Measurement of Droplets in Vaporized Fluids using Machine Learning Techniques

Bachelor Thesis

submitted by

Jan Claar

on March 01, 2023

Augsburg University
Faculty ofApplied Computer Science
Institute of Computer Science
Chair for Machine Learning & Computer Vision

1st Corrector:	Prof. Dr. Rainer Lienhart
2nd Corrector:	Dr. Andreas Hörner
Supervisor:	Daniel Kienzle

Contents

1. Introduction

The advancement of all major natural sciences has often gone hand in hand with advances in the respective measurement techniques used to formulate and validate hypotheses through experiments. Being able to measure key metrics in the system of interest is vital not only in scientific research, but also the development of new technologies and products.

The field of *machine learning*, especially *deep learning* has seen an explosive growth over the past two decades and has become a major research focus for many in the discipline of data and computer science. With storage capacity and processing speed of computer hardware, such as specialized **G**raphics and **T**ensor **P**rocessing **U**nits (GPUs/ TPUs) experiencing similar advancements, the times are long gone when using neural networks to solve a problem was almost always unreasonable because of its large demand for data and computational resources.

It is no wonder, then, that we ask how we can apply the power of machine learning to the field of measurement science. In fact, this question was asked even before the 2000s [1], at a time when access to computing power was much more limited. Since then, with emergence of deep learning improving what can be done by neural networks, such techniques have been and will continue to be successfully used to improve measurement techniques by helping develop better sensors [2], processing raw sensor data or deriving meaning from already processed data.

One way in which machine learning (in this case neural networks) can be used to improve the measuring process is by automating tasks that are difficult to solve through classical algorithms, but which are easy for a human to do. This is no surprise, since the idea of artificial neural networks in the first place is inspired by their biological counterpart. These problems often are very intuitively solved by the brain for a single instance, but require a large amount of effort when scaled. It is this aspect of machine learning that we hope to employ in our research, where we try to use neural networks to automate the process of measuring droplet size in a vapour produced by a **S**urface **A**coustic **W**aves (SAW).

Surface Acoustic Waves can be produced by using electrodes to induce surface vibrations in piezoelectric substrates. These SAW chips have several useful applications, one of them being using them to vaporize fluids into very small droplets, which could be beneficial in fields such as e.g. medicine, for producing fine vapour of solutions of certain drugs to ensure optimal absorption in the body. While researching and developing this technology it is important to obtain insight about the characteristics of the produced vapour, primarily droplet size and distribution. While there are already techniques to measure the size of droplets in a vapour, they all come with certain caveats. A simple solution is then, to take high speed image data of the vapour and measure the droplet size directly on the images, which is easy for one image, but constitutes an arduous task for each measurement run if one desires to obtain any statistically robust data.

The goal of this thesis is to explore the use of neural networks to automatically identify droplets in image data taken of SAW vaporized fluids and measure them with sufficient precision. This approach will be briefly compared to using a classical algorithm for image segmentation. Finally, a simple application will be developed with the purpose to find use in current research.

2. Theory from the perspective of Physics

Although the thesis doesn't research the atomization technique itself in a large capacity, it is important to understand the physical concepts behind it, since they are responsible for the data that is used in this work.

With this in mind, the following sections will explain how Surface Acoustic Waves are produced and how they can be used to vaporize liquids as well as how the resulting droplets can be measured.

2.1. Surface Acoustic Waves and how they can be created

True to their name, *Surface Acoustic Waves (SAW)* are sonic waves that propagate along the surface of a solid material. The expression is an umbrella term for several kinds of waves that fall under this description, but the type of waves that is relevant to the application researched in this thesis is called *Rayleigh Waves* [3].

Rayleigh waves are a superposition of a longitudinal (P) and a shear vertical (SV) wave component and propagate through the surface of the substrate, with the amplitude of the particle motion decreasing exponentially along the depth of the material. Typically, the effective penetration in the direction normal to the surface is less than a wavelength. Because of how these longitudinal and vertical components the waves produce an elliptical motion in the surface particles, with the plane of the ellipsis being parallel to the direction of propagation and normal to the material surface (see Figure 2.1).

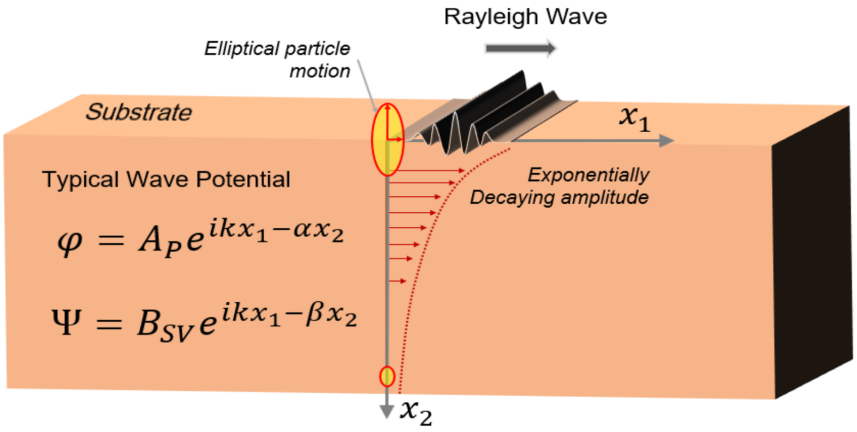


Figure 2.1.: Depiction of Rayleigh-Waves propagating through a substrate. [3]

Other types of SAW waves include *Shear Horizontal Waves* and *Lamb Waves*, however these wave types do not cause the strong vertical displacement which is of interest for the purpose

of liquid atomization. From now on the term SAW will be used synonymously for Rayleigh Waves.

For most applications, SAW are produced by fixing an *interdigital transducer* (IDT) to a *piezoelectric* substrate. An IDT is a type of electrode structure consisting of two sets of interleaved metal fingers, which are connected to the opposite poles of a radio frequency signal source. Applying an RF voltage to the IDT produces surface waves by exploiting the piezoelectric properties of the substrate.

