Layer L_5. The features of the input layer are represented as x_1,…,x_p. There are activation functions for neurons that are denoted as $a^{(i)}$. These functions act as the gateway for the signal to pass from one neuron to another. This depends on the threshold value. When the signal value is higher or lower than the threshold value from the activation function, the signal has been sent from the input neuron. The activation function can also act as a mapping function where the input signals map into the output signal [KRIE07] [SHAR17]. For a deeper understanding

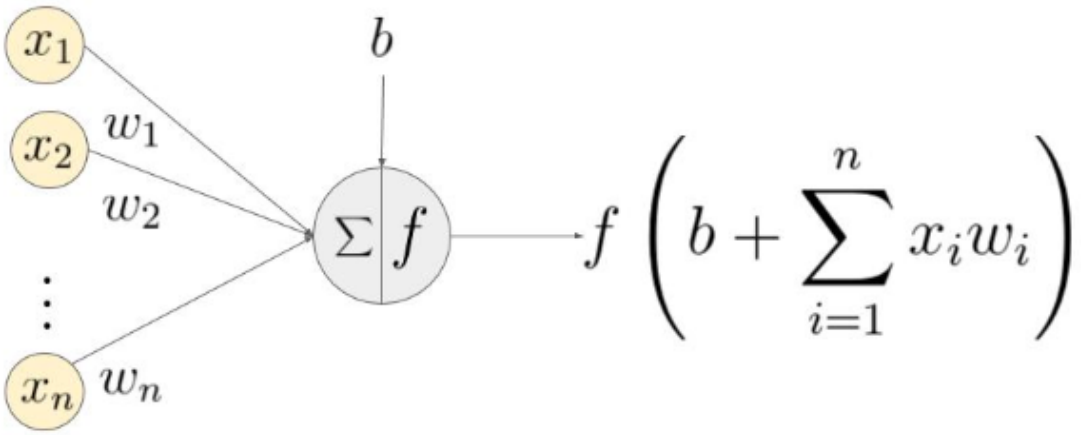

Abbildung 2.9: Sample artificial neuron [SHAR17].

of weights, biases, and activation function, considering an artificial neuron as shown in figure 2.9.

Bias is a constant which is an additional parameter that is added along with the weighted sum of the input which goes to the neuron. The corresponding signal vector x and the weight w is multiplied and added together as $x_1w_1+x_2w_2+...+x_nw_n$ and later the constant bias b is added to this as shown in figure 2.9. The activation function f is then applied to this value. The activation function transforms the linear signal which was converted by the action of weights and biases to non-linear signal. To learn the complex transformations between input and output, it is necessary to convert the linear signal to non-linear signal. Therefore choosing an activation function while creating a neural network plays an important role during training[SHAR17].

2.3.2 Activation functions used in DNN

There are two types of activation functions. They are:

- Linear activation function
- Non-linear activation function

The linear activation functions have some complication such as the function may provide the values which can be in between -infinity to +infinity. When we use a linear function throughout all the layers in a neural network, there won't be any difference such that the linear function of the first layer and the last layer will be linearly connected. The derivative of the linear activation function is constant. Therefore back-propagation is not possible. The linear activation of the consecutive layers will return different values which can be greater or lesser than the previous layer. This will continue till the end of the output layer where the final values lie in between -infinity to +infinity. To avoid this problem, non-linear activation functions are used. Some of the non-linear functions are shown below:

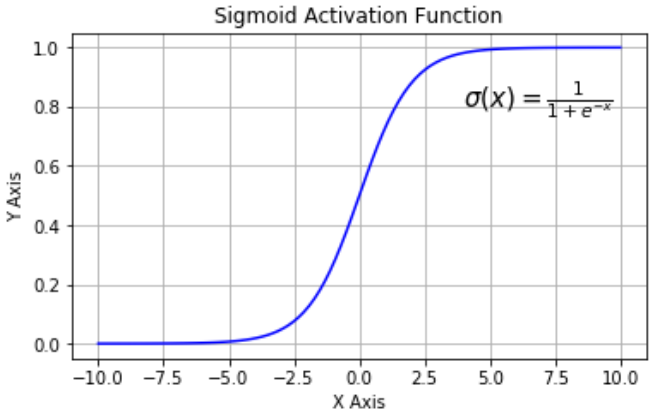

Abbildung 2.10: Sigmoid function[SHAR17].

Sigmoid functions have a smooth gradient as shown in figure 2.10. The output values are bounded between 0 and 1. But some of the disadvantages are the outputs are not zero centered and also the

prediction for higher and lower values of x remains constant which can lead to vanishing gradient problem. The equation for the Sigmoid function is given by:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

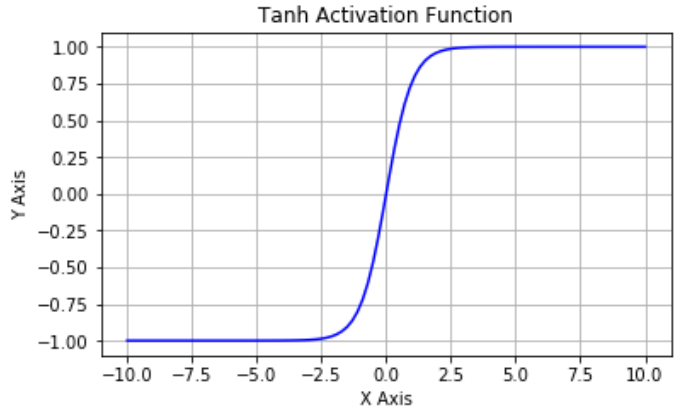

Abbildung 2.11: Tanh function [SHAR17].

The Tanh function gives out the output between [-1,1]. Since the graph is zero centered as shown in figure 2.11, depending on the input signal, they are mapped strongly negative, neutral or strongly positive values. Similar to the sigmoid function, Tanh function is also computationally expensive.The equation for the Tanh function is given by:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

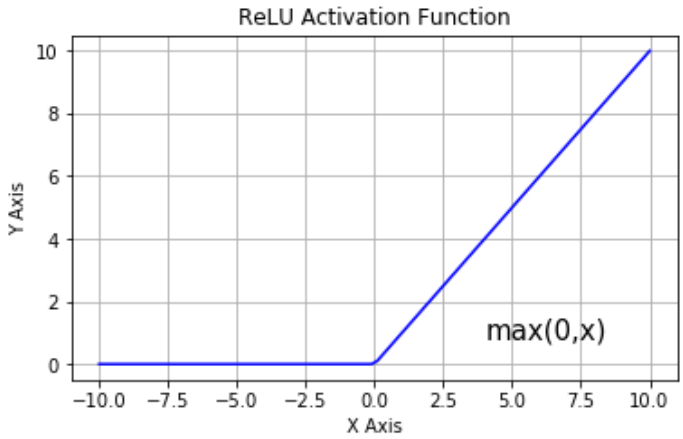

Abbildung 2.12: ReLU function [SHAR17].

The range of ReLU function is between o and +infinity. This function allows the network to converge very fast which makes it computationally efficient. But one of the disadvantages of using ReLU function is the dying neuron problem where most of the neurons become inactive while the gradient of the function becomes zero when the input becomes zero or negative. When the number of dead neurons increases, the model performance will be declined. The equation for the ReLU

function is given by:

$$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \tag{2.3}$$

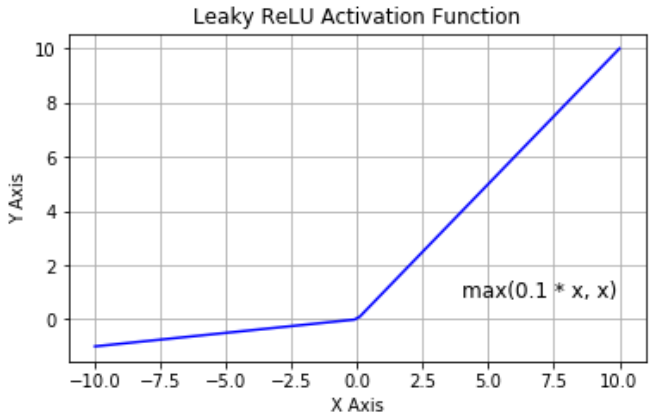

Abbildung 2.13: Leaky ReLU function [SHAR17].

Leaky ReLU is introduced to solve the dying neuron problem by a slight change on the slope of the negative x-axis which extends the range of ReLU as shown in the figure 2.13. This will resolve the dying ReLU problem as explained before. The equation for the Leaky ReLU function is given by:

$$f(x) = \begin{cases} 0.01x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \tag{2.4}$$

The non-linear activation functions can create a complex mapping between input and output where the input data are more complex such as images or videos. Moreover, the derivative of non-linear functions which are related to the inputs allows performing backpropagation effectively. [SHAR17] [RIZW18].

2.3.3 Loss function used in DNN

A loss function is essential as a performance metric during the training of the neural network. During training the model, the model predicts an output, and the predicted output is compared to the actual output which is provided along with the training dataset. Then the loss function calculates the error value by comparing the predicted output and the actual output. One of the main loss function used in DNN is:

•**Mean Squared Error(MSE)** MSE is calculated as shown below:

$$MSE = \frac{1}{N} \sum_{n=1}^{p} \sum_{n=1}^{N} (t_{pi} - y_{pi})^2 \tag{2.5}$$

Where, t_{pi} is the predicted value for data point i, y_{pi} is the actual value of the data point i and N

is the total number of data points in the dataset [SING14].

2.3.4 Optimisers used in DNN

Optimisers are used to adjust the attributes such as weights depending on the loss function by adjusting the hyperparameters such as learning rate and momentum. This will improve the training of the model by reducing the losses. Some of the commonly used optimisers [HANS19] are:

- Stochastic Gradient Descent (SGD)
- Adaptive Moment Estimation (Adam)

One of the common optimisation technique performed during training a neural network is a gradient descent. Gradient descent is applicable only for the small dataset which contains very less number of samples. To find the best fit curve a lot of mathematical computation occurs repeatedly computing the derivatives and finding the step sizes until the steps become small. In the case of large datasets having more than 100,000 samples, gradient descent is too slow.

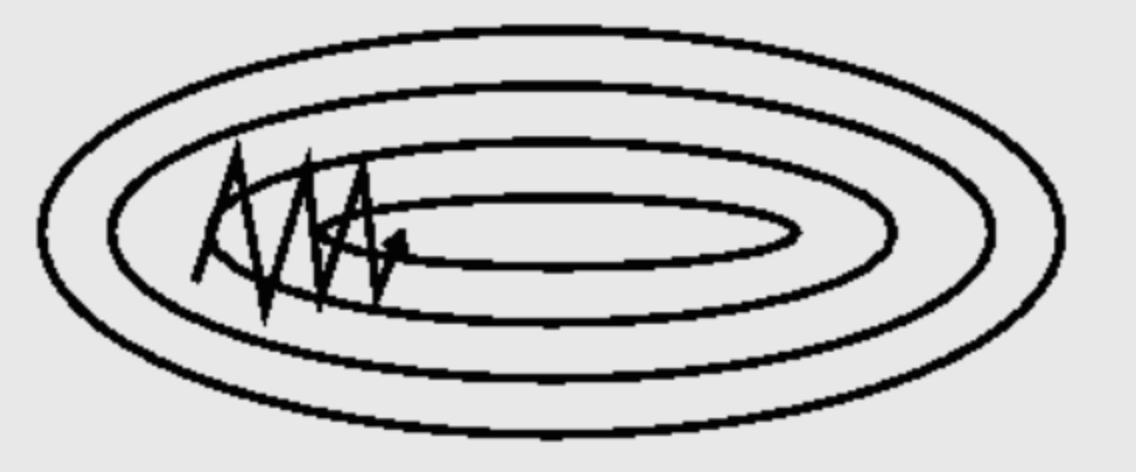

Abbildung 2.14: SGD without having a momentum [RUDE16]

Stochastic gradient descent is one of the variants of gradient descent. One of the advantages of SGD is that to make the computation process efficient and useful, computation is done on a single sample or smaller subset or mini-batch of a dataset having a large number of parameters instead of taking the whole dataset. SGD uses only one sample per step to do computation. Therefore the number of terms computed by SGD for a larger dataset will be small compared to the gradient descent. If there are redundancies in the data, SGD is useful. SGD is sensitive to the hyperparameters such as learning rate and momentum. Choosing a learning rate is important during the training process. Using minibatch, multiple samples can be used to find the best fitting curve by finding slope and intercept by performing derivation. One of the advantages of SGD is that, if a new sample is introduced, for fitting the best curve, the model doesn't need to redo all the calculations such as slopes and intercepts.

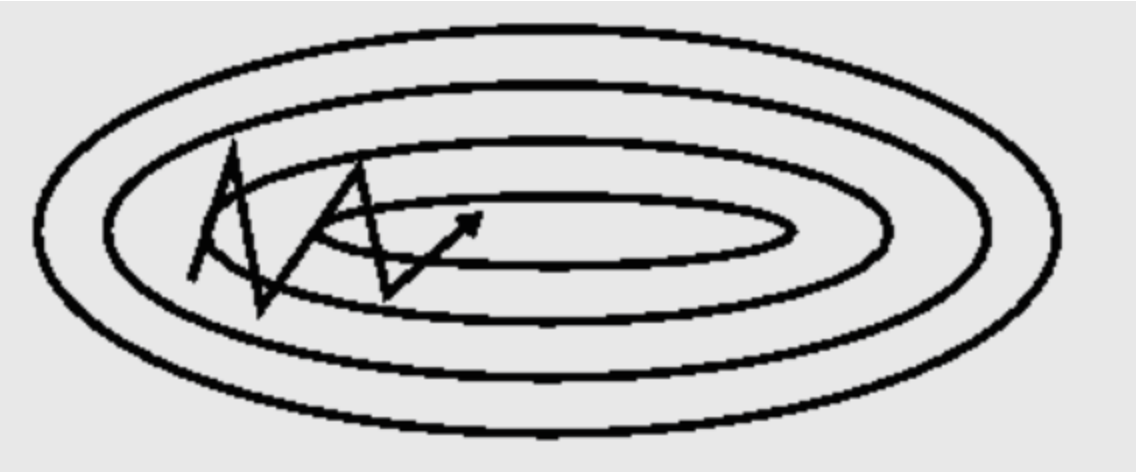

Abbildung 2.15: SGD having a momentum [RUDE16]

Momentum is crucial hyperparameter for guiding SGD without getting stuck at a local minimum. There is steep gully around the local optima, where SGD can get stuck while propagating forward. Without using momentum, the SGD oscillates across the gully slopes. Hence the movement towards the local optimum becomes slower as shown in figure 2.14. Momentum accelerates the motion of SGD. It dampens the oscillation and pushes SGD faster along the bottom to reach local optimum.Figure 2.15 shows the movement of SGD with momentum. By using a higher value for momentum, oscillations are reduced and faster convergence is achieved [KING14] [RUDE16].

2.3.5 Different types of Neural network

The table 2.3.5 shows the comparison between different types of neural networks. According to different types of the dataset [NIGA18], the architecture of the neural networks is changed to obtain good results. In a deep neural network, there are a few architectures available such as a recurrent neural network (RNN), Multilayer Perceptron (MLP) and convolutional neural network (CNN).Some of the main differences while implementing CNN and RNN are CNN is more suitable for image dataset because it is considered to be spatial data while RNN is used for sequential data. RNN uses time-series information which makes it more suitable for handling text and speech data and analyse them. When compared to CNN, the feature compatibility of RNN is very less.

Features	RNN	CNN	MLP
Datatype	Sequence data	Image data	Tabular data
Parameter sharing	yes	yes	No
Spatial relationship	no	yes	no
Recurrent connection	yes	no	no

Tabelle 2.1: Difference between RNN, CNN and MLP

2.3.6 About Convolution Neural Networks

In this project, image dataset is used. Therefore CNN is more adaptable than RNN and MLP. CNN algorithm can take an image and performs machine vision tasks such as object detection, classification, or semantic segmentation. CNN consists of convolution layers, pooling layers, normalization layers, and fully connected layers.

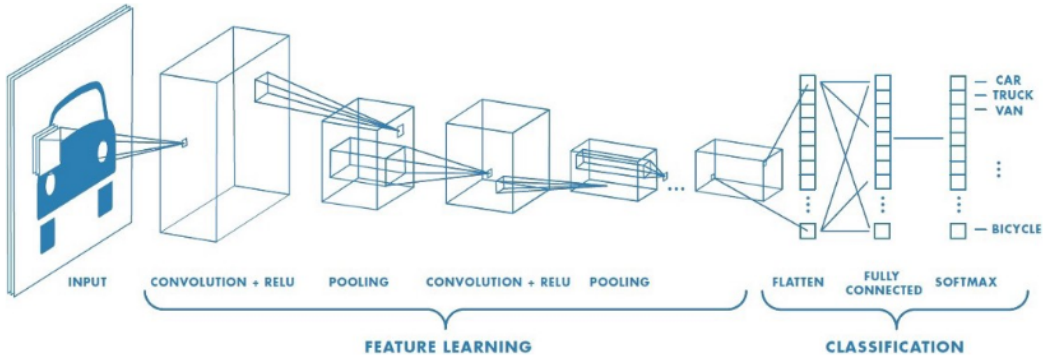

Abbildung 2.16: CNN which consists of convolutional layers, pooling layer, fully-connected layers, and normalisation layer which is softmax function [SAHA18]

Each layer of CNN performs different tasks.

The convolutional layer performs a convolution operation. Figure 2.17 shows an image of size 5x5x1. A 3x3x1 filter is convoluting on the image. The output of the image is also 3x3x1. The green

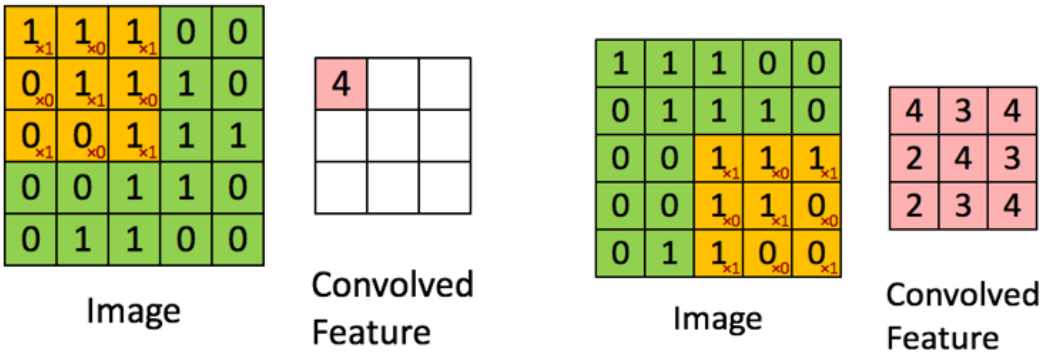

Abbildung 2.17: Convolution operation using a 3x3 kernel [SAHA18].

matrix represents the input image and the kernel is represented by the yellow matrix. The kernel has a stride movement of 1 which makes the filter/kernel move 9 times across the input image and perform a matrix multiplication operation. The convolved feature is the result of the convolution operation. The filters are used for feature extraction from images. Low-level features such as edges, colours are extracted in the first convolutional layer, and as the number of layers increases the network can perform better and recognize complex patterns with the help of more filters. Some of

the other parameters which can affect the convolved feature or the feature map are the depth of the filter, stride value, and padding. Depth is represented by the number of kernels that are used for convolution operation. The stride value represents how the kernel should convolve across the image. As per the figure 2.17, the stride value is 1. As the stride value changes, the feature map also will change. While doing convolution operation, the dimensionality of the feature map can be reduced or remain the same depending on another operation which is called padding. There are two types of padding which are called valid padding and same padding [NIGA18] [SAHA18].

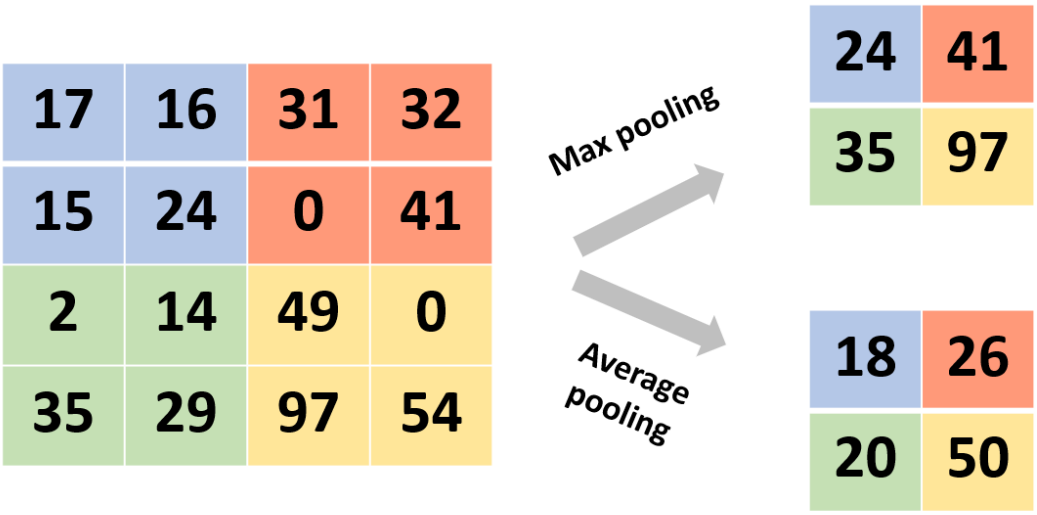

Abbildung 2.18: Max pooling and average pooling operation.

The pooling layer performs pooling operation which is also called a sample-based discretization process. It is a technique performed to reduce computational overhead by downsampling the dimension of the feature maps. It extracts much more relevant features such as positional or rotational invariant. There are three types of pooling operation which are called max pooling, min pooling, and average pooling. The figure 2.18 shows an example for max pooling and average pooling. During max pooling, only the higher values are chosen among the region covered by the filter. In figure 2.18, the filter size is 2x2. Each region covered by the filter is shown in different colours. Since max-pooling returns the higher value, it can be used to avoid noises inside the images. During min pooling, the min value among other values of a specific region is returned. Average pooling takes the average value of all the values in a specific region which is covered by the kernel.

A fully connected layer connects all the neurons in one layer to all neurons in another layer. In a fully connected input layer, the output of the previous layers is taken and flatten into a single vector. The next fully connected layer takes these single vector and predicts the label by applying the weights. Activation function such as softmax is used to get the probability score of the predicted labels [SHAB19].

Some of the commonly used CNN architectures are [SAHA18]:

•LeNEt

•Alexnet

•VGGNet

•GoogLeNet

•ResNet

•ZFNet

2.4 Transfer learning

Transfer learning is a technique where the deep learning model is used to train for a specific dataset and later using this pre-trained weights for training another dataset. Training a model from

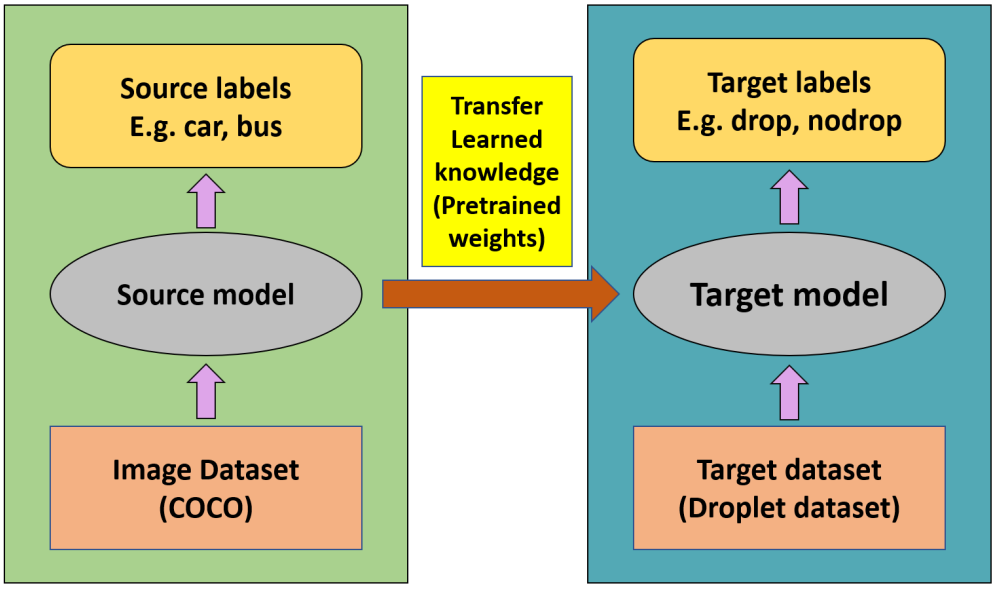

Abbildung 2.19: Pictorial representation of training from scratch and using a pre-trained weights for training.

scratch is a tedious process and it takes a lot of time to train the model for each dataset from the beginning. To avoid this problem, sometimes the model is trained on a general dataset for learning common features and later use these pre-trained weights for training different dataset for a specific application. This approach is called the pre-trained model approach [WEIS16].

2.5 Computer Vision Tasks

Computer vision tasks [BANE19] deal with the tasks related to the understanding of visual environment similar to human perception. It is one of the important fields in artificial intelligence. Similar to human beings regarding understanding and performing tasks in their subconscious mind, training a computer is much more a hectic task to achieve the same performance. But due to advanced research and development in the field of machine vision techniques using deep neural networks and advanced peripherals, these tasks are performed and really good results are achieved for different types of applications. Some of the important machine vision tasks are

- Face Recognition
- Visual Relationship Detection
- Image Captioning
- Image Reconstruction or Image Inpainting
- Face Recognition
- Object Detection
- Image Classification
- Instance Segmentation
- Semantic Segmentation

2.5.1 Object Detection

Object detection task is to detect all the objects which are present inside a single image with the help of rectangular boxes which are called bounding boxes. Through this task, the class of the object and the position of the object is also determined as shown in figure 2.20. Some of the important applications of an object detection algorithm are face detection and pedestrian detection. There are special features that are learned by the model which helps to classify each object. As shown in the figure, the coffee cup is round compared to the smartphone which is a square. Some features are unique as the shape of the object become complex.

Some of the models that are used for object detection are [LE18]

- Region-based Convolutional Neural Network (R-CNN)
- Fast R-CNN
- Faster R-CNN

Ross Girshick et al. in 2014 introduced R-CNN for object detection. Instead of the sliding window

Abbildung 2.20: Object detection task where detected objects are shown along with the bounding box[BANE19].

approach, R-CNN uses the selective search algorithm to determine the region proposals effectively as shown in figure 2.21.

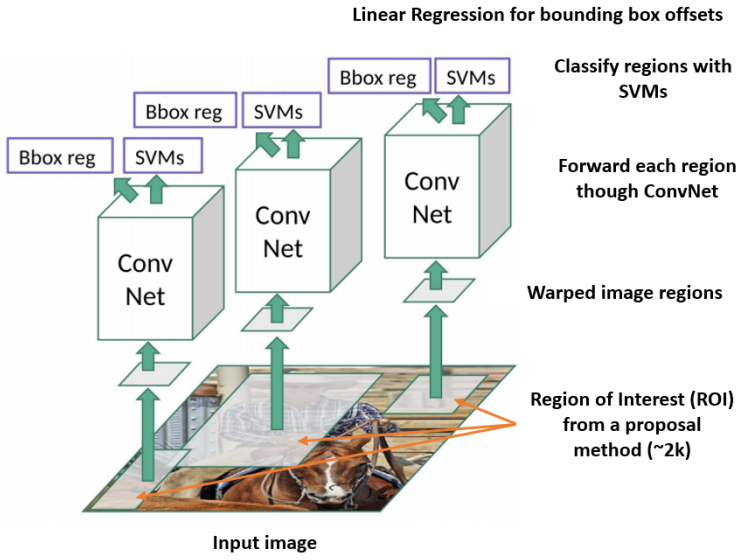

Abbildung 2.21: R-CNN representation [LE18].

In this approach, more than 2000 region proposals can be proposed and later pass on these proposals to each CNN. The output of each CNN is passing on to a support vector machine (SVM) model for classification of the object and also for finding out bounding box regressor for localization of the object. But some of the drawbacks of R-CNN is the time consumption to compute all the proposed regions and the space utilization for saving these convolved features of region proposals. Therefore the computational expense for training and analysing the data is higher. To overcome this problem Fast R-CNN is introduced.The figure 2.22 represents Fast R-CNN architecture.

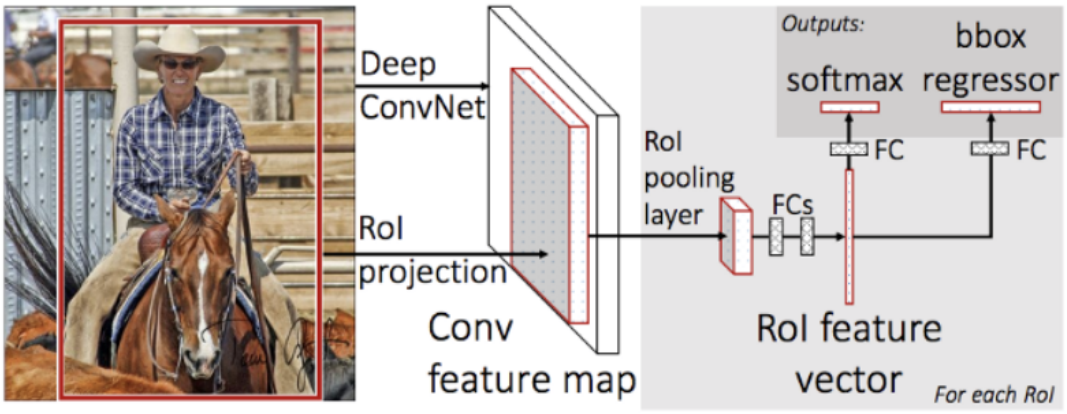

Abbildung 2.22: Fast R-CNN representation [Abdu18].

Instead of talking each region proposals separately to CNN architecture, it takes the image as a whole and the region proposals in one forward propagation. The SVM model is replaced by a softmax layer for better accuracy and speed. But still, due to the selective search algorithm, the time taken during detection is much higher.

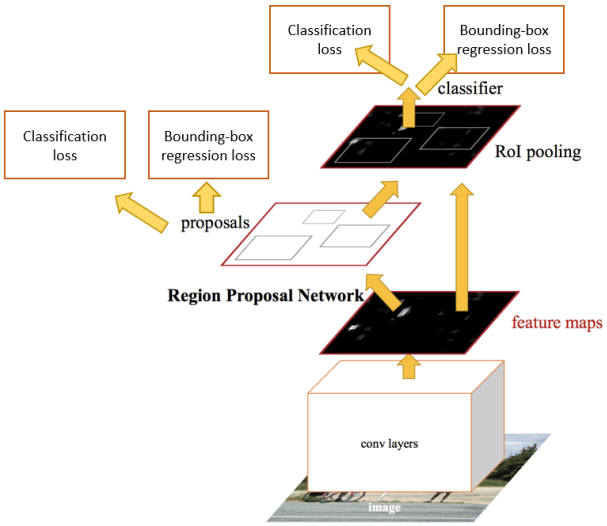

Abbildung 2.23: Faster R-CNN representation [Le18].

Faster R-CNN is much more advanced compared to Fast-RCNN. To avoid the bottleneck created by the selective search algorithm in Fast R-CNN, Region proposal network (RPN) is introduced in Faster R-CNN. The figure 2.23 shows the architecture of Faster R-CNN. In this model, the input image is fed to the backbone network such as Resnet or other architecture to obtain convolved features. By using RPN, the anchors are predicted with the help of these feature maps. Thus RPN reduces overall computational overhead by deciding where to search inside the image during analysis. The proposals from the RPN are fed to the classification and regression layer for performing

classification and localization task [LE18].

2.5.2 Image Classification

During image classification tasks, the whole image is considered for prediction and it is classified into its corresponding class label. The classes are predefined by the user for each input images while training the model.

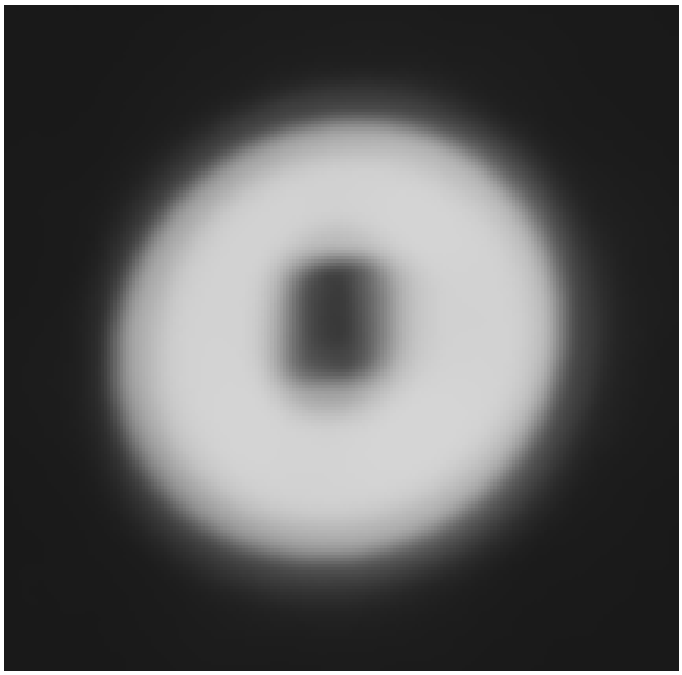

Abbildung 2.24: Image of a droplet which can be used for the classification task.

2.5.3 Semantic Segmentation

Semantic segmentation is also known as pixel-wise segmentation where the task is to identify different objects in the image and provide a pixel-wise representation for each class. This task is also known as the dense prediction [JORD18b]. The figure 2.25 shows an example of a semantic segmentation task.

During semantic segmentation, each pixel of the image is assigned to a certain class. If there are more objects which belong to the same class, semantic segmentation cannot distinguish between these objects. Some of the main application of semantic segmentation are:

- Autonomous vehicles
- Medical image diagnostics

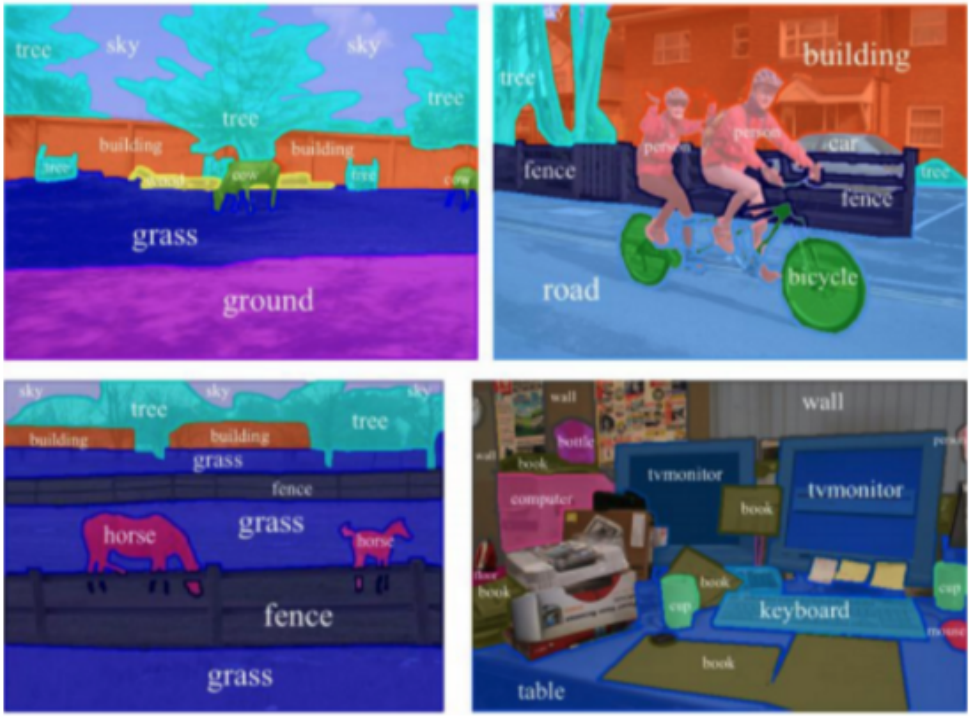

Abbildung 2.25: Application of semantic segmentation in different scenario [BANE19].

During semantic segmentation, An RGB image or a grayscale image can be used as an input and the segmentation map is returned as an output. This segmentation map consists of pixel-wise

Abbildung 2.26: Segmentation map representation [JORD18b].

information regarding where and to which class each pixel belongs to. The figure 2.26 represents an example of semantic segmentation. For understanding the image the author has provided a low-resolution map corresponding to the image while in reality, the resolution of the segmentation maps is equal to the input image. The segmentation map is created by collapsing multiple output channels which are the predictions created by a one-hot encoding process for each class. Each prediction for each class can be represented as a mask [JORD18b].

2.5.4 Instance Segmentation

Instance segmentation is the combination of other machine vision tasks such as object detection and semantic segmentation. Instance segmentation models are used to predict each instance separately along with the pixel-wise segmentation. The figure 2.27 shows an example for instance segmentation.