A piezoelectric material generates an electrical voltage in response to applied mechanical stress, and conversely generates mechanical displacement in response to applied electrical voltage. The piezoelectric effect stems from the relative displacement of oppositely charged ions in a crystal with asymmetrical unit cells causing a displacement of the charge concentration and resulting in a larger electric dipole moment. If the dipoles in the crystal are aligned, the effect causes the whole crystal to be polarized, creating an electric voltage. Since the dipole moments in a crystal are typically only aligned locally in their respective Weiss domains, materials usually need to be poled in order to exhibit strong piezoelectric properties [4]. All piezoelectric materials also exhibit the reverse effect; applying an electrical field exerts electrostatic force on the dipoles which causes displacement of the ions.

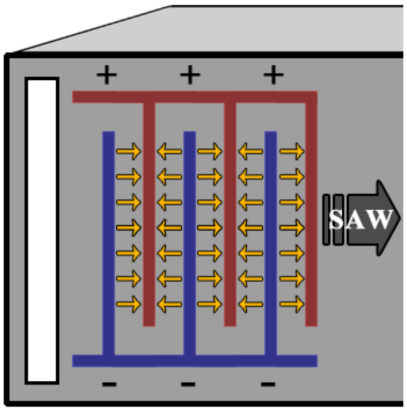


Figure 2.2.: The principle of SAW generation in a piezoelectric substrate. A voltage is applied to the two parts of an IDT, which causes mechanical tension in the substrate.

Since the mechanical force caused by the phenomenon is parallel to the electrical field, applying high frequency voltage to the electrodes arranged in this interwoven pattern (see Figure 2.2) generates surface waves perpendicular to the direction of the electrodes.

The wavelength of the SAW is determined by the distance between the electrodes, which allows for a fine control of the wave frequency for each particular device.

Conversely, the same construction can be used to to pick up vibrations and convert them back to electrical voltage. This is the principle commonly used in SAW based sensors, which is only one of many useful applications of SAW technology like filtering in radio frequency technology or

compact voltage transformers. In the next section the possible application for fluid atomization will be further discussed.

2.2. SAW as a method of fluid atomization

Using SAW technology to produce small scale fluid atomization devices is not a recent idea. The concept was already demonstrated by Kurosawa et al.[5] in 1995, but mass produced SAW-based atomizers are still not commercially available, likely due to challenges related to the precision engineering required for mass production.

However, because of the many possible applications of low-power, compact fluid atomizers such as inhalation therapy [6], thin film deposition [7] or nanoparticle synthesis [8], there is still a substantial interest in improving the technology and optimizing it for production.

Depending on the boundary conditions various acoustofluidic effects take place during the interaction between SAW and a fluid [9]. The geometry of the liquid volume has a large influence on the physical phenomena at play and may result in different atomization regimes, some of which are not entirely understood yet [10, 11]. Describing these highly complex microfluidic phenomena in their entirety is beyond the scope of this work, which is why the following section will focus on the general principle of fluid atomization without going into detail about the governing equations.

When the SAW hits the liquid surface, it is diffracted into the liquid volume at the *Rayleigh angle* $\theta_R = \sin^{-1}(c_l/c_{SAW})$, which depends on the speed of sound in the liquid c_l and the propagation speed of the SAW c_{SAW} and is $\sim 22°$ for water. The acoustic radiation *leaked* into the liquid causes a longitudinal pressure wave which leads to a bulk recirculation of the liquid known as *acoustic streaming*.

This phenomenon is useful for a variety of applications, such as the mixing of liquids or the transport of particles in a liquid, but is not primarily responsible for the atomization process. However, it has an important influence on the geometry of the liquid volume, which has a significant impact on the droplet formation, as mentioned earlier.

The main driving force behind the atomization process is the *capillary waves* that form on the liquid surface. The vertical surface displacement observed for Rayleigh waves is only around 10 nm, however particles are accelerated at around 10^7 m/s^2 [12]. If enough power is used, the capillary waves can be amplified to a point where they overcome the surface tension to break and form droplets.

The exact relationships between the excitation frequency, the capillary wave frequency and the droplet size are not fully understood yet and different models haven been proposed over time. The main influences in the relationship appear to be the liquid viscosity, the liquid surface tension and the liquid density [11, 13]. The latest findings suggest that that the droplet size D

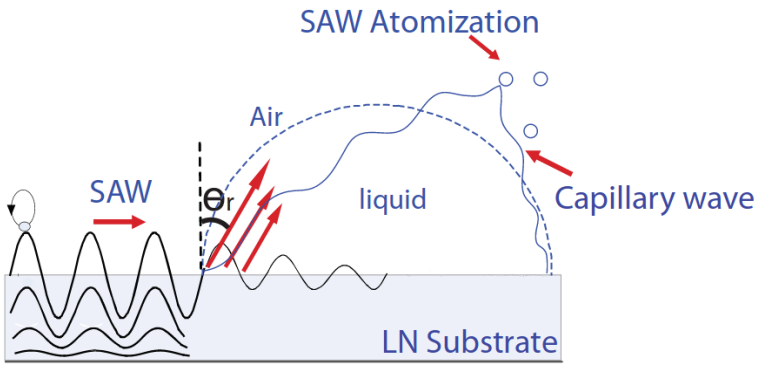


Figure 2.3.: Schematic depiction of the SAW atomization process. The wave hits the and causes capillary waves on the liquid air interface. *LN* stands for *lithium niobate*, a commonly used piezoelectric material. [13]

is in the same order of magnitude as the wavelength of the capillary waves λ_c, which seems to be inversely proportional to the excitation frequency f [10].

$$D \sim \lambda_c \sim \frac{1}{f}$$

Different SAW atomizers differ primarily in how they supply liquid to the atomization area, examples including a *droplet-on-demand* system, where the liquid is supplied by a syringe pump, or a *continuous flow* system, where wetted laboratory paper is placed on the substrate and the liquid is supplied by a reservoir, with atomization taking place at the meniscus that forms at the edge of the paper [9].