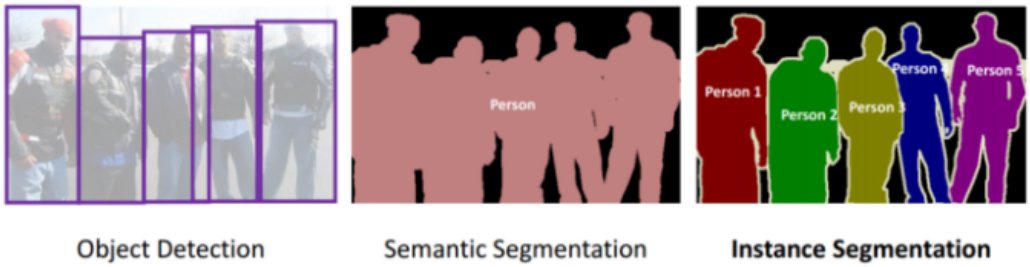

Abbildung 2.27: Instance segmentation [LIU20].

One of the current state of the art for instance segmentation is the Mask R-CNN. The basic functionality of Mask R-CNN is to perform object detection and then followed up by the segmentation approach [LIU20].

2.6 Previous works

As a prior development that led to the instance segmentation model, the concepts of bounding box object detection-based Region-based CNN (R-CNN) were implemented which uses pre-trained CNNs for extracting the image features by performing a forward computation. Because of the slow speed, many follow up alterations have been made to speed up the evaluation and higher accuracy. Faster R-CNN was one of the follow-ups by replacing the selective search method for attention mechanism with Region Proposal network [REN15]. Many approaches have been experimented to achieve instance segmentation by combining object detection and semantic segmentation techniques. Deep Mask is an example for instance segmentation. Deep mask was slow while the segmentation and recognition tasks were not executed parallelly and the performance was less accurate [PINH15]. For achieving faster performance without compromising the accuracy, the approach was to make the model simpler and more flexible by parallel prediction of masks and class labels. Li et al [LI17] introduced fully convolutional instance segmentation (FCIS). But the accuracy of the model was declining during the evaluation of overlapping instances. Mask R-CNN follows the basic structure of Faster RCNN and introducing a fully convolution layer for locating the objects at the pixel level. Instead of ROI pooling, the ROI alignment layer is used which helps to retain spatial

information through bilinear interpolation of the feature maps. For producing instance segmentation, a fully convolutional neural network is added as a network head as shown in figure 2.28. Mask R-CNN is based on ResNet-FPN backbone which indicates a feature pyramid network (FPN) which uses ResNet-50 or ResNet-101 [He17].

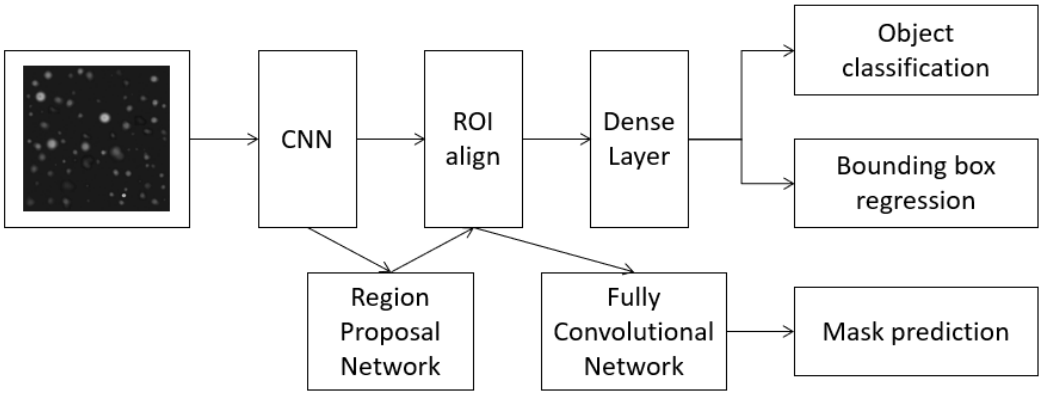

Abbildung 2.28: Mask R-CNN representation

2.7 Mask R-CNN

Mask R-CNN consists of 2 stages. Figure 2.28 represents the Mask R-CNN structure. The input image is taken and analysed. With the help of region proposal networks, anchors/proposals are generated. In the second stage of the architecture, the proposals were classified and bounding boxes and masks are generated. The backbone structure plays a crucial role in these two stages [He17][Abdu18].

2.7.1 Backbone network and style

The backbone network used in Mask R-CNN is a CNN which can be Resnet or VGG combined with feature pyramid network (FPN). Usually, pretrained weights from Resnet model are used for the training of a custom dataset. These convolutional neural networks are used for extracting the features from the input image. In the initial layers of CNN, small features such as edges and corners are extracted. These are called low-level features. The deeper layers will extract high-level features such as the complex shape of the actual object. In deep neural networks, as the architecture goes deeper, the network struggles to train effectively while it will come across a problem call vanishing gradient. During backpropagation, the gradient is sent back to the previous layers and further mathematical computation such as multiplying these gradients repeatedly will make this gradient small. Therefore the performance of the model could decline rapidly or achieve satura-

tion while training. To avoid this problem, Resnet consists of a residual block which works as an identity shortcut connection. Figure 2.29 shows a residual block that consists of a skip connection between multiple layers to avoid vanishing gradient problem.

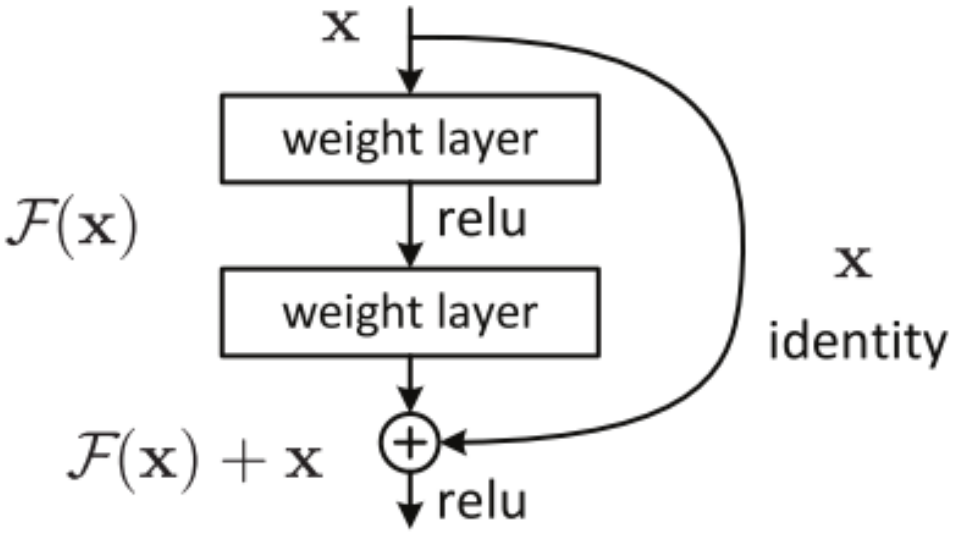

Abbildung 2.29: Residual block representation [HE16]

This identity shortcut connection performs identity mapping. As in the figure, the identity "x" which is the output of the previous layer is added to the output of the next layer "F(x)" through skip connection [HE16]. FPN style consists of bottom-up, top-bottom approach which consists of lateral connection as shown in figure 2.30.

Abbildung 2.30: FPN architecture [ABDU18].

As in the figure 2.30, the bottom-up stage of the FPN is performed by Resnet for feature extraction from the input images. The feature maps are shown as normal blue outlines. The resolution of the image decreases from bottom to top but the feature extraction is low at the bottom and strong at

the top. The feature pyramid maps are generated during the top-bottom path. With the help of lateral connection which is convolution and addition operation, strong features with less resolution from the top-bottom stage are converted into semantically strong features as represented as thicker blue line as shown in figure 2.30 [ABDU18][KUMA19].

2.7.2 Mask R-CNN Stage I

In the first stage of Mask R-CNN, a lightweight neural network called region proposal network (RPN) is used to generate proposals that represent the region of interests (ROI) from the convolved features. RPN works during the top-bottom stage of FPN and generates proposals. The feature maps are scanned in such a way that, there are boxes with specific dimensions that are placed on already known locations which are called anchors. There are around 200K anchors of different dimensions and aspect ratios that covers most of the image by overlapping. RPN uses a sliding window feature which is a convolutional operation on the feature maps to obtain relevant anchor boxes. These anchor boxes undergo binary classification such as foreground (fg) which consists of an object or background (bg) and bounding box regression to determine the class as well as to refine bounding boxes. Each anchor has an Intersection over Union (IoU) value which determines whether the anchor has a positive or negative label. If the anchor box for the foreground has higher IoU value when overlapped with the actual ground truth box, the anchor can be classified as a positive label. The threshold IoU value can be changed. When there are multiple anchor boxes which has positive labels, the anchor which has the highest foreground score is chosen and send it to the next stage while the rest of the anchor boxes are avoided. This technique is called non maximum suppression. The advantage of using RPN is that it reuses the feature maps efficiently and also to avoid duplicate calculations [ABDU18][KUMA19]. Non maximum suppression [HOSA17] acts as a filter to choose the most appropriate anchor and discard all other anchors depending on the threshold value. The figure 2.31 represents the non maximum suppression technique.

- Out of the limited number of bounding boxes, only one bounding box which is having the highest confidence score is chosen.
- The Intersection over Union value is calculated between the chosen bounding box and the rest of the bounding boxes one by one.
- A specific value for IoU is chosen to be the threshold and all the calculated IoU values are compared to this threshold value.
- This process is repeated until all the bounding boxes are suppressed leaving only one adequate bounding box is left behind.

Abbildung 2.31: Using non-max suppression for choosing the best anchor box [HOSA17].

2.7.3 Mask R-CNN Stage II

In the second stage, the chosen ROI is used to extract feature maps for performing classification, bounding box regression, and segmentation mask. This can be done in two ways

- ROI pooling:

The region of interest will be chosen from the provided feature map by cropping accordingly. After cropping, the size of the feature map will be decreased. Figure 2.32 shows an example of an ROI pooling. In the figure 2.32, the feature map is of size 8x8 and the ROI has a size of 7x5 which is situated at the bottom left corner.

The ROI pooling layer which follows max pooling or average pooling gives an output of a 2x2 feature map as shown. Here max pooling is considered as an example. By taking this output, the fc layer performs the classification of the objects and also refines bounding boxes for each class [KUMA19].

- ROI Align:

One of the drawbacks during performing ROI pooling is that the entire process is quantized. For predicting class and refine bounding boxes, ROI pooling does not have any negative impact. But for generating a segmentation mask, ROI pooling does not function as expected. For fixing this problem, ROI align is introduced.

ROI align can preserve spatial information whereas, in ROI pooling, this finer spatial information is

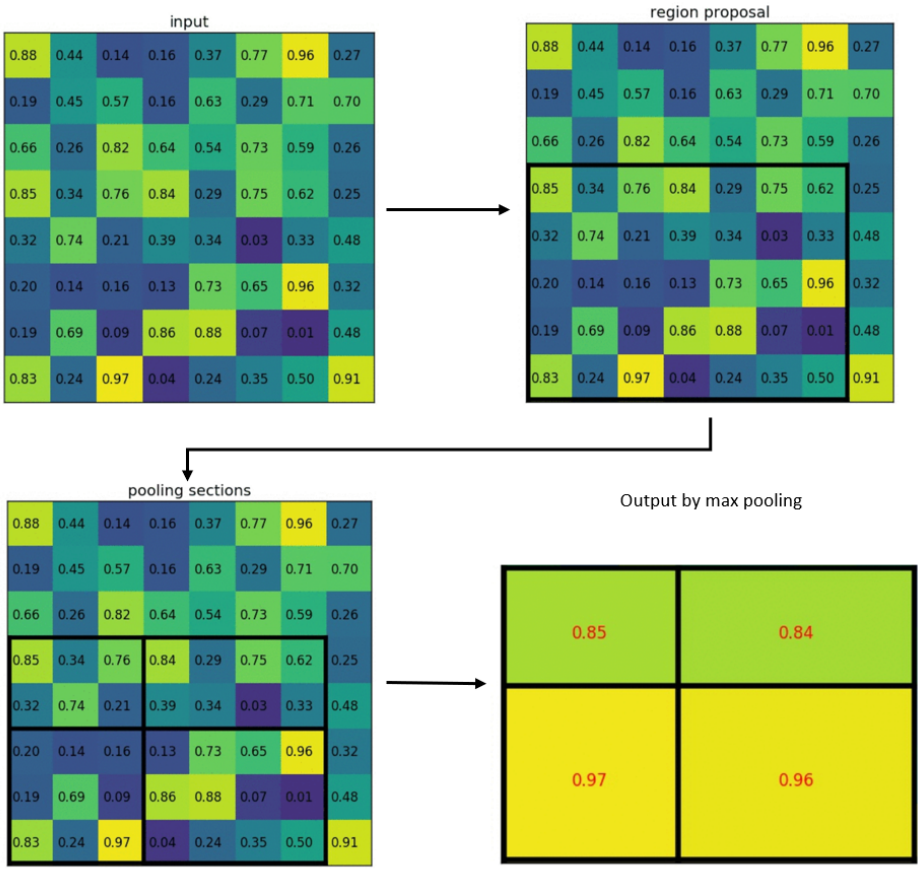

Abbildung 2.32: ROI pooling [KUMA19].

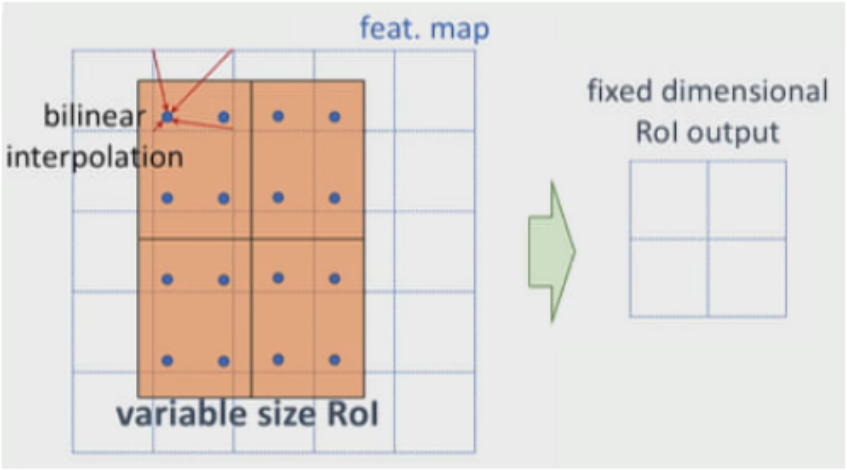

Abbildung 2.33: Pictorial representation of ROI align [KUMA19].

inaccurate and misaligned. ROI align performs binary interpolation as shown in figure 2.33 instead of pooling operation to create the feature maps. Mostly in the second stage of Mask R-CNN, ROI pooling is substituted by ROI align [KUMA19] [HE17]. The positive output from the ROI align is taken and fed to a fully convolutional network (FCN) for generating a segmentation mask.

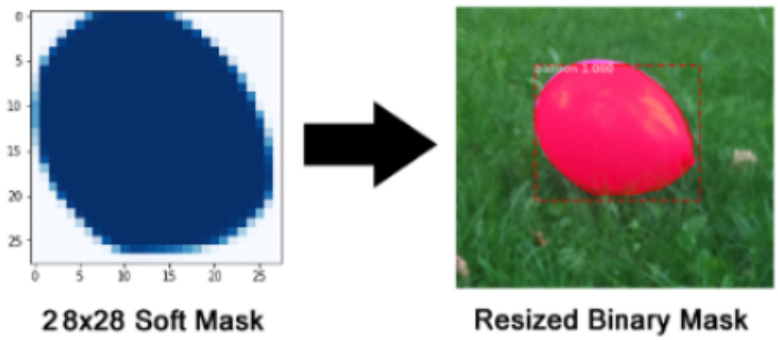

Abbildung 2.34: Segmentation mask [ABDU18].

Figure 2.34 shows a 28x28 soft mask generated for an object.The generated masks are having a low resolution. But since these masks are holding float numbers instead of binary numbers, more information is stored. The ground truth mask is scaled down to the soft mask dimension to calculate the loss and while inferencing the predicted mask is resized to the original dimension of the ROI bounding box and generates segmentation masks [HE17][ABDU18].

There are 5 losses while training the model, they are:

•**rpn_class_loss,** L_{cls1} **:** RPN (bbox) anchor binary classifier loss

•**rpn_bbox_loss,** L_{bbox1} **:** RPN bbox regression loss

•**fastrcnn_class_loss,** L_{cls2} **:** loss for the classifier head of Mask R-CNN

•**fastrcnn_bbox_loss,** L_{bbox2} **:** loss for Mask R-CNN bounding box refinement

•**maskrcnn_mask_loss,** L_{mask} **:** mask binary cross-entropy loss for the mask head

The loss function of Mask R-CNN can be represented as:

$$L = L_{cls} + L_{bbox} + L_{mask} \tag{2.6}$$

Where L_{cls} is the classification loss represented as (L_{cls1} + L_{cls2}). L_{bbox} is the bounding box loss represented as (L_{bbox1} + L_{bbox2}) and L_{mask} represents the mask prediction loss. Classification loss and bounding box loss shows how well the model performs during classification and localization. By performing binary cross entropy between ground truth mask and predicted mask, mask prediction loss can be calculated [KUMA19].

2.8 Evaluation metrics

Evaluation metrics are used to measure the quality of DNN model. Some of them are mentioned below.

- Intersection Over Union

Intersection over union also called Jaccard Index is a metrics system that can be implemented directly using TensorFlow. IOU is calculated by the division of the area of intersection between the predicted segmentation image and the ground truth image (A to the union of the predicted segmentation image and the ground truth image (B) [JORD18a]. It can be represented as

$$IoU = \frac{|A| \cap |B|}{|A| \cup |B|} \tag{2.7}$$

For calculating Mean IoU, the groundtruth, as well as the prediction mask, has to be generated as input for the function provided by TensorFlow. To generate ground truth as a ternary image, first, the image has been visualized by using the segmentation boundary information for generating the masks, and also the class id which is stored in the JSON file which will be explained in section 3.1.2. After visualization, the region has been saved as an image with ternary values of 0,1,2 according to the class ids give to each class and also the same dimensions such as height and width. The predicted output is also saved as a ternary image . Figure 2.35 represents the ground truth and predicted mask of a sample image from the synthetic dataset.

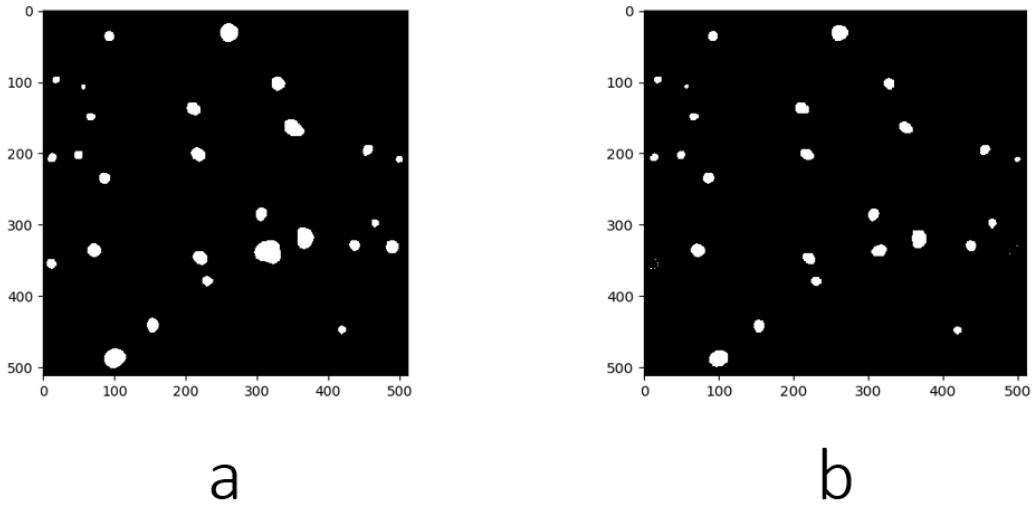

Abbildung 2.35: Representation of ground truth mask (a) and predicted mask (b).

- Pixel Accuracy:

Pixel accuracy is represented as the percentage of pixels in an image that is correctly classified. Pixel accuracy is calculated by

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

Where TP denotes true positive which is the actual number of pixels that are correctly classified as the corresponding class according to ground truth mask. TN represents true negative which is the

actual number of pixels that are correctly predicted that does not belong to the corresponding class according to ground truth mask. FP indicates false positive, which is the actual number of pixels incorrectly classified but belong to the corresponding class and FN denotes false negative in which the predicted pixel values are incorrectly classified and also does not belong to the corresponding class [JORD18a].

3 Implementation and Approaches

In this project, for understanding the droplets that are sprayed using a fuel injector, an instance segmentation model in which Mask R-CNN is chosen throughout the experiment. 3 different synthetic datasets are generated and used for training throughout the experiment. The synthetic images differ in the resolution and number of objects placed in each image. Finally, the best model will be chosen while training each dataset and this model will be evaluated using an evaluation metric mean iou. Furthermore, an analysis is conducted on the raw data using these models to understand the statistics of the real data.

3.1 Synthetic dataset generation

The real images are obtained in the laboratory using the shadowgraphy technique. From these images, the droplets are acquired using computer vision-based object detection algorithm (CVOD) using python. Preprocessing includes normalization, thresholding, denoising. The final ROIs are shown in 3.1. These ROIs are sorted and saved into different folders inside the database. Depending on the physical parameters such as solidity, shape, size the droplets are saved as regular droplets or irregular droplets. CVOD algorithm will be changed accordingly to the complexity of the raw images.

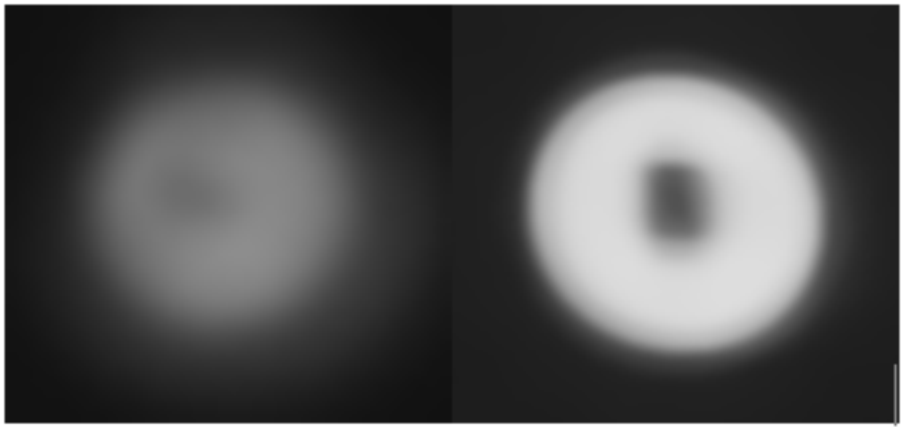

Abbildung 3.1: ROI which shows irregular droplet on the left and regular droplet on the right

The synthetic dataset consists of 2 parts.

•Synthetic Images

•JSON files

3.1.1 Synthetic image

Synthetic images are generated based on the computer vision algorithm with a random number of objects placed on a background image having similar noise compared to the real images. From the database, an image has been selected depending on the random probability that was manually assigned. The probability ratio of choosing the regular droplets to the irregular droplets are assigned to 7 : 3. The number of objects is defined with a random number generator between 0-150 or 0-300 per synthetic image. Once an image is chosen, it is passed through a data augmentation function which does data augmentation which produces diverse images. There are two functions defined for irregular droplets and regular droplets. If an irregular droplet has been chosen, it undergoes the smoothing technique, where the gradients are smoothened using a Gaussian filter. The standard deviation of the gaussian kernel is chosen randomly within a range of values. The probability of an irregular image to undergo this data augmentation function is 20 %. Since the image is already distorted, it is not necessary to perform smoothing augmentation on all irregular drop images. If a regular droplet has been chosen, first the histogram of the image has been plotted which represents the frequency distribution of the data. 3.2 shows the histogram of a regular drop image. Histogram returns two values such as values of the histogram and the bin edges. It represents the pixel count on the x-axis and pixel values on the y-axis. The pixel value is chosen from a specific bin which consists of more general pixels values. The average of these pixel values is calculated in prior and saved in an empty array which is later used to create noise value for the generated background image while to avoid the sharp edges of the augmented droplet images while placing it on the background image of size 512x512 or 1024x1024 resolution. This technique used to avoid sharp edges while pasting drop images on to the background. The augmentation techniques that follow for regular droplets are the image is rotated with a random angle which lies between 0 to 360. The rotated image performs smoothing technique using Gaussian filter as explained earlier. Not all the rotated images perform smoothing technique. Later the returned image from the previous augmentation carries out resizing operation. The height and width during resizing will be randomly generated but does not cross the limit, while the droplets are meant to be average-sized or much smaller in the real image. The output of the augmented image is later passed through another function which defines the category/class label for these droplets. Currently, there are only two classes are defined. They are “drop” and “nodrop”. Using the histogram of the model, the minimum and maximum values of the pixel values are found out. Later these values are used to normalize the actual image by subtraction and division operation. Some of the

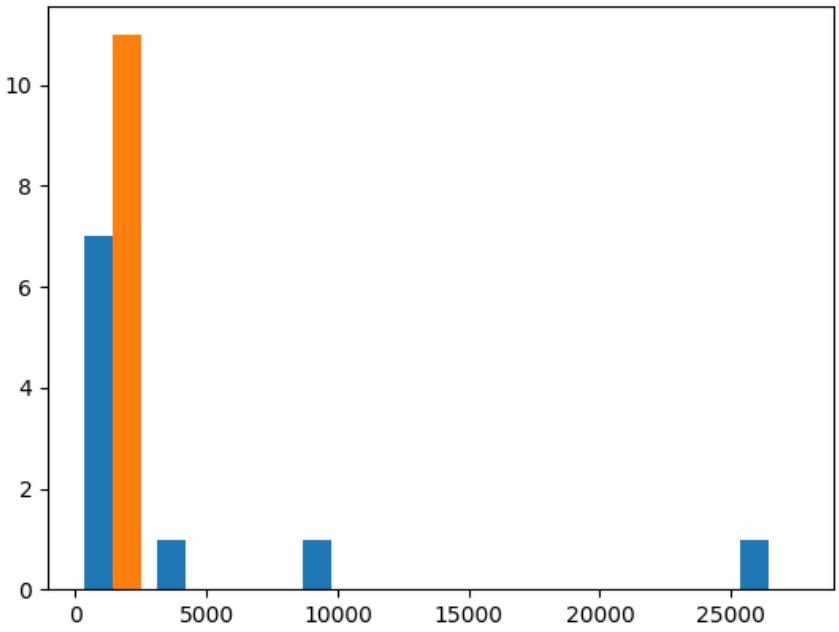

Abbildung 3.2: Histogram representation of an ROI from the database

key parameters regarding the size and intensity of the pixel values, conditions are specified for labelling whether the input image belongs to “drop” or “nodrop” category. This function will return the label of the image. The augmented image is passed on to the next procedure where it is placed randomly on the predefined background as shown in 3.3 An offset is generated which is chosen

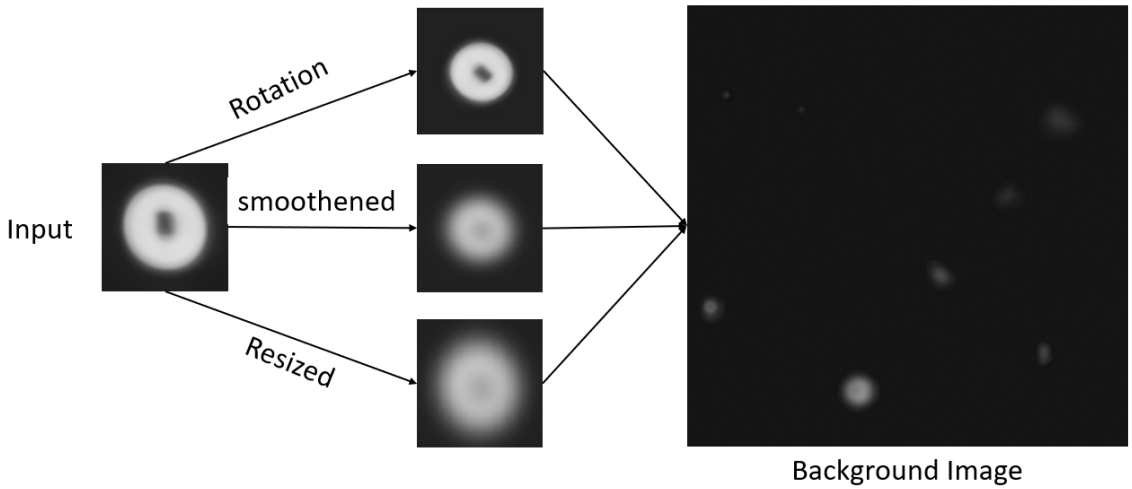

Abbildung 3.3: Placing augmented regions on the background image.

randomly within the boundary of the background image. This will be the initial coordinates for the augmented images which have to be placed. A mask image with value zero is generated which has a similar resolution of the background image. Once the image has been placed, there will be a value in the mask image instead of zeros. When a new augmented image is introduced, the new

calculated random offset will be compared with the previous offset value saved in the mask image so that the images won't get overlapped to each other. A counter is provided so that when the background image gets crowded with many droplet images, there won't be any place to paste the image in the background. Once all the objects are placed on the background, the rest of the spaces are filled with the mean value of the pixel values which are stored previously on an array from the histogram. While generating synthetic images, the label of the object, position of the object which includes bounding box as well as the contour of the object are also saved and written parallelly on the JSON file which is explained in the next section. The contour of the object is saved by performing threshold operation on the augmented object and saving the remaining pixel values on a mask image of a similar size of the object image filled with zeros. Coordinates along the side if these pixel values are noted and the actual offset/coordinates of the background image are added or subtracted to obtain the actual contour with reference to the coordinates of the background image. While creating synthetic images, the total number of synthetic images are predefined. After generating a single image, the program saves this image to respective folders which are train, validation and testing. The probability of saving these images to the train, validation and test folder are 50:30:20 and 60:30:10 for dataset 3. In this study, 3 types of synthetic datasets are generated. The first two datasets are created with regular and irregular droplet images while the third dataset is created with complex objects which consist of distorted droplets. The 3.4 shows the difference between the regular dataset and complex dataset.

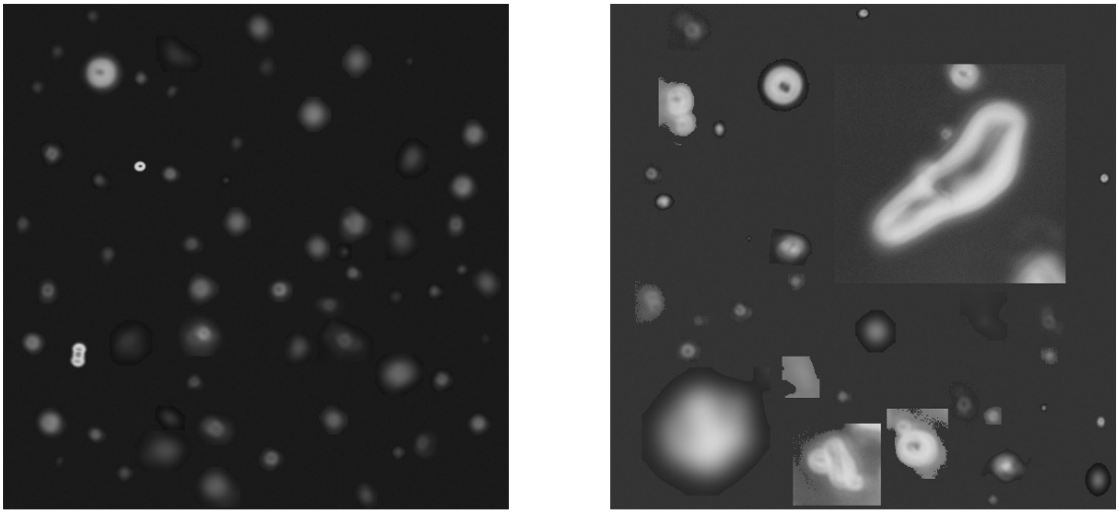

Abbildung 3.4: The difference in synthetic image generation while creating regular dataset(on the left) and complex dataset (on the right).

3.1.2 JSON file

The dataset has been created according to the COCO dataset format inspired by a programming tool created by Waspinator. The annotation format is given in 3.5.

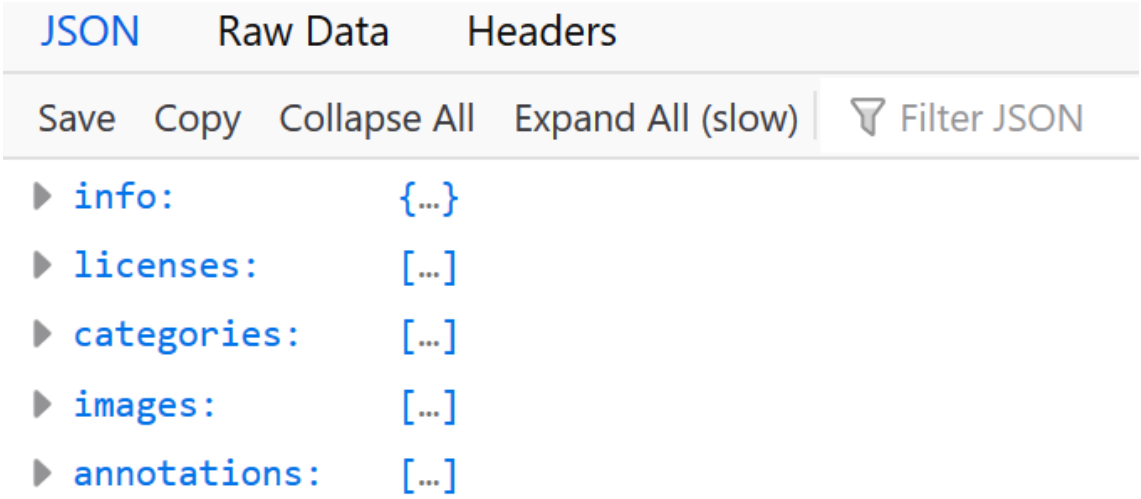

Abbildung 3.5: JSON file format similar to coco dataset

Inside the info, the general data such as the description of the dataset, year of the generation of the dataset, contributor, and also the date in which the data has been created.Under categories, each class that is drop or nodrop is assigned to a category id such as 1 or 2. The supercategory remains the same as “Spray” because the classes belong to the same category. Under images,

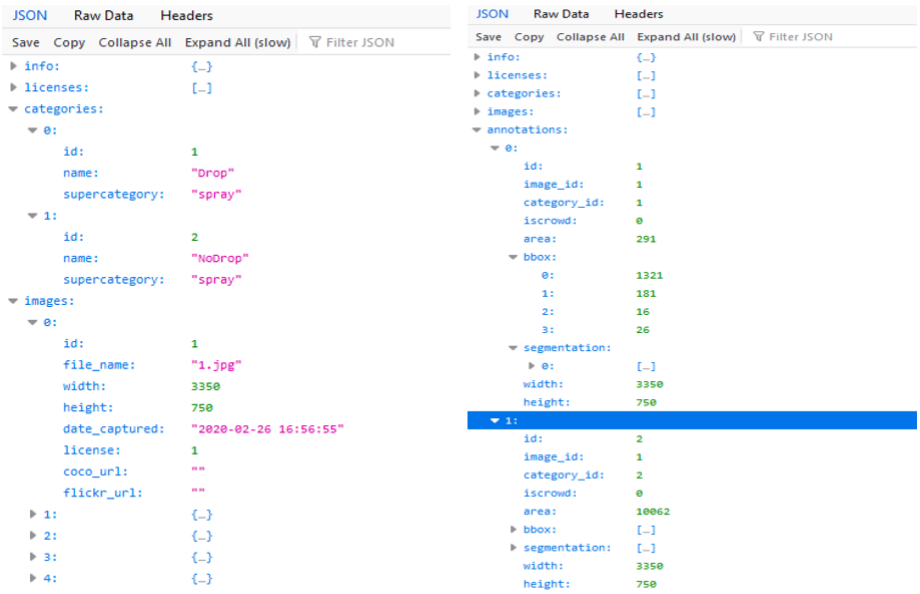

Abbildung 3.6: JSON file format: further details

an image id is given to each synthetic image that is generated using python programming. Image information such as width and height of the images are given. 3.6 consists of annotation details such as the annotation id or id which represents a single object inside an image.A single image consists of more number of objects which can come under category id 1 or 2. Depending upon the annotation for a single object or crowded object(overlapping), the option “is crowded ” is set to 0 or 1. The information about the object bounding box is saved under”bbox”.The binary mask annotation which is the coordinates of the contour has been stored in the segmentation. The binary mask format has been generated according to the COCO dataset style format. While generating synthetic images, the JSON file is written parallelly. while creating each synthetic images the program runs in a loop where the where all the information are stored and dumbed in JSON file [WASP18].