The method of liquid supply developed at our institute uses a small capillary channel that is mill-cut directly into the substrate, which has direct contact to a liquid reservoir, constantly refilling itself because of the capillary forces on the liquid. This method is very simple and does not require any additional components, which makes it very suitable for mass production. Ongoing research explores the influence of channel geometry on the atomization process [14].

2.3. Droplet Measurement Techniques

For most applications of microparticle aerosols droplet size is a key variable that needs to be controlled precisely to achieve the best results.

For example, in the case of a drug delivery system, the size of the droplet determines the amount of drug that is absorbed in the lungs, as well as the locations it is deposited in, with smaller droplets reaching more deeply into the lungs, but releasing a smaller amount of drug per unit surface, while larger droplets are more likely to be deposited in the upper airways, but release a larger amount of drug per unit surface area [15].

Therefore, when developing such a system accurate measurement of the droplet size is essential. There are a number of techniques that can be used to measure the size of a droplet, but most of them have some drawbacks or are not suitable for use in this particular application. This section gives an overview on the most common techniques as well as our method.

Laser diffraction techniques make use of the fact that the diffraction angle of light passing a sphere is inversely proportional to the diameter of the sphere, with large particles scattering the light at smaller angles and smaller particles scattering the light at larger angles, according to *Mie theory* for electromagnetic wave scattering [16, 17].

Devices pass a laser beam through the vapour and record the angular scattering intensity, which is then analyzed to calculate the size of the particles.

Theoretically, laser diffraction techniques are very accurate and able to measure particles as small as 0.1 µm and have been used to measure SAW atomized particle distributions previously [13]. However, a company which is in a cooperative relationship with our institute for the purpose of researching SAW atomization has reported inconsistencies in their use of a commercially available device, which is why we have chosen to investigate other techniques. Another reason for choosing other techniques is the potentially prohibitive pricing of the laser diffraction devices in this stage of the research.

Phase Doppler Particle Analysis (PDPA) uses two laser beams that cross each other in a volume with an ellipsoidal cross section in which an interference pattern between the lasers is produced, forming fringes of differing light intensity [18]. When a particle passes through the volume, this fringe pattern is scattered, which produces pulses of light called *doppler bursts* that can be picked up by two or more well placed photo detectors. Information about the particle can be extracted from the phase difference of the doppler bursts picked up by the different detectors.

PDPA is a very accurate technique, but requires very large and specialized equipment, with commercial devices facing similar drawbacks to laser diffraction devices in terms of price and size.

It is also known to produce inaccurate readings for inhomogeneous particles or other disturbances in the measurement environment, which requires a more complicated set up [19].

Image Analysis The technique used during our research up until this point employs a more pragmatic approach. A high speed camera in combination with a microscope objective lens and a parallel light source is used to record images of the aerosolized particles as they are being produced. The images were then analyzed manually one by one to determine the size of the particles. While being very flexible and easy to calibrate, this is an extremely tedious and time consuming process.

This is the aspect of the research that this thesis aims to improve upon by utilizing machine learning techniques to automate the process of droplet identification and subsequent size measurement.

Using this kind of image analysis method is not unprecedented, as Sijs et al.[19] have demonstrated a similar approach in 2021, where they use image processing techniques to identify and measure droplets in image data. However, they do not disclose the details of their image processing pipeline, making it unclear if they employ machine learning models for this purpose. While the idea behind their approach seems to be similar, Sijs et al. report a minimum droplet size of 150 µm for their technique, which is one to two orders of magnitudes larger than our expected droplet size of 1 to 30 µm. As a result, their images seem to have much less noise and uneven lighting conditions, which makes droplet identification easier. Our method should therefore be more applicable to the kind of images that we are working with, but could be used for the same purpose when measuring larger droplets.

3. Theory from the perspective of Computer Science

As stated in the introduction, the goal of this thesis is to use a neural network to identify the droplets in the captured image data to measure their size. This problem lies in the domain of *instance level image segmentation*. However, since individual droplets almost never overlap and droplets themselves have a relatively uniform shape, as long as we know what kind of object each pixel belongs to we can identify the different droplet instances after inference. This allows us to simplify the problem to *semantic segmentation*, which is a much easier task. The employed network is a *(Deep-) Convolutional Neural Network* (DCNN).

In the following section, a brief overview of the basics of neural networks, how they work in general and how they can be adapted to a certain task is given. It also explains a classical image processing technique that might be used on a problem like this one in the absence of neural networks.

3.1. Basics and building blocks of neural networks

The idea of artificial neural networks came long before the capability to employ them as a tool and stems from research in trying to model a biological neuron's function [20]. Biological nerve cells receive signals from several other neurons and then send their own signal based on the strength of their inputs. Translating this behaviour to a mathematical model brings us to the *single layer perceptron*.

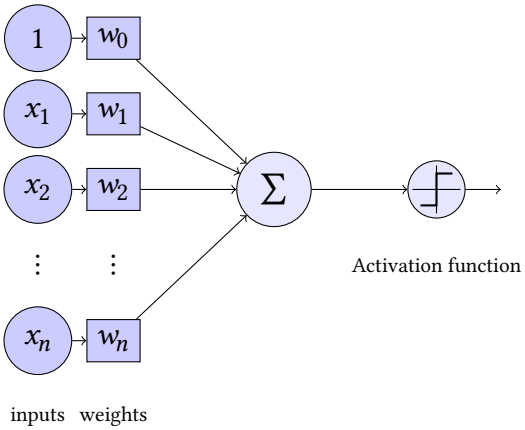


Figure 3.1.: Displayed is a schematic of the single layer perceptron. x_i are the inputs that get multiplied by the weights w_i. w_0 is also called the *bias*. The product of weights and inputs is then summed up and passed to an activation function that computes the output of the perceptron. [21]

Perceptron The perceptron displayed in 3.1 computes its output like

$$o = f\left(w_0 + \sum_i w_i x_i\right)$$

where f is the activation function, which is some differentiable nonlinear function. State of the art networks mostly use the *Rectified Linear Unit* (ReLU) [22] as their activation function, which has proven to be preferable to its predecessor, the sigmoid function, and is computed as such:

$$\text{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & \text{else} \end{cases}$$

Naturally, modern networks contain much more than one neuron and instead contain many such nodes, that are interconnected with each other. These neurons are typically organized in *layers* and form, in the case of feedforward networks, an acyclic directed graph, where neurons of one layer are only connected to the neurons in layers further down the pipeline. Often the layers between output layer and input layer are called *hidden layers*, since the intermediate representation of the input data they compute is not immediately of interest.

While a single neuron doesn't have the capability to solve complex problems, it has been shown that networks with as little as one *fully connected* hidden layer (all neurons of the previous layer output to all neurons of the following layer) can function as a *universal approximators* for functions between two Euclidean spaces[23] or indeed any L^p-Space[24]. This means if there is a mathematical correlation between our desired input and output, so long as both can be represented in these spaces, we can find a neural network that approximates this correlation very well.

While this is an extraordinary finding, modern neural networks often place much more emphasis on the number of layers (depth) than the number of neurons in one layer (width) of a network, hence the terms *deep learning* or *deep neural networks*, since deep networks are easier to train than wide networks. However, the size of a network's parameters and with that its computational complexity grows fast when using many fully connected layers. One tool that enables the use of very deep networks is the *convolutional layer*.

Convolutions and convolutional neural networks Similarly to a discrete 2d convolution in mathematics, a convolutional layer in a neural networks moves over the input space with a comparatively (to input dimensions) smaller kernel and computes the output by multiplying the weights stored in the kernel with the corresponding input values.

The example seen in Figure 3.2 shows only one channel and a procedure that employs no padding, leaving the output smaller than the input (called a *valid convolution*). Other than valid convolutions, padding the input by $\lfloor \frac{k}{2} \rfloor$ or $k - 1$, where k is the size of the kernel, leaves us with a *same* (same size as input) or *full* (larger than input) convolution. Additionally, in reality

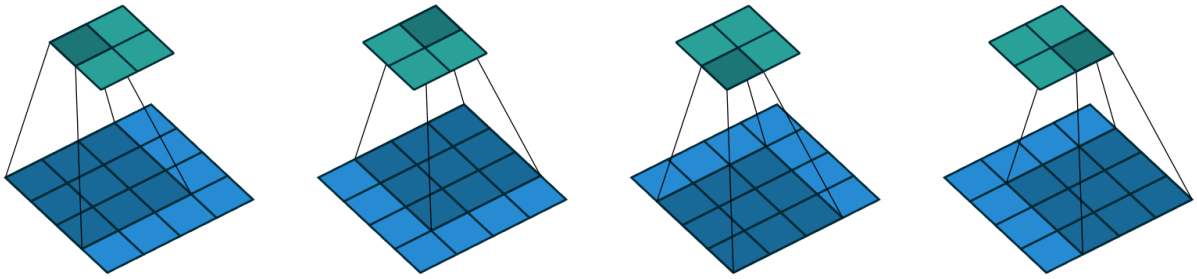


Figure 3.2.: Depiction of how a convolution computes its outputs. A 4×4 input convolved with a 3×3 kernel produces a 2×2 output. Since no padding is used this constitutes a valid convolution. [25]

inputs to a layer often have many channels and the layer should be able to produce an arbitrary amount of output channels. In this case there exists a kernel $K_{\text{in,out}}$ ($K \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}} \times W_{\text{K}} \times H_{\text{K}}}$) for each combination of input channels C_{in} and output channels C_{out} whose convolution results are then added up for each output channel, so that for an input $I \in \mathbb{R}^{C_{\text{in}} \times W \times H}$ the output $O \in \mathbb{R}^{C_{\text{out}} \times H' \times W'}$ is calculated like

$$O_j = \sum_{k=1}^{C_{\text{in}}} K_{k,j} * I_k \quad ,$$

where $*$ is the convolution operation. There are other approaches to handle multi channel scenarios, but this is the simplest one and is employed in the thesis. Another parameter which is important in the context of convolutions is the *stride*, which determines how many units the kernel is moved in each step. The example shown in Figure 3.2 has a stride of 1, but often, a stride of 2 or more may be used when utilizing convolutions to downscale inputs.

Using a convolution layer has several advantages over using fully connected layers. Using a convolutional layer is a reduction in *learnable parameters* (see section 3.3) for identical input and output dimensions. This reduces computational resources required and thereby speeds up training. It introduces the inductive bias that positionally close data points are correlated and relevant to each other, which is reasonable especially in tasks concerning computer vision.

For example, take a model that tries to detect circles in an image. It makes sense to assume that features which make up a circle are located closely to each other. Convolutions are also translationally invariant, because they use the same kernel weight for the entire image, regardless of the position. In this example that means that for two identical circles at different locations in the input, the identical output is produced at their corresponding position, which is not the case for fully connected layers, where very different weights might me learned for different positions in the picture.

Many architectures in the computer vision field now purely employ convolutional layers (*fully convolutional neural networks*) or use very few fully connected layers only to compute the final output of the network at the end.

There are additional constructs that can be used in neural networks, such as *pooling* layers, which also move a kernel over the input like a convolution, but instead take the average or maximum of the covered elements as their output. As such, the pooling layer contains no learnable parameters. They are generally used to reduce the dimensions of the feature map to further speed up training time and make the model more stable by promoting invariance to small perturbations in the input as well as increasing the receptive field of the neurons.

3.2. Semantic Image Segmentation

With the knowledge of how neural networks perform their computation in general we need to think about how a specific task can be accomplished by them. To do this we need to find a specialized representation of the problem which networks can be applied to.