3.2 Hyperparameter tuning

Hyperparameters [SMIT18] are the key factors to train a neural network more efficiently. While training the model, the validation or test loss shows how efficiently the model has been trained. By plotting the graph between training loss and validation loss will show whether the model has been overfitted or underfitted. 3.7 shows a representation of a graph plotted with prediction error on y-axis and model complexity on the x-axis. There is a trade-off between underfitting and overfitting

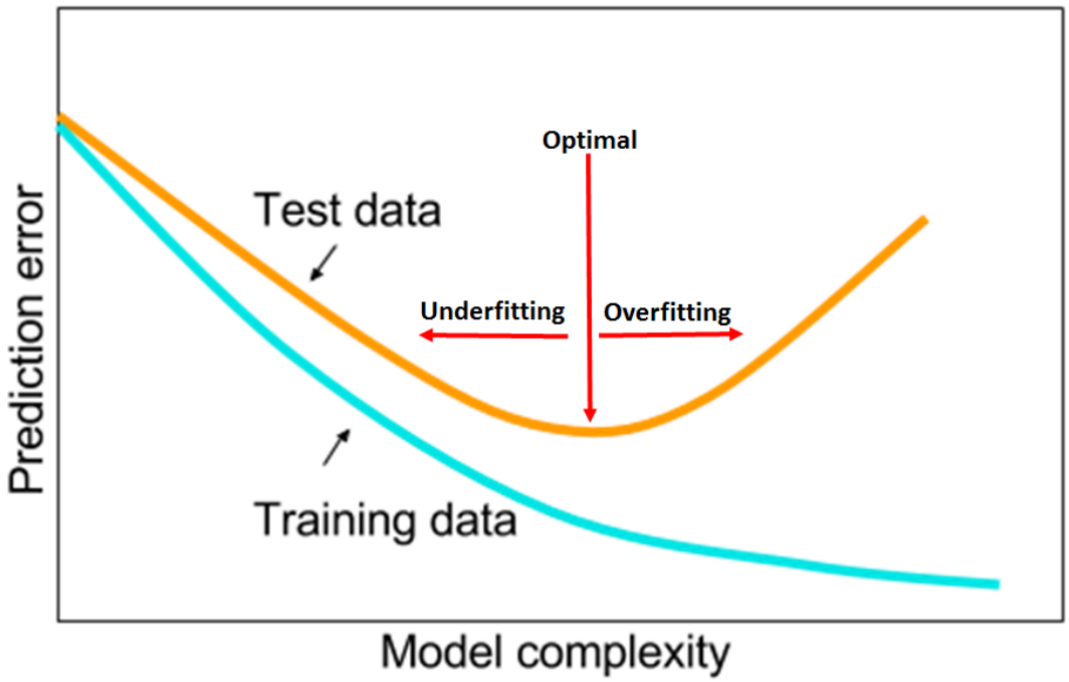

Abbildung 3.7: Tradeoff between overfitting and under fitting[SMIT18]

where an optimal point is laying between these two. As the test data line passes horizontally near

to the training data, the model is said to be trained effectively for generalizing the prediction on new data. Therefore by referencing the test data graph, hyperparameters can be tuned to achieve good training. The main parameters which are tuned accordingly are:

- **Learning rate**
- **Batch size**
- **Momentum**
- **Weight decay**

Learning rate plays a crucial role while training the model. Overfitting might occur if the learning rate is too small. If the value is too large, it will lead to superconvergence. Superconvergence is a situation where the model has been trained with less number of iterations. The training can be done using a single value for the learning rate or a range of value for the learning rate throughout the training. According to the [Smit18], the author proposes to train the model with a range of learning rate value which starts from a low value at the beginning of the training and gradually increases the value during the training process. Batch size variations can also lead to underfitting and overfitting. Using smaller batch size can provide regularization effect. The regularization effect tunes the learning algorithm in such a way that the model can generalize better. The batch size has to be selected in such a way that while training the model, the system should not run out of memory. By using a higher value of momentum, the network training can be speeded up. Learning rates and momentum has a similar effect while updating the weights during training. As per the author, the momentum can also be changed during training. While training the model, the momentum can be set up to a value in between 0.95-0.9 and decrease the value to 0.85 and later to 0.8. During the cyclic update of learning rate and momentum, the relation should be in such a way that learning rate decreases as the momentum increases. Weight decay also performs regularization effect. As per the author, cyclic change in the value of weight decay is not effective during training. Therefore the value has to be set constant. When the dataset is smaller and also the architecture of the model is not large, larger values of weight decay are suitable for training. For larger dataset and architecture, a smaller value for weight decay is effective during training. In this project, the training will be done with cyclic learning rate and momentum and also for single value for learning rate and momentum. Weight decay will be chosen between 0.001-0.00001.

3.3 Training the model

Matterport Inc.'s Mask R-CNN version which is written in python and deep learning libraries such as TensorFlow with Keras backend with further modification is used in this project [Abdu17]. The convolution backbone of the Mask R-CNN is the Resnet-101-FPN backbone which is used to extract the features from the image. For classification and regression, the network head is used(Bounding-

box recognition) and predicted mask output is applied over each ROI separately [He17]. The model was trained by using pre-initialized weights obtained by training the model on the MS COCO dataset [Lin14]. This helps to train the model faster with the help of a technique called transfer learning where the pre-trained weights on the MS COCO dataset are customized using transfer learning for our application [Weis16]. The training can be done in 3 stages such as training the network heads or training the upper layer such as from stage 4 and up in the ResNet model or training all the layers which can be considered as an end to end training [He17]. During training, image-centric training is used. There are 3 different datasets used to train the model.

•**Dataset 1:** which contains normal and distorted droplets. Image resolution is 512 x 512.

•**Dataset 2:** which contains normal and distorted droplets. Image resolution is 1024 x 1024.

•**Dataset 3:** which contains complex droplets figures. Image resolution is 512 x 512.

While training progresses, the images are resized into 512x512 except for dataset 2. The images are resized as square images by resizing and padding with zeros. As per the Matterport' version of Mask RCNN, it was introduced to improve the accuracy rather than memory efficiency, the number of images per GPU is assigned to 1 during training to avoid out of memory error. In our application, for making the model more memory efficient, a better mask/groundtruth generation technique has been introduced. As per the normal approach, a mask has been generated for each object in an image separately. For example, if there are 100 objects, there are 100 masks produced each mask has an actual resolution of the original image.The figure 3.8 represents a sample input image having four objects. The figure 3.9 shows the corresponding mask generated for one

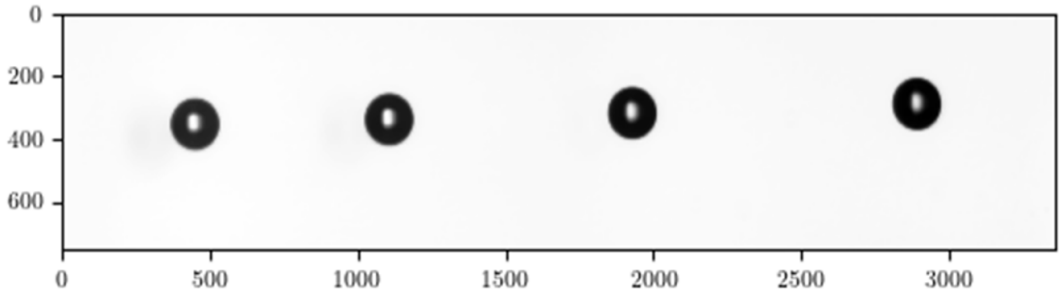

Abbildung 3.8: A synthetic image which includes 4 objects.

object among 4 objects. Creating masks separately for each object takes more memory. To avoid this problem and to make the training faster and memory-efficient, only one groundtruth mask is generated for a single input image irrespective of the number of objects. Inside a single mask image, instead of one object, all the objects are placed accordingly with specific pixel values for each class. Pixel value for the background can be" 0", for droplets its "1" and nodroplets will have the value "2". Therefore one single mask is generated with all the ground-truth information for training the model. The model is trained by using a stochastic gradient (SGD). The training has be-

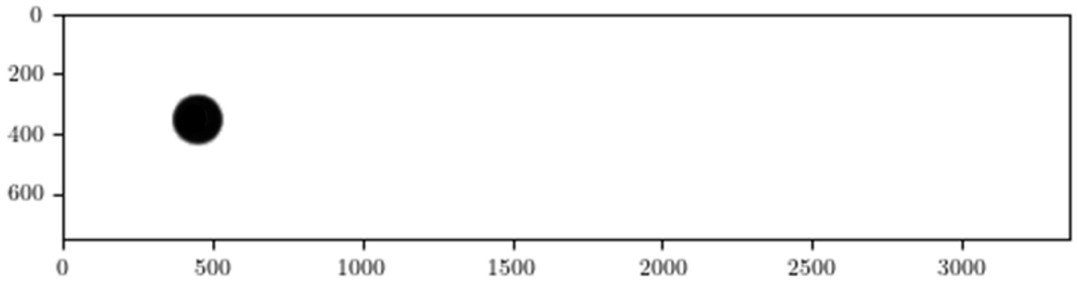

Abbildung 3.9: A mask image having only one object.

en done as an end to end training. The default learning rate and weight decay are 0.01 and 0.001 respectively. While training the model, the learning rate and weight decay have been set to 0.0001 and 0.00001. These hyperparameters are chosen after multiple trails of training approaches. The transfer learning technique has also been used by calling the previously trained model weights for the next training. RPN anchor sizes are reduced as the size of the droplets is quite small. To improve memory efficiency and to fasten the training process, the number of channels in the input image has been changed to a single-channel image. While during training, no augmentation techniques are used while most of the major changes have been made during the generation of the synthetic image dataset. The training was done on Nvidia RTX 2070 with a VRAM of 8 gigabytes and 16 gigabytes of Ram.

Dataset	Resize resolution	No of epochs	Learning rate	Momentum	Weight decay
Dataset 1	512x512	150	0.0001	0.9	0.00001
Dataset 2	1024x1024	150	0.0001	0.9	0.00001
Dataset 3	512x512	111	0.0001	0.9	0.00001
Dataset 3 (cyclic)	512x512	50 50 50	0.0001 0.001 0.005	0.95 0.88 0.82	0.0001

Tabelle 3.1: Tabular description regarding training procedure.

Dataset 1 and dataset 2 are trained for 150 epochs and the learning rate is 0.0001 and the momentum is 0.9 and it remains constant throughout the training. The training was initialized by using pre-trained model weights that were trained on the COCO dataset. Dataset 3 is used to train the model twice. In the first training approach with dataset 3, the learning rate and momentum remain the same till the end of the training. The learning rate is 0.0001 and the momentum is 0.9. Weight decay is set to be smaller and has a value of 0.00001. The model has been trained for 111 epochs. In the second training process, the dataset has been trained with a cyclic learning rate and momentum. There are a total of 150 epochs where in the first 50 epochs, the momentum is set to be 0.95 and the learning rate will be 0.0001. The second 50 epochs are trained by increasing the value of the learning rate to 0.001 and the momentum is decreased to 0.88. Last 50 epochs

are trained by changing the learning rate to 0.005 and momentum of 0.82. In this training process, weight decay remains the same and has a value of 0.0001.

3.4 Evaluation procedure

Evaluation metric that is used in this project is Intersection-Over-Union metric. This is a common evaluation metric that is used for semantic segmentation of the images. Tensorflow provides mean IoU function where te predictions are saved inside a confusion matrix. From this confusion matrix, IoU for individual classes as well as mean IoU for the entire tested samples is calculated. IoU is calculated by :

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.1}$$

For the evaluation of the model, A ground truth image and a predicted image are needed. By using the information from the JSON file, the mask images are created as ground-truth for each image in the dataset. By using the trained model, the prediction has been made for respective images in the dataset and from these predictions, another mask image is generated. These two images are fed along with the number of classes to the mean IoU function. The prediction can be done in two different ways.

- The image is fed as a whole for prediction.
- The image is tiled and stitched back after prediction.

Feeding image in its whole resolution is a common practice followed for prediction. But if the image resolution is higher, It takes a lot of time for prediction because of the computational overhead. For avoiding this situation, the images are tiled and fed one by one for prediction and rejoined after completing the prediction of all the tiles.

3.4.1 Image tiling

Larger images which are having dimensions greater than the expected input image size for the trained model is considered for tiling. The number of tiles is determined by the relation between the expected input size and the actual size of the image. There will be a predefined pixel/region overlap for avoiding the cutting edge while stitching back the images together. The overlapping value will be equal to the largest size of the droplet which can be obtained from the raw images in the laboratory. The bounding box of the actual image is saved as the boundary line. While tiling the images, the tiling coordinates are saved corresponding to the actual position from the real images. The figures 3.10 and 3.11 shows an example for tiling and stitching of the input image. For the evaluation of the model, Image tiling is not required while the synthetic datasets are generated

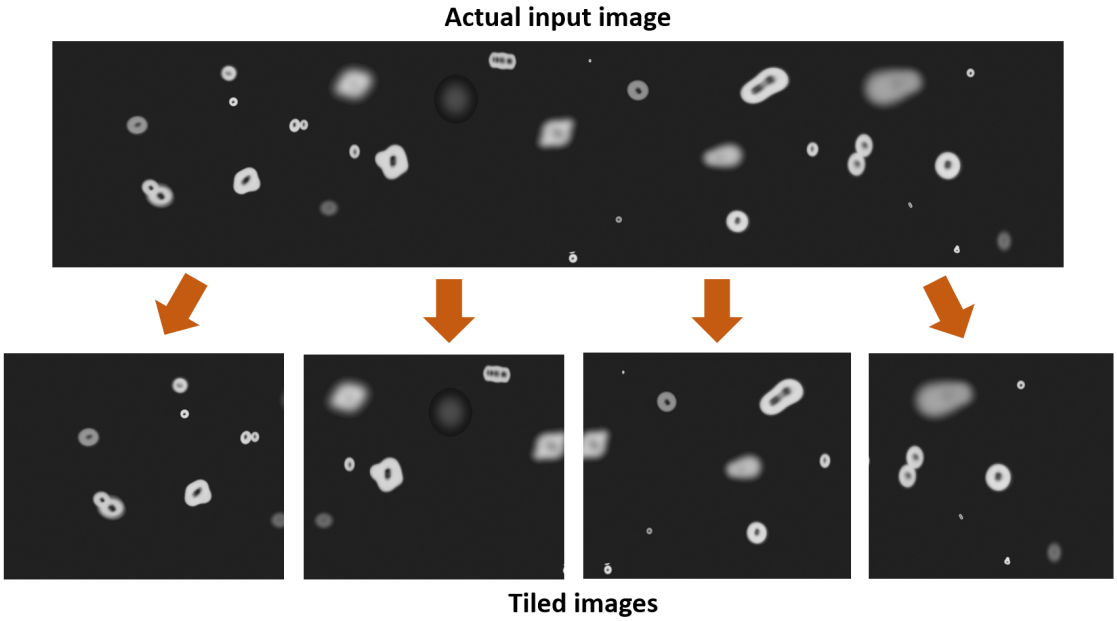

Abbildung 3.10: Visual representation of tiling an input image.

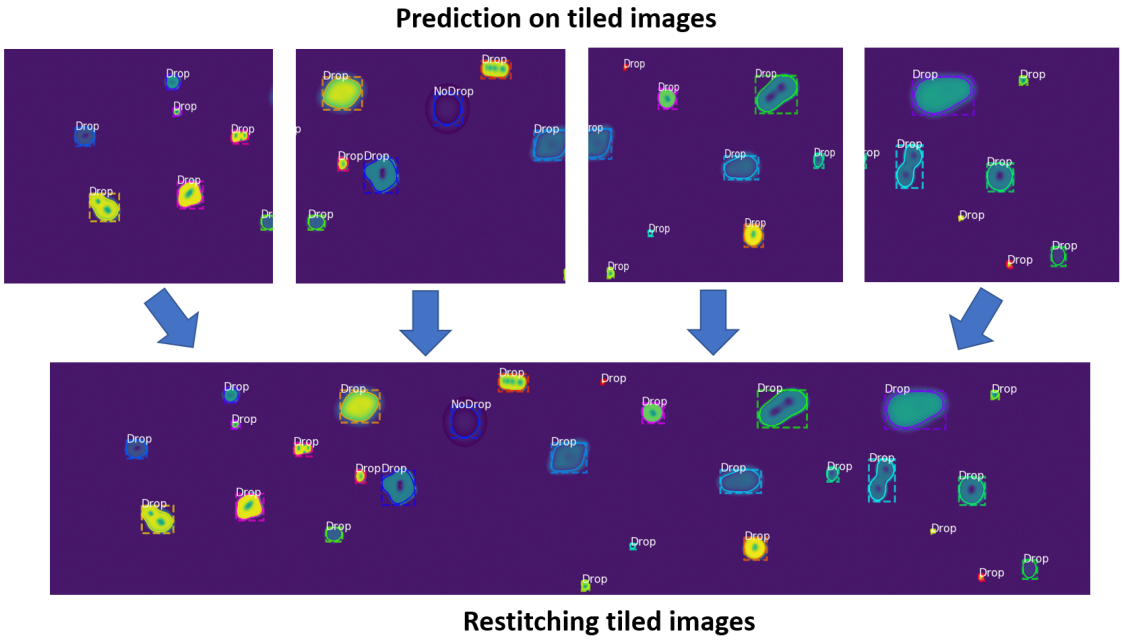

Abbildung 3.11: Stitching the tiles back to its original resolution.

with a resolution of 512 x 512 and 1024 x 1024 respectively. In this project, for analysing and retrieving statistical data from each raw images, Image tiling is used.

During prediction, there is a possibility that the same objects can be present in two different tiles. In this situation, during stitching back the image tiles, a comparative evaluation has been made concerning objects in the boundary line and also duplicated objects are eliminated through verification procedures. Hyperparameters such as pixel overlap and tiling dimensions are crucial for evaluation.

4 Data evaluation, analysis, and inference

In this chapter, the epoch loss graphs that are obtained from Tensorboard for each training conditions is plotted and compared. All the trained models are evaluated on the corresponding test data set. The evaluation of the model is based on the evaluation metric mean iou. With the help of Keras backend TensorFlow, mean iou is calculated and the results are compared. The trained models are also used to visualize the prediction (semantic segmentation) of a random image from the dataset and also from the raw image dataset that acquired from the lab. Finally, using these trained weights, analysis of the real dataset is carried out. The physical parameters such as area, contour, perimeter, solidity, centroid are measured and saved in JSON file. These parameters are plotted into graphs for further visual analysis and understand the commonality of the physical and chemical state of the fuel droplets in the real dataset.

4.1 Visualization using Tensorboard

Tensorboards helps to monitor as the training progresses. From the tensorboard graphs, both training and validation losses are mapped on the y-axis and the number of epochs is mapped on the x-axis. By interpreting these graphs, we can understand whether the training leads to underfitting or overfitting. In these graphs, the orange curve illustrates the training loss and the blue curve represents validation loss. Both graphs are smoothened to the factor of 0.6. The actual representation of these learning curves can be seen as a shaded curve on the graph. For saving the best model, tensorboard provides a callback function which monitors the validation loss or validation accuracy during the training. If the validation loss is monitored, the mode should set to “min” and for validation accuracy, the mode has to be set to ”max”. In this project, four training sections have been carried out as explained in the section 3.3.