There are several common types of problems within computer vision that have different kinds of encodings for their solution space. As stated in the introduction, the task which this thesis is trying to solve lies in the category of *Instance Segmentation*, but can be accomplished by instead doing *Semantic Image Segmentation*. For this task the network input is an image and the network is supposed to assign each pixel one of a predetermined set of *classes* to which the pixel belongs to. For example in the case of the *Cityscapes Dataset*[26], the data consists of images of inner city traffic situations from the perspective of a car, and the classes are subjects like *car*, *road*, *person*, *building* etc.

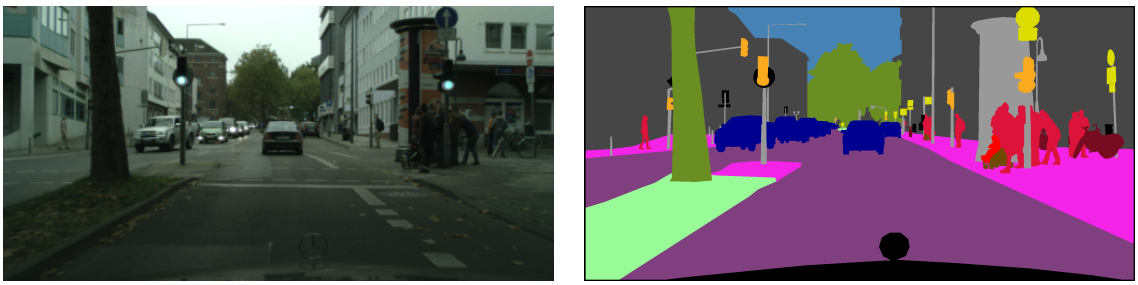


Figure 3.3.: A sample from the dataset *Cityscapes*. On the left is the image that acts as an input for the network and on the right is the corresponding ground truth label mask showing different classes in different colors. [26]

A model for this task would now output a map $O \in \mathbb{R}^{C \times M \times N}$ with the same spatial dimensions as the image and a number of channels C equivalent to the number of possible classes. Each output pixel consists of a vector $\mathbf{x} \in \mathbb{R}^C$ where each entry x_i corresponds to how much the network thinks the pixel belongs to class i. The channel with the highest value is the class the network has classified the pixel as.

These outputs can now be normalized to probabilities by applying the *softmax* function to them or converted to a segmentation mask by using the *argmax* function.

To evaluate how well a network performs on the segmentation task, the *Intersection over Union* (IoU) for each class is computed. For the set A of pixels labeled as a certain class by the network and the set B of pixels which are part of that class in the ground truth the IoU is defined as

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \quad .$$

This metric is very informative to look at per class when trying to figure out specific strengths and weaknesses of the model, but typically the mean over all classes (mIoU) is used to evaluate the model as a whole.

A big challenge in developing machine learning solutions for this kind of task is its demand for varied data to train models on. High quality annotated data is very time and cost consuming to produce. For example according to the Cityscapes team, one annotated image such as seen in Figure 3.3 took 90 min to create. Further in this thesis techniques to mitigate this problem and achieve good results on a small set of annotated data will be discussed.

3.3. Training neural networks

As mentioned previously neural networks with appropriate weights can be used to approximate any mathematical mapping between two Euclidean spaces (and more), but which weights are appropriate is not at all apparent for any nontrivial task. Similarly to how humans don't immediately perform at their potential best at a new task, but can train the brain to improve at solving a particular problem by strengthening the corresponding nervous pathways through repeated exercise, a neural network can *learn* the correct weights for a task during *neural network training*.

In most cases, training a neural network requires annotated data, where the ground truth for the problem at hand is known and encoded in a way that is compatible with the output format of the neural network. Keeping in line with the topic of this thesis, for a pixel level semantic segmentation task, this would be in the form of segmentation masks where each pixel has the ID of the correct class as its value.

Training happens in several steps. First, a sample from the training data is taken and the model, typically initialized with (semi-) random weights, computes its prediction for the problem (a segmentation mask). Then, a *loss function* is applied to measure a distance between the prediction and the ground truth in the solution space. There are several functions which can be used as a loss function for one particular problem which may or may not perform better than others in expressing how wrong the model was with its prediction. Next, the gradient of the loss with regards to each learnable parameter is computed by using the chain rule for derivatives to propagate the error backwards through the network (*backpropagation*). The parameters are then very slightly adjusted to reduce that error. There are several different rules for updating the

weights with this gradient in each step, but widely used procedures include *stochastic gradient descent (SGD)* and *adam* [27].

There are several parameters that influence the progress of the training or even how the model operates, but aren't learnable parameters like the weights of the model. These parameters are called *hyperparameters*. This includes the *learning rate*, which influences how much the parameters get updated in each step, but also things like the kernel size of a convolutional layer. Most of the time, samples aren't processed one by one, but stacked up in *batches*, whose size (the *batch size*) constitute another important hyperparameter. Processing multiple samples at once can be more efficient and also make the training process more stable by presenting more varied data to the network during each optimization step. However, batch size is limited by the size of the training samples in relation to the memory of the hardware used, so it can't be made arbitrarily large.

Which hyperparameter values are used during training can have a huge impact on performance, so finding optimal hyperparameters is vital when developing a model. The appropriate hyperparameters depend a lot on the training data and model architecture and are largely determined through trial and error. There are systematic approaches to hyperparameter tuning, but these are often very computationally expensive, since they perform a lot of trial runs. For simpler problems it is often enough to perform the search manually, by starting at values which are known to be in the correct order of magnitude for similar problems and then make informed adjustments based on the results of a few trial runs.

Another important part of model training is *data augmentation*, which means to apply random perturbations to the inputs or even the model itself (in the case of *dropouts*) during training to prevent the model from *overfitting*. This refers to the model *memorizing* the training data instead of learning the concept of the problem, leading to high performance on the training data at the cost of worse performance on unseen data, which is undesirable in most cases. The noise applied by augmentation reduces the likelihood of this happening and generally improves model performance. Since the augmentations are random each time, they functionally produce additional samples, which can help especially in cases with little training data. Some typical augmentations for computer vision tasks include *random horizontal/vertical flipping*, *scaling and resizing* or *applying gaussian noise*.

3.4. Classical Algorithms: Hough Transform

The *Hough Transform* is an example of a classical algorithm (meaning not involving the use of machine learning) used in computer vision to solve the problem of line detection. It was invented by Paul Hough [28] and further developed by Duda and Hart [29] and works by employing a voting procedure in a parameter space to find the most likely lines in an image.

A line can generally be described by the equation

$$y = mx + b \quad ,$$

where m is the slope and b is the y-intercept. In the algorithm, we start with an empty accumulator array, each of whose entries corresponds to a combination of discreet values for m and b. Then we iterate over the image and for each pixel that could be part of a line (decided by e.g. its greyscale value), we vote for all possible lines that could pass through that pixel by incrementing the corresponding entries in the accumulator array. The lines with the highest votes are then most likely to be present in the image.

Since a circle can similarly be described by the equation

$$(x - a)^2 + (y - b)^2 = r^2 \quad ,$$

where a,b are the x,y-coordinates of the center of the circle and r is its radius, the same principle can also be applied to adapt the algorithm to detect circles. However, since since the parameter space for circles is 3-dimensional, this becomes very inefficient for large images.

It is possible to improve the efficiency of the algorithm by first calculating the gradient of the image to detect edges and then only voting for points in the direction normal to the edge of the circle [30, 31].

Since droplets are generally circular, applying the Hough transform in some way to detect them might be a good idea.

4. Architectures

When approaching a problem with a machine learning solution in mind, the first decision one has to deliberate on is the choice of network architecture used.

In the case of neural networks, the word *architecture* means which kind of layers are connected to each other in what order to produce the output of the network.

The landscape of machine learning architectures has become incredibly diverse, with improvements being made constantly. Since some architectures are better suited for certain problems or priorities, which architecture one chooses has a significant impact on the results.

In this section, an overview over the architectures used in the thesis and their main ideas will be given.

4.1. The U-Net

The *U-Net* is a network architecture first proposed by Ronneberger et al.[32] in 2015 for application in biomedical segmentation tasks, for example segmenting cell borders in microscopic images of HeLa cells. The challenges the authors were facing at the time are similar to the ones in this thesis, where training data for biomedical segmentation tasks was scarce, which is why the U-Net makes sense as a starting point for the problem of droplet segmentation.

It is a fully convolutional network, meaning it consists of convolutional layers only. The network employs an *encoder-decoder* structure, meaning the architecture is made up of a contracting path and an expanding path, the *encoder* and *decoder* respectively, as seen in Figure 4.1.

The encoder path computes a number of features at different spatial resolutions with two 3×3 convolutions followed by a ReLu, which are then downsampled by a 2×2 max-pooling layer. With each of these blocks, the spatial resolution is halved along each axis, while the number of feature channels is doubled.

The features output by each block depend on differently sized regions of the initial input. This means the values of blocks with high spatial resolution consider only small patches in the input image, while a lot of pixels influence each value for the lower blocks. The large *receptive field* of the lower blocks allow for rich contextual information to be encoded in their feature maps.

In a classification problem, these kinds of contracting networks would be used by feeding the highly dense information of the last encoder block to a few fully connected layers which make the final classification decision. In this case, since a classification for each pixel is needed instead

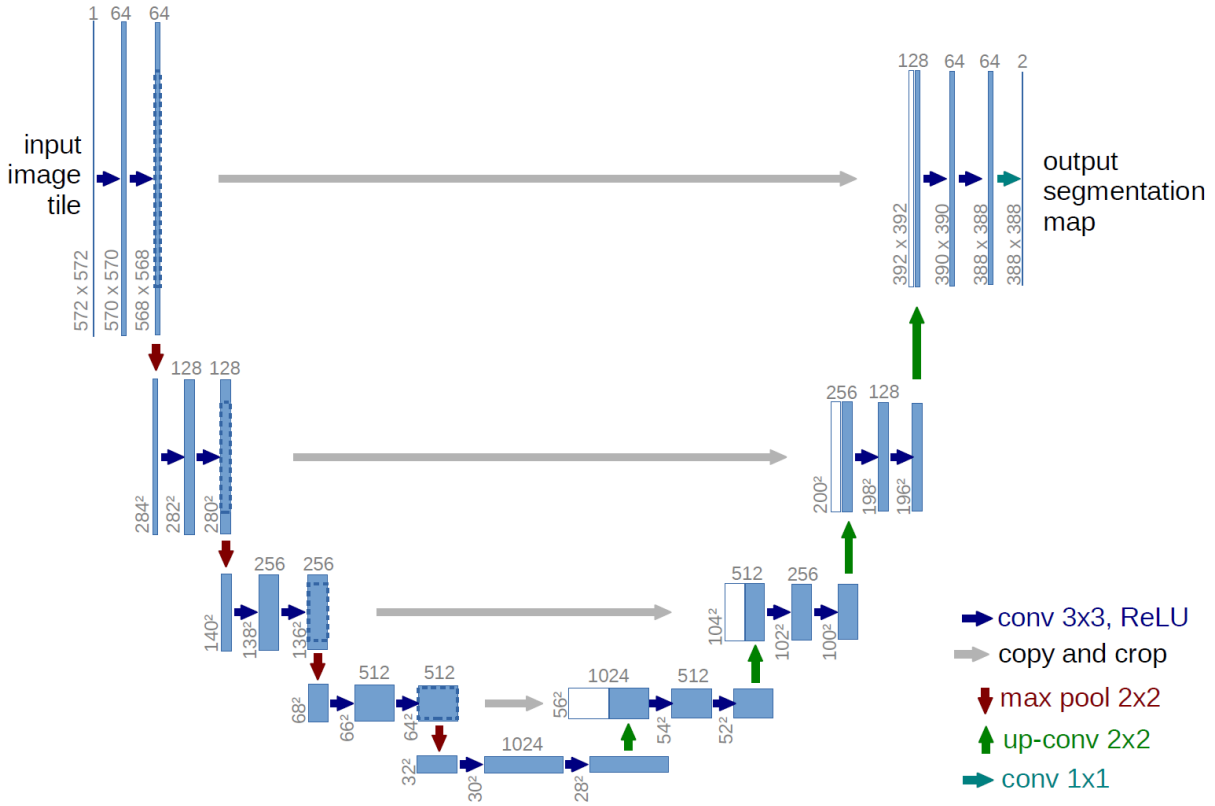


Figure 4.1.: The original U-Net architecture proposed in [32], for an example of a 572×572 input, with the number of channels of the layer written above the blue boxes representing the feature map after each layer passthrough. The legend shows which operation was used between each feature map. [32]

of just one for the entire image, the encoded information must now be scaled back up to the desired resolution.