Graph 4.1 represents the training with dataset 1. As the training progresses, the training loss is consistently decreasing while the validation loss has some fluctuations where some points, the loss is higher. This might be due to the less number of validation samples used during the validation of the model. To evaluate the generalizing ability of the model, more validation samples has to be provided. Therefore the validation loss decreases inconsistently and the validation loss can go under the training loss at some points. This shows that the validation dataset might be easier to predict by the model compare to the training dataset. In this training process, the best model was

saved at epoch 143.

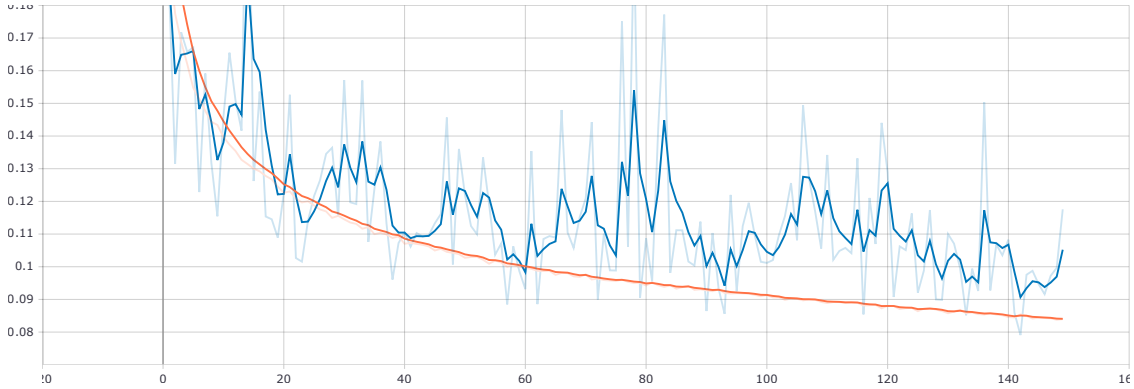

Abbildung 4.1: Train and validation learning curves for dataset 1.

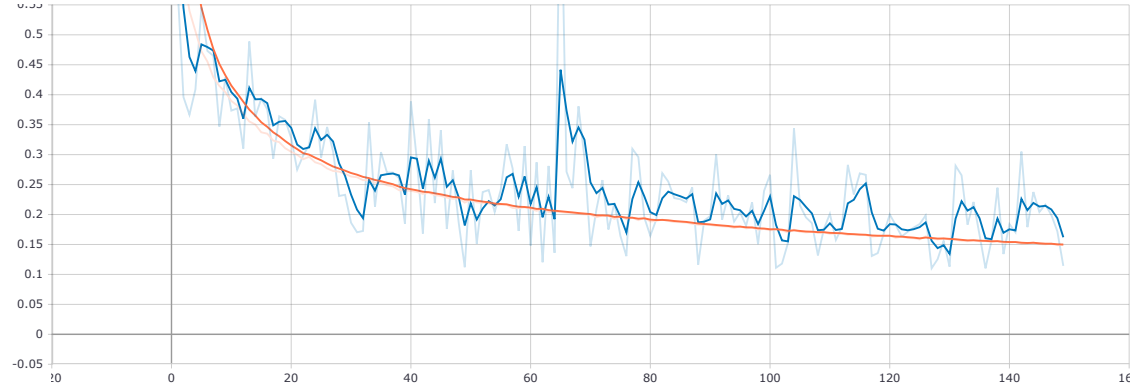

Abbildung 4.2: Train and validation learning curves for dataset 2.

The second graph 4.2 illustrates the training with dataset 2. In this training process, the training loss is decreasing gradually. But the validation loss has some noise similar to graph figure 1. But the noisy movement of the validation loss is less compared to the graph figure 1. This might be because of the more number of objects present in a single image in dataset 2. Even though the validation steps assigned during training is the same in both scenarios, an increasing number of objects have a similar impact on more validation samples. This makes the validation loss less noisy. In this training process, the best model was saved at epoch 137.

The third graph 4.3 describes the training with dataset 3. These datasets consist of more complex droplet figures. Even though the image resolution of dataset 1 and dataset 3 are the same, the validation learning curve is different because of the complexity of the dataset 3. The more complex validation data have more information for evaluation compared to dataset 1. In this training process, the best model was saved at epoch 58.

The fourth graph 4.4 represents the cyclic training procedure by varying the hyperparameters. The training has been done using dataset 3. In this training, the hyperparameters are changed every 50th epoch. Therefore a peak in the graph can be seen at 50th and 100th epoch. During the first 50 epochs, the training loss and validation loss is decreasing gradually. But there is some noise in the

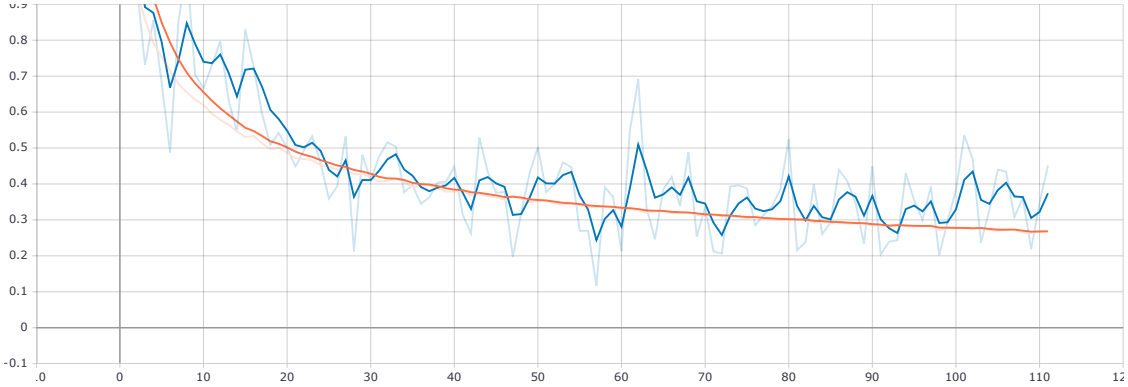

Abbildung 4.3: Train and validation learning curves for dataset 3.

validation loss. After 50th epoch, the validation loss has more noise and loss started to increase while the training loss was decreasing. This phenomenon is called overfitting. This might be due to fewer samples in the validation loss or the hyperparameter has to be tuned in a better way to reduce the validation loss. In this training two best models are being saved. The first model is at epoch 46 and the second model is at epoch 126.

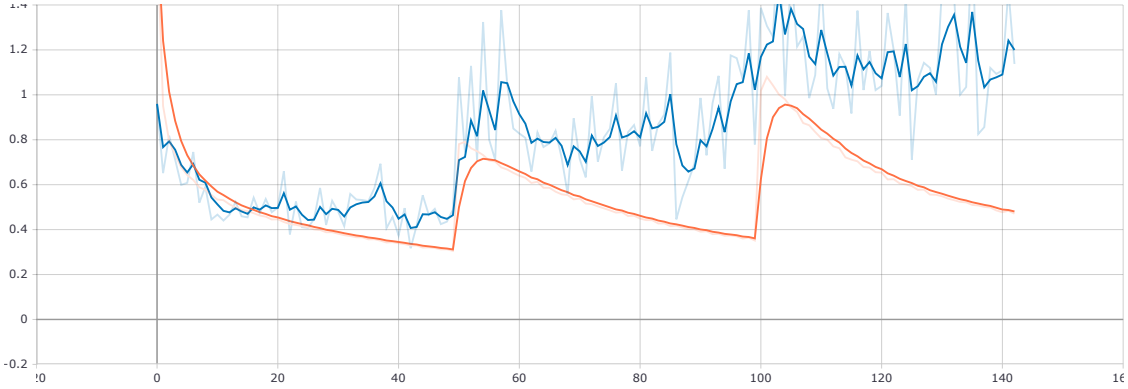

Abbildung 4.4: Train and validation learning curves for dataset 3 with cyclic hyperparameter tuning.

For further evaluation, the best models that were saved during training are used. These models are used to evaluate on both normal and complex test dataset.

4.2 Evaluation results

The results that are obtained by evaluating each model are formulated in the form of a table as given below. The evaluation is performed on 400 test samples from corresponding datasets where the model has been trained.

The table illustrates some of the interesting results from each trained model. It helps to compare the results and choose some model for further statistical analysis. With the help of table 3.3 and 4.2, the results and the coresponding hyper parmeter that were used during training can be

compared to choose best hyper parameters for future training of the model.

The background IoU value is almost similar in all cases. In the first case, the drop IoU value and nodrop IoU value is less than 0.8 whereas in case 2, the model achieved good result where the drop IoU value and nodrop IoU is around 0.9. the mean IoU value obtained in case 2 is the highest among all other cases. The value obtained in case 2 is 0.92. On the other hand, case 3 results are not up to the mark while the mean IoU value achieved during the evaluation was 0.65. However, the case 4 has shown better results compared to case 1 and case 3. In case 4 the model is trained cyclically as explained before. The mean IoU achieved in with two different models are 0.84 and 0.86 respectively.

Dataset	Best epoch	Background IoU	Drop IoU	nodrop IoU	Mean IoU
Dataset 1 (Case 1)	143	0.993150	0.794735	0.724253	0.836933
Dataset 2 (Case 2)	137	0.998257	0.905204	0.884862	0.929441
Dataset 3 (Case 3)	58	0.97990	0.383152	0.58382	0.648961
Dataset 3 (Case 4)	43 126	0.987421 0.991189	0.722528 0.728870	0.829574 0.880576	0.846508 0.866878

Tabelle 4.1: Evaluation results of each trained model.

To understand how these models behave in complex scenarios, the cross-evaluation technique can be used. For cross-evaluation, the trained model in case 2 can be used on test samples in case 4 and vice versa. In case 4, the trained weight at epoch 126 is used for evaluation. The table 4.2 shows the result of cross-evaluation.

Dataset used	Trained weights used	Background IoU	Drop IoU	nodrop IoU	Mean IoU
Dataset 1	Case 4 (epoch 126)	0.993614	0.514264	0.570983	0.692954
Dataset 2	Case 1 (epoch 137)	0.962267	0.171429	0.279537	0.471078

Tabelle 4.2: Cross evaluation result .

However, for a better understanding of the model results, the semantic segmentation results of each model are shown in 4.2.1.

4.2.1 Visualisation of the test dataset

In this section, the predictions obtained from each trained model for a few sample images are illustrated in the form of figures. A sample input image is taken from the test dataset and its ground-truth image is generated as a ternary image. Similarly, the corresponding prediction is also converted into a ternary image for the easiness of understanding. The yellow regions represent

"nodrop" regions whereas the blue regions represent "drop" regions. Finally, the prediction is visualised along with the input image where each object in the image are semantically segmented with a bounding box and corresponding predicted label.

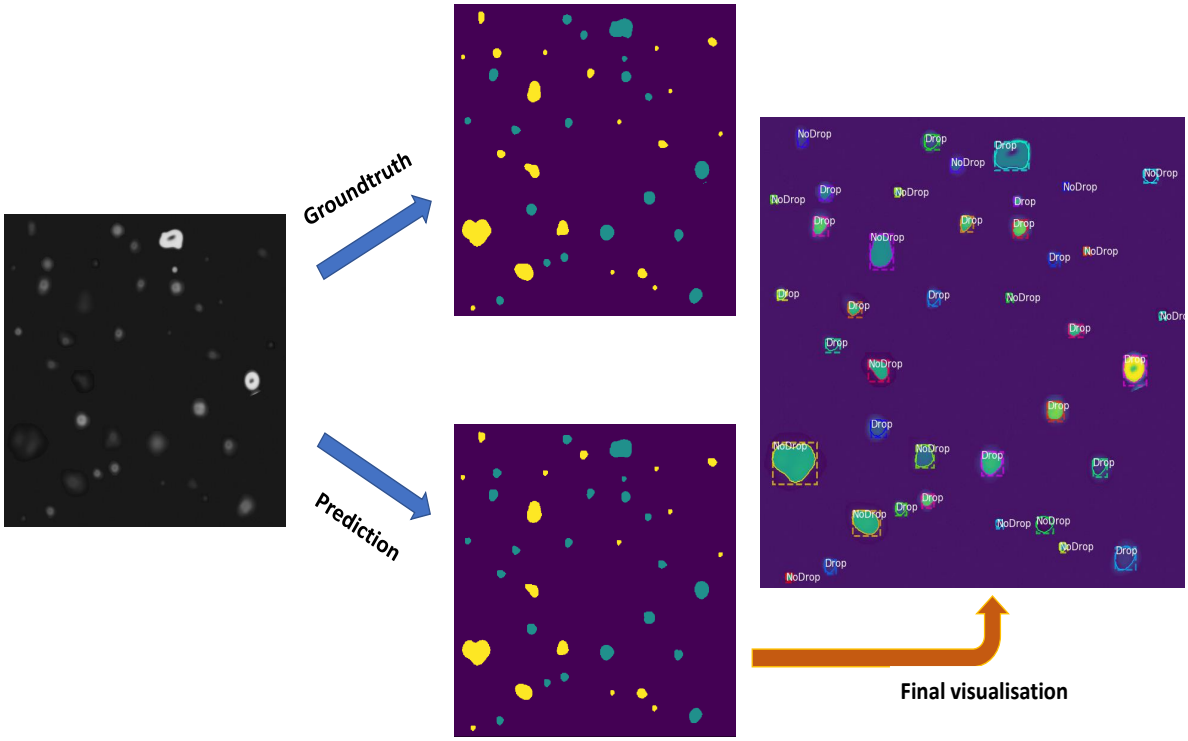

Abbildung 4.5: Case 1: A sample test image with the prediction result (a)

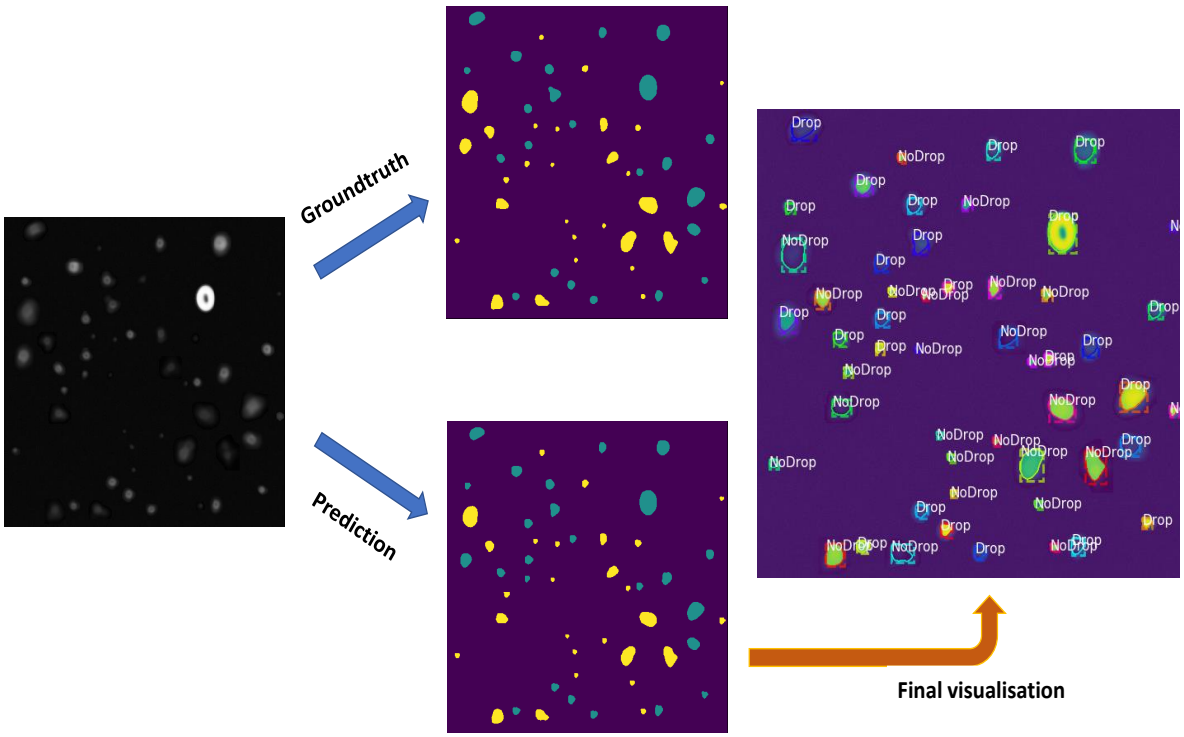

Abbildung 4.6: Case 1: A sample test image with the prediction result (b)

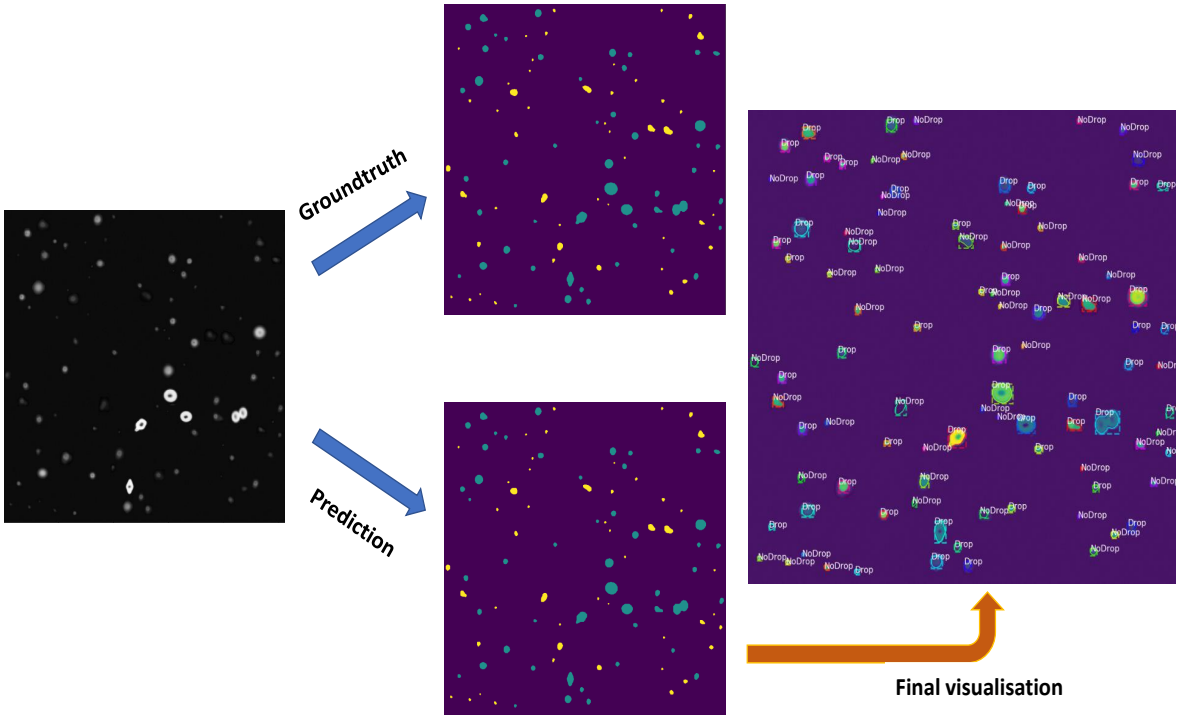

Abbildung 4.7: Case 2: A sample test image with the prediction result (a)