The decoder path is symmetrical to the encoder path, halving the number of feature channels in each block and upsampling the spatial resolution by using 2×2 *transposed convolutions*, which act like a backwards pass through a normal convolution. After the last upsampling block a final 1×1 convolution is used to decide the final class for each pixel.

However, as is, this structure has a key flaw when it comes to creating segmentation masks. While it is feasible to deduce general locations of objects from the highly contextual encoder features, it is difficult to infer their exact boundaries, because the spatial resolution is compressed so much.

This is where another key aspect of the U-Net architecture comes in. By concatenating the outputs of each encoder block to the input of their respective decoder block, we allow the decoder to not only utilize the context information provided by the last encoder layers, but also the very localized features of the high resolution blocks.

This combination allows the U-Net to predict the correct classes along with their precise spatial location.

The U-Net outperformed its predecessors by a large margin and has since been developed further. Its concepts still serve as the basis for popular segmentation networks.

One way in which the original U-Net architecture is often modified is using more sophisticated models for the encoder module, which is the approach taken in this thesis. Obtaining better encoder features has proven itself to lead to better overall results, so investing more capability in the encoder structure is often a good idea.

This also comes with the added benefit that pretrained weights for common feature extractors like the *ResNet* (more in section 4.2 and 5.1) are readily available.

4.2. The ResNet

The *Residual Network (ResNet)* is a deep neural network architecture first introduced by He et al.[33] in 2015, which addresses the seemingly paradoxical circumstance that adding additional layers to a deep neural network would degrade its performance compared to a shallower network, even though the solution space of the shallower model is a subspace of its deeper counterpart.

Working from the idea that adding more layers to a network should not produce a higher error, since the additional layers could potentially just learn identity mappings, the authors surmised that the problem had to do with the difficulty for the optimization process to learn the underlying desired mapping for a set of layers if the ideal mapping is closer to an identity mapping than a zero mapping.

The solution to this, proposed by ResNet, is to utilize residual connections between the input and output of a layer group, by directly adding the input to the output. Instead of having to learn the desired output mapping $\mathcal{H}(x)$, the layers now have to fit the residual mapping $\mathcal{F}(x) := \mathcal{H}(x) - x$, which the authors argue is easier for the optimizer to do.

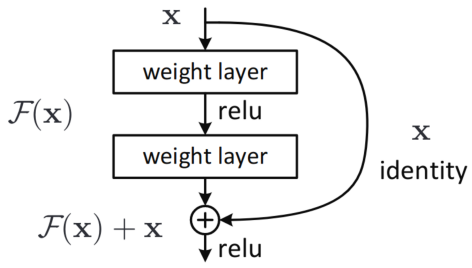


Figure 4.2.: A depiction of a *residual block* employed in the ResNet network architecture. The input of a group consisting of several layers is added directly to the output. The weighted layers can be linear layers but in practice, mostly convolutional layers are used. The blocks also consist of 3 layers the majority of the time. From [33]

Employing these *residual blocks* (Figure 4.2), the convergence rate of the network is significantly improved, without adding any computational complexity. This enables the use of very deep networks, gaining substantial accuracy from the added layers.

Apart from the residual connections, the ResNet still follows a typical encoder architecture, with each block reducing spatial resolution, while increasing the number of feature channels. There are several versions of the ResNet architecture, differing mainly by the number of layers used. He et al. looks at networks with up to 152 layers (ResNet152), but deeper networks have already been explored.

Since the residual connections allow the network to propagate information from the shallower layers to the deeper layers more easily, it may also combat the problem of vanishing/exploding gradients, which was a challenge for increasingly deep models at the time. However, the authors argue that this was already sufficiently addressed by regularization techniques such a *Batch Normalization* [34].

ResNet was able to significantly outperform its state-of-the-art predecessors at the task of image classification and is widely used today, often as part of a larger model ensemble.

In this thesis, *ResNet34 D* is used in most experiments, which makes several minor improvements over the regular ResNet. This includes employing an average pooling layer before the strided 1×1 convolution present in the identity path of each downsampling block (first block in each colored stage in Figure 4.4) to include all data points, switching the stride of convolutions in the residual path for the same reason and replacing the 7×7 initial downsampling convolution with three 3×3 convolutions. These modifications see an improvement in classification error while only increasing computational cost slightly [35]. For visual comparison, see Figure 4.3.