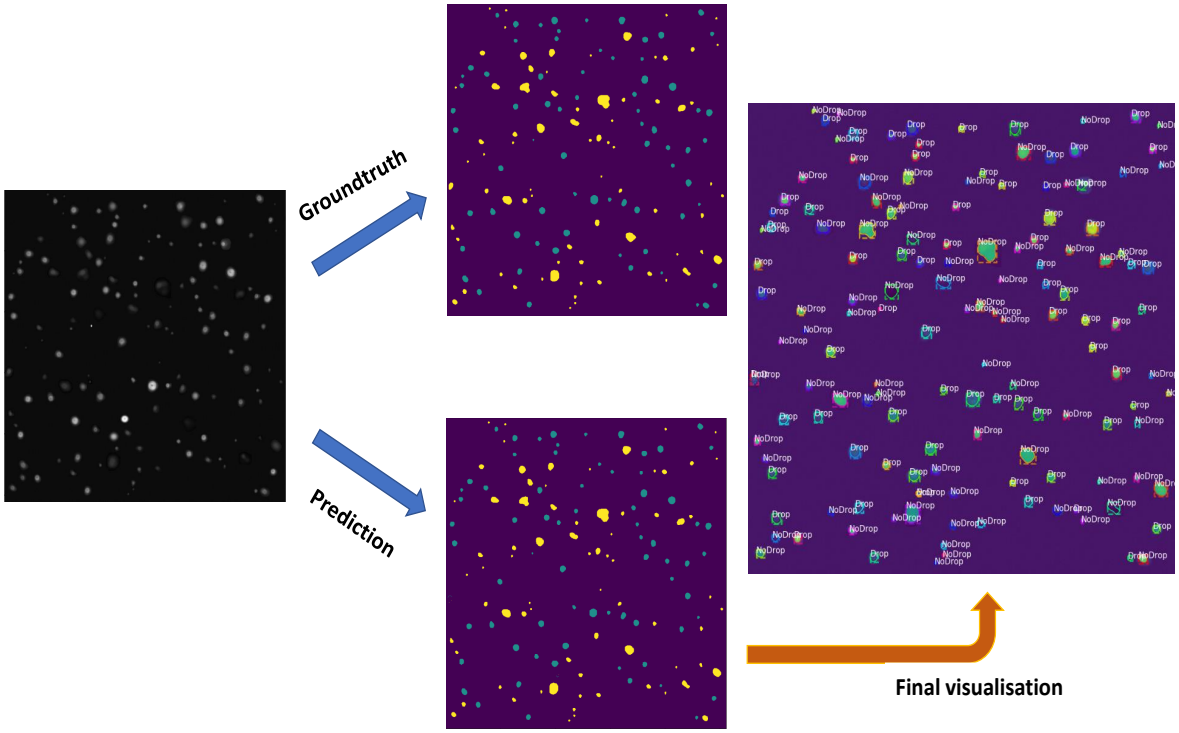

Abbildung 4.8: Case 2: A sample test image with the prediction result (b)

Case 2 shows better evaluation results compared to all other datasets. One of the possibilities for good evaluation is that the dataset is bigger compared to case 1 and contains more number of objects.

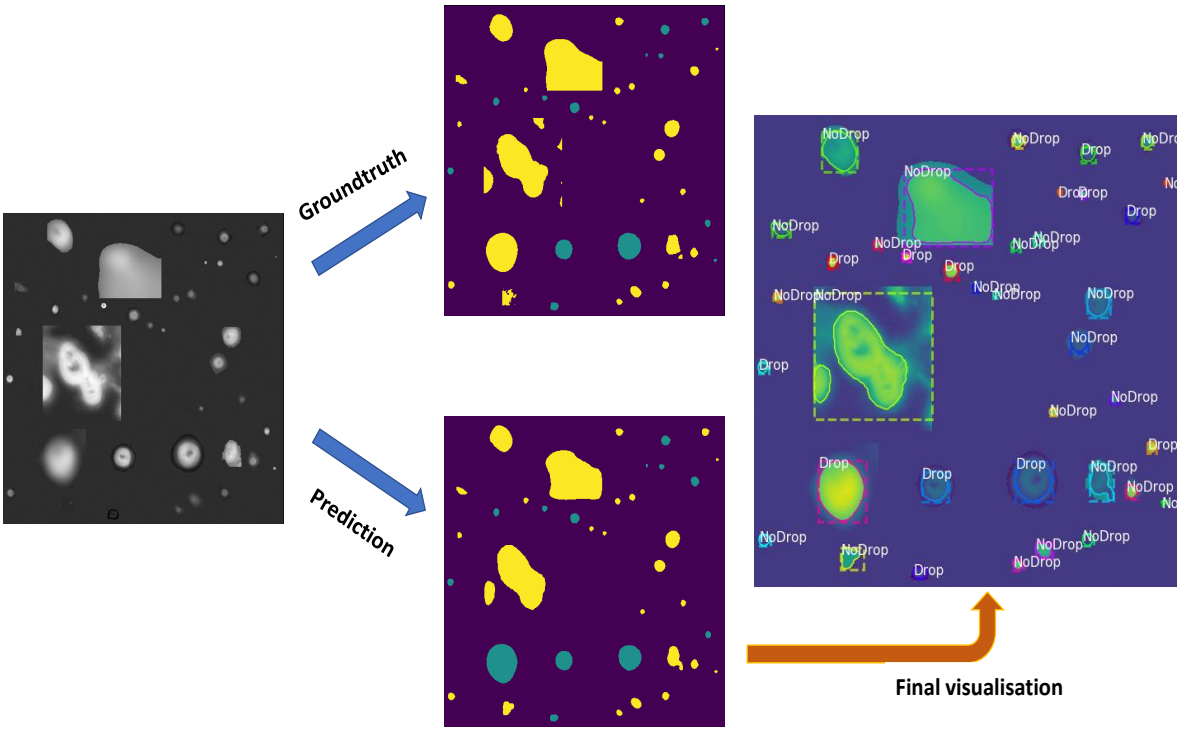

Abbildung 4.9: Case 3: A sample test image with the prediction result (a)

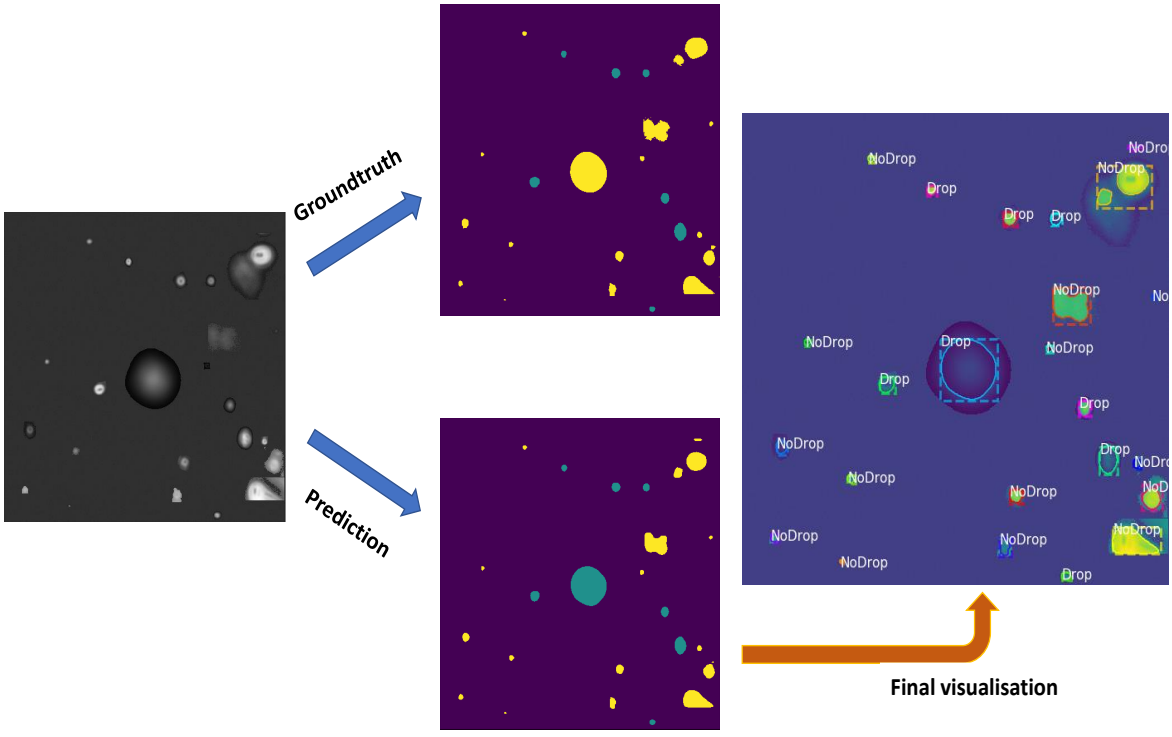

Abbildung 4.10: Case 3: A sample test image with the prediction result (b)

In case 4, there are 2 models available. But only the last one is used for predicting the result while by comparing the evaluation metric results between these two models, the trained weight at epoch 126 shows a slightly better result than the trained weight at epoch 43.

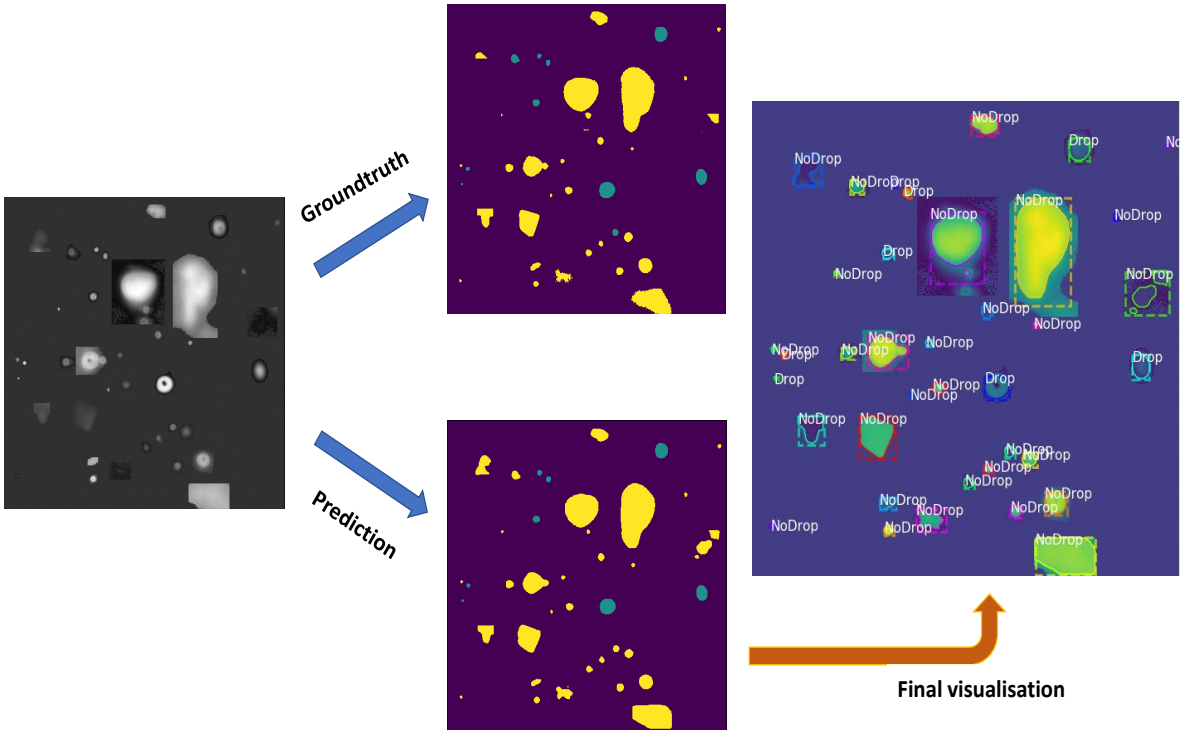

Abbildung 4.11: Case 4 (epoch:126): A sample test image with the prediction result (a)

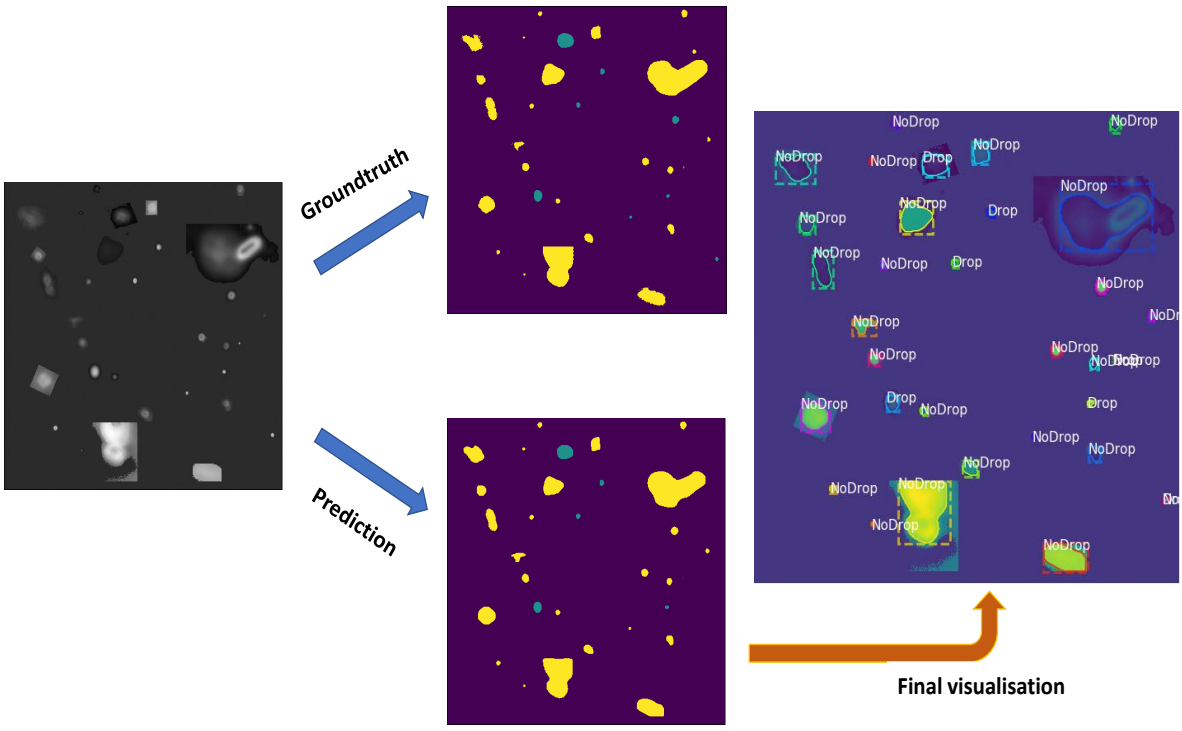

Abbildung 4.12: Case 4 (epoch:126): A sample test image with the prediction result (b)

4.2.2 Visualisation of the raw dataset

In this section, a sample image is chosen from the raw dataset and the trained models are used to predict the results. For better visual analysis and compare the robustness of the trained model, the same image is used for testing for all the trained models.

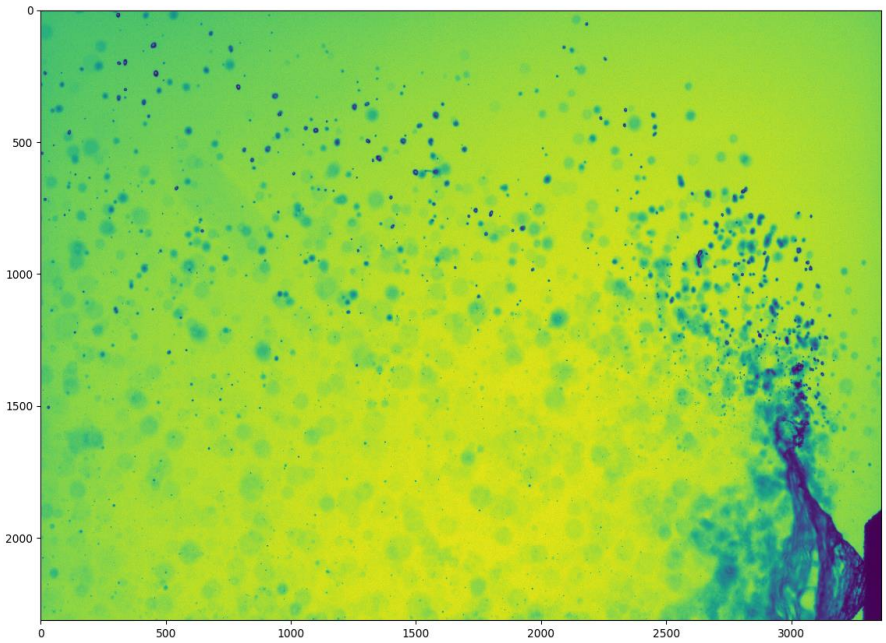

Abbildung 4.13: A sample image taken from the raw dataset for prediction.

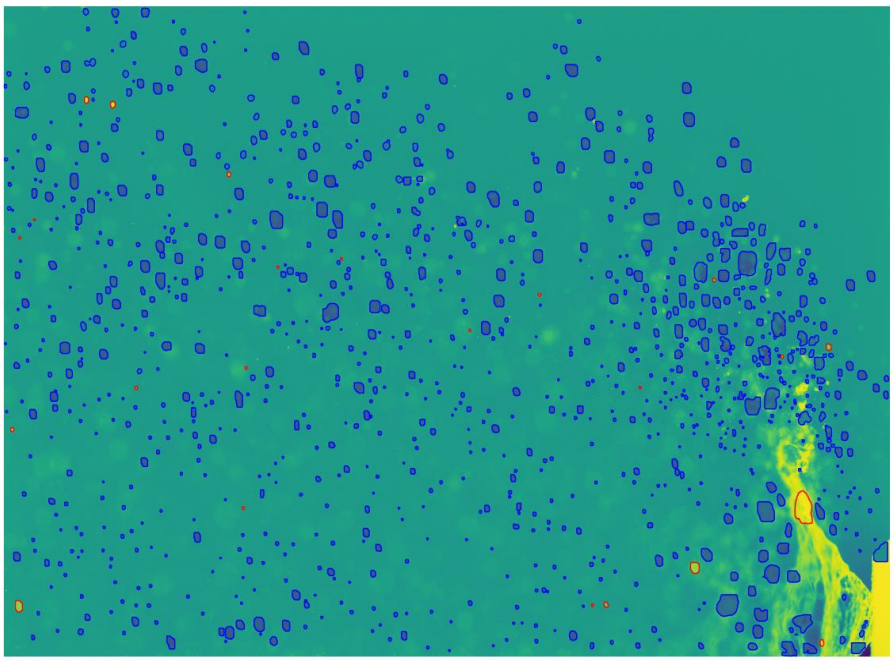

Abbildung 4.14: Prediction using the trained model from case 1.

Image 4.13 shows a sample of the droplet spraying through an injector. This image illustrates the complexity of the actual dataset. As the fuel is sprayed through an injector, the physical and che-

mical state of the fuel droplets are changed in a matter of seconds. Some of the droplets do not diffuse to the surrounding that easily. The complexity of the dataset is determined by the distribution of fuel droplets.

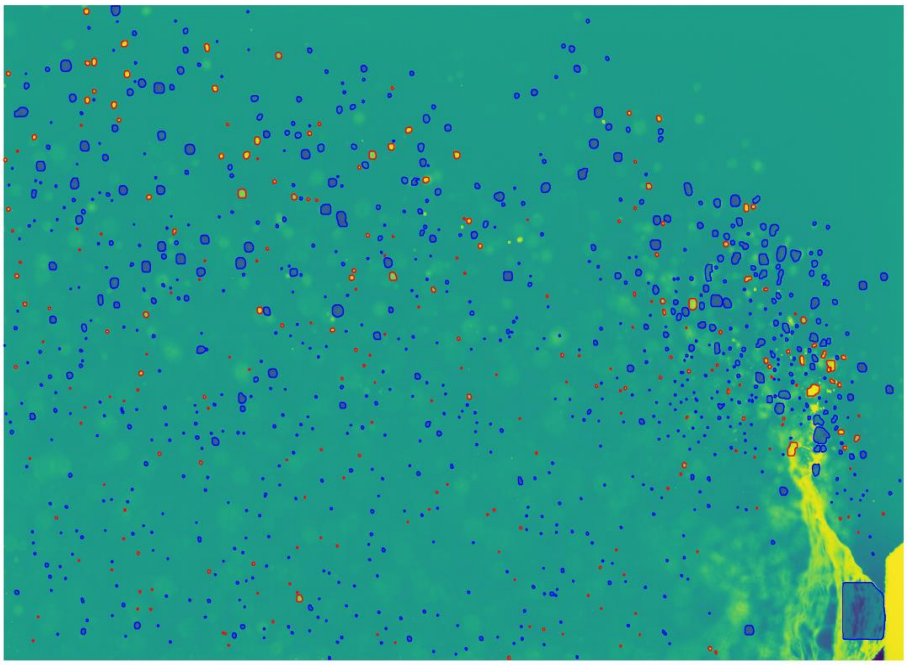

Abbildung 4.15: Prediction using the trained model from case 2.

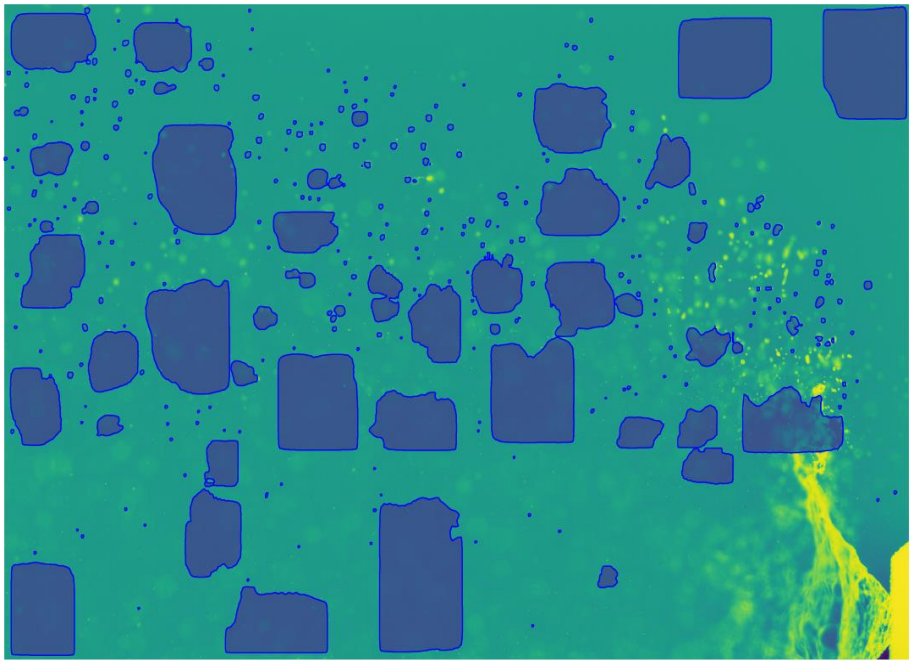

Abbildung 4.16: Prediction using the trained model from case 3.

Image 4.14 shows the predicted result obtained from the trained model of case 1. The model detects more number of nodrop regions which is represented as blue regions with blue outline compared to the drop regions which is represented as a yellow region with a red outline.

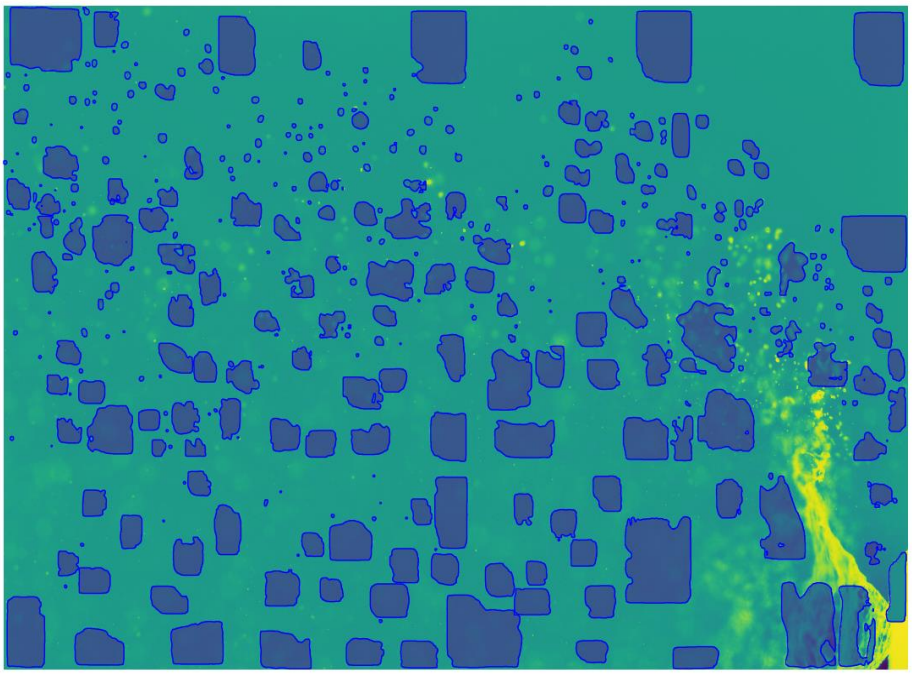

Abbildung 4.17: Prediction using the trained model from case 4 (epoch:43)

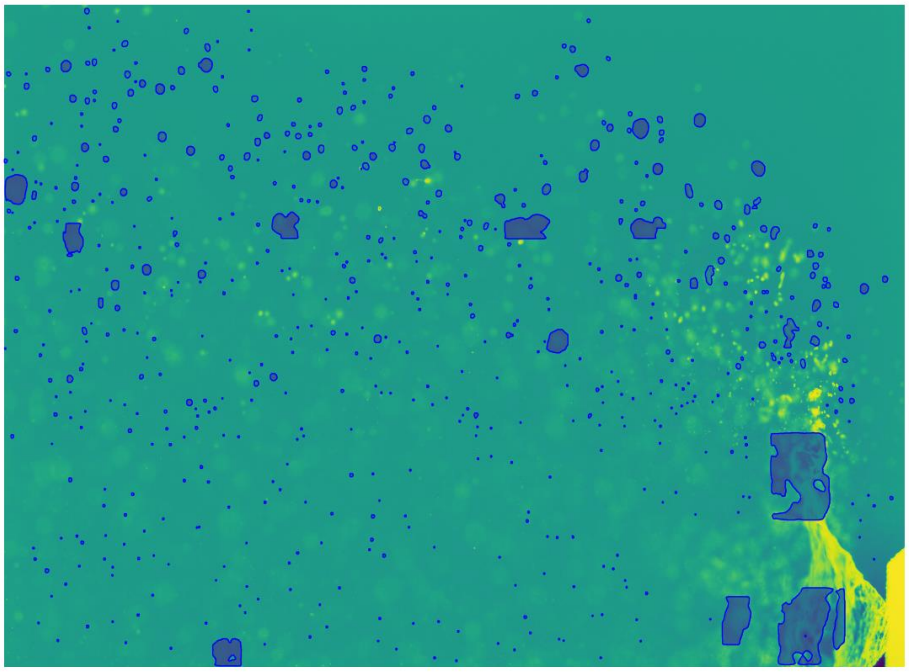

Abbildung 4.18: Prediction using the trained model from case 4 (epoch:126)

However, Image 4.15 shows much more good results compared to Image 1. The trained model from case 2 had detected more drop regions. Image 4.16 and 4.17 shows the detection of nodrop region as blocks and failed to detect drop regions. Image 4.18 shows much better predictions of nodrop region in the form of small blue circles but failed to detect drop regions compare to image 4.15.

4.3 Analysis of the real data

5 Discussion and future work

This chapter summarises the implementation of DNN and analysis of the results of the previous section. The discussion leads to new ideas for future progress in the field of research and development of efficient combustion systems or other applications which involves fuel injectors.

5.0.1 Summary

5.0.2 Conclusion and inference

5.0.3 Future work

As the data acquisition is from the industrial cameras that are unstructured and complex, the preprocessing of these data is tedious while data preprocessing takes the 1/3 rd of the actual time provided for most of these projects. Semi automatization of synthetic data generation which resembles the actual data can make remarkable progress in terms of time and money. By creating more complex images for expanding the database and make it more robust will provide a significant change for training a deep neural network and improve the results for the future outcome.

Literaturverzeichnis

[ABDU17] ABDULLA, Waleed. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017.

[ABDU18] ABDULLA, Waleed. *Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow*. https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c7 Dec 2018.

[AL-M19] AL-MASRI, Anas. *What Are Overfitting and Underfitting in Machine Learning?* https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690. Jun 2019.

[BANE19] BANERJEE, Ananya. *Different Computer Vision Tasks*. https://medium.com/@ananya.banerjee.rr/different-computer-vision-tasks-b3b49bbae891. Oct 2019.

[BOEH18] BOEHMKE, Bradley. *Feedforward Deep Learning Models*. http://uc-r.github.io/feedforward_DNN#:~:text=Deep%20feedforward%20neural%20network. Apr 2018.

[BROW19] BROWNLEE, Jason. *What is Deep Learning?* https://machinelearningmastery.com/what-is-deep-learning/#:~:text=WhyDeepLearning?SlidebyAndrewNg,allrightsreserved. Aug 2019.

[ESTE17] ESTEVA, Andre; KUPREL, Brett; NOVOA, Roberto A.; KO, Justin; SWETTER, Susan M.; BLAU, Helen M. ; THRUN, Sebastian: Dermatologist-level classification of skin cancer with deep neural networks. In: *nature* 542 (2017), Nr. 7639, S. 115–118.

[HANS19] HANSEN, Casper. *Optimizers Explained - Adam, Momentum and Stochastic Gradient Descent*. https://mlfromscratch.com/optimizers-explained/#/. Oct 2019.

[HE16] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing ; SUN, Jian: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 770–778.

[HE17] HE, Kaiming; GKIOXARI, Georgia; DOLLÁR, Piotr ; GIRSHICK, Ross: Mask r-cnn. In: *Procee-*

dings of the IEEE international conference on computer vision, 2017, S. 2961–2969.

[HOSA17] HOSANG, Jan; BENENSON, Rodrigo ; SCHIELE, Bernt: Learning non-maximum suppression. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 4507–4515.

[HURW18] HURWITZ, Judith; KIRSCH, Daniel: Machine learning for dummies. In: *IBM Limited Edition* 75 (2018).

[JABB15] JABBAR, H; KHAN, Rafiqul Z.: Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). In: *Computer Science, Communication and Instrumentation Devices* (2015), S. 163–172.

[JOHN19] JOHNSON, Jeremiah W.: Automatic nucleus segmentation with Mask-RCNN. In: *Science and Information Conference* Springer, 2019, S. 399–407.

[JORD18a] JORDAN, Jeremy. *Evaluating image segmentation models.* https://www.jeremyjordan.me/evaluating-image-segmentation-models/. Dec 2018.

[JORD18b] JORDAN, Jeremy. *An overview of semantic image segmentation.* https://www.jeremyjordan.me/semantic-segmentation/. May 2018.

[KAEL96] KAELBLING, Leslie P.; LITTMAN, Michael L. ; MOORE, Andrew W.: Reinforcement learning: A survey. In: *Journal of artificial intelligence research* 4 (1996), S. 237–285.

[KING14] KINGMA, Diederik P.; BA, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014).

[KRIE07] KRIESEL, David: A brief introduction on neural networks. (2007).

[KUMA19] KUMAR, Harshit. *Quick intro to Instance segmentation: Mask R-CNN*. https://kharshit.github.io/blog/2019/08/23/quick-intro-to-instance-segmentation. Aug 2019.

[LE18] LE, James. *The 5 Computer Vision Techniques That Will Change How You See The World*. https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world- Apr 2018.

[LI17] LI, Yi; QI, Haozhi; DAI, Jifeng; JI, Xiangyang ; WEI, Yichen: Fully convolutional instance-aware semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, S. 2359–2367.

[LIN14] LIN, Tsung-Yi; MAIRE, Michael; BELONGIE, Serge; HAYS, James; PERONA, Pietro; RAMANAN, Deva; DOLLÁR, Piotr ; ZITNICK, C L.: Microsoft coco: Common objects in context. In: *European conference on computer vision* Springer, 2014, S. 740–755.

[LIU17] LIU, Weibo; WANG, Zidong; LIU, Xiaohui; ZENG, Nianyin; LIU, Yurong ; ALSAADI, Fuad E.: A

survey of deep neural network architectures and their applications. In: *Neurocomputing* 234 (2017), S. 11–26.

[Liu20] Liu, Patrick L. *Single Stage Instance Segmentation-A Review*. https://towardsdatascience.com/single-stage-instance-segmentation-a-review-1eeb66e0cc49. Jul 2020.

[Mins90] Minsky, Marvin: The age of intelligent machines: thoughts about artificial intelligence. In: *KurzweilAI. net (en línea) http://www. kurzweilai. net/meme/frame. html* (1990).

[Niga18] Nigam, Vibhor. *Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning*. https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-c Sep 2018.

[Nils96] Nilsson, Nils J.: Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach. In: *Artificial intelligence* 82 (1996), Nr. 1-2, S. 369–380.

[Pere18] Perez, Javier A.; Deligianni, Fani; Ravi, Daniele ; Yang, Guang-Zhong: Artificial intelligence and robotics. In: *arXiv preprint arXiv:1803.10813* (2018).

[Pinh15] Pinheiro, Pedro O.; Collobert, Ronan ; Dollár, Piotr: Learning to segment object candidates. In: *Advances in Neural Information Processing Systems*, 2015, S. 1990–1998.

[Prat16] Pratt, Harry; Coenen, Frans; Broadbent, Deborah M.; Harding, Simon P. ; Zheng, Yalin: Convolutional neural networks for diabetic retinopathy. In: *Procedia Computer Science* 90 (2016), S. 200–205.

[Ren15] Ren, Shaoqing; He, Kaiming; Girshick, Ross ; Sun, Jian: Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*, 2015, S. 91–99.

[Rizw18] Rizwan, Muhammad. *How to Select Activation Function for Deep Neural Network*. https://engmrk.com/activation-function-for-dnn/. May 2018.

[Rude16] Ruder, Sebastian: An overview of gradient descent optimization algorithms. In: *arXiv preprint arXiv:1609.04747* (2016).

[Saha18] Saha, Sumit. *A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2 Dec 2018.

[Shab19] Shabbeer Basha, SH; Dubey, Shiv R.; Pulabaigari, Viswanath ; Mukherjee, Snehasis: Impact of Fully Connected Layers on Performance of Convolutional Neural Networks for

Image Classification. In: *arXiv* (2019), S. arXiv–1902.

[SHAR17] SHARMA, Aditya. *Understanding Activation Functions in Deep Learning*. https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/. Oct 2017.

[SIMO14] SIMONYAN, Karen; ZISSERMAN, Andrew: Very deep convolutional networks for large-scale image recognition. In: *arXiv preprint arXiv:1409.1556* (2014).

[SING14] SINGH, Sapna; SINGH, Daya S. ; KUMAR, Shobhit: Modified Mean Square Error Algorithm with Reduced Cost of Training and Simulation Time for Character Recognition in Back-propagation Neural Network. In: *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013* Springer, 2014, S. 137–145.

[SMIT18] SMITH, Leslie N.: A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. In: *arXiv preprint arXiv:1803.09820* (2018).

[WASP18] WASPINATOR. *Create your own COCO-style datasets*. https://patrickwasp.com/. Jun 2018.

[WATA19] WATANABE, Thomio; WOLF, Denis F.: Instance Segmentation as Image Segmentation Annotation. In: *2019 IEEE Intelligent Vehicles Symposium (IV)* IEEE, 2019, S. 432–437.

[WEHL17] WEHLE, Hans-Dieter: Machine learning, deep learning, and ai: What's the difference? In: *International Conference on Data scientist innovation day, Bruxelles, Belgium*, 2017.

[WEIS16] WEISS, Karl; KHOSHGOFTAAR, Taghi M. ; WANG, DingDing: A survey of transfer learning. In: *Journal of Big data* 3 (2016), Nr. 1, S. 9.

[XI19] XI, Yan; KUN, Lu ; KAI, Wang: Summary of Research and Development of Intelligent Combustion Optimization System. In: *IOP Conference Series: Earth and Environmental Science* Bd. 267 IOP Publishing, 2019, S. 062040.