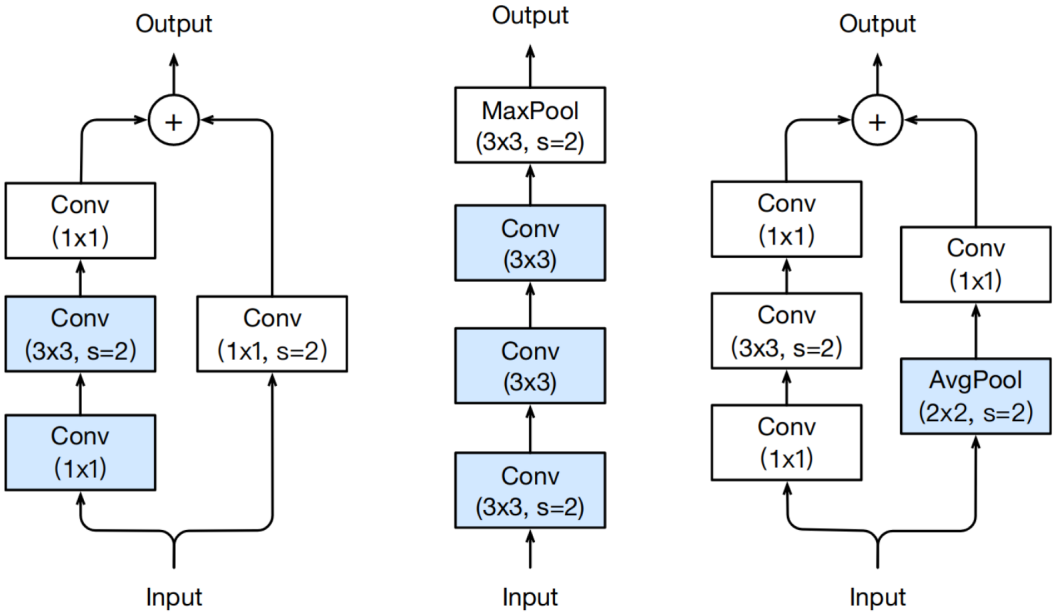


Figure 4.3.: Illustration of the three improvements made by ResNet34 D, with the blue boxes representing the changes made. Left depicts the change made to the residual path of the downsampling block, middle illustrates the changes made to the initial block of the network and right explains the changes to the downsampling identity path. s represents the stride of the operations ($s = 1$ if omitted). From [35]

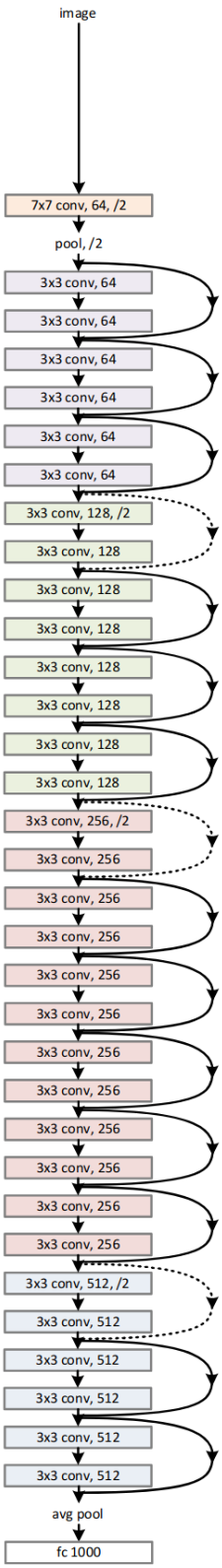


Figure 4.4.: Diagram of the ResNet34 model. Curved arrows represent a residual connection, with dotted arrows meaning the connection also downsamples the input. The four encoder blocks are colored differently. For use as an encoder, the fully connected layer at the end is omitted. From [33]

5. Techniques used to improve model accuracy

Basic steps such as optimizing hyperparameters for training are not the only methods available to try to improve the performance of the resulting model. Other techniques may be used to achieve better results, either by facilitating learning in some way or exploiting more data. In the following sections, the techniques employed in this thesis will be explained more in depth.

5.1. Transfer Learning

Coming back to the human learning analogy, it is often the case that knowledge or skills gained in one field can be applied to other fields, without having a lot of experience in these new disciplines. For example one might take their knowledge about composition from drawing to apply it to taking good pictures in photography. Although the technical details of both fields are different, they have some shared concepts, which help a person proficient in one to perform well in the other. That same person might also be able to improve in this new discipline faster than others, who do not have any applicable prior knowledge, because they can focus on learning other key skills needed to excel.

Transfer leaning is the idea to apply this concept to neural network training. From a high level perspective, what happens during computation in a neural network, is that each layer transforms the input into an *intermediate representation* of the data, which extracts different features present in the input. For example, one layer might learn to detect edges, which the next layer then combines to knowledge about the location of corners. As you can imagine, these types of features are useful for recognizing objects, such as a tennis ball, in a picture. It also stands to reason that these same features would be useful to detect circular droplets. The extracted features are rarely this clear cut or human-understandable as in this example, but the point remains the same.

Using a model trained on one dataset and using these weights as a starting point to fine tune the model on the actual problem dataset is also called *pretraining*. This is especially useful if the data for the target problem is scarce, as pretraining is often done on very large datasets such as *ImageNet*[36], which features over 14 million images (smaller subsets available) annotated for *image classification*. If the target problem is similar to the problem the model was pretrained on, the whole model can be used as a starting point for training. However, pretraining is also possible on a task different from the actual problem, in which case the pretrained model can only be used in a larger ensemble.

Pretrained weights for popular architectures are readily available for download and are employed in this thesis by using them as a feature extractor in an *encoder-decoder* architecture (see section 4)

There are other forms of transfer learning, such as *knowledge distillation*, where the concept is to use a complex model that is well adapted to the task to train a comparatively smaller model. The larger model has a higher knowledge capacity, but not all of this capacity has learned important knowledge. When training the smaller model on the soft outputs (before argmax) of the larger model, it may learn correlations which it might not have been able to learn on its own given its limited capacity. The smaller network could then be deployed instead of the larger one to save computation resources on weaker hardware.

In this thesis pretraining is used in two places. Firstly, for the encoder module of the U-Net architecture, a ResNet that is pretrained on the ImageNet dataset is used. Secondly, the experiments examine if pretraining the model on the Cityscapes dataset helps to improve performance on the vapour image dataset.

5.2. The mean teacher approach to semi supervised learning

As mentioned already, the starting hurdle for building machine learning solutions is obtaining enough high quality annotated data. For image segmentation tasks, this means annotating segmentation masks, which are notoriously time consuming to produce. In our case, no annotated data was present in the beginning and it would be unreasonable to spend a large amount of time to annotate a lot of images, so only a limited amount of labeled samples were produced. In contrast, unlabeled images can be obtained very quickly and in large amounts, as one good measurement run can produce hundreds to thousands of images (even though not all of them may be usable).

Thus it would be very beneficial if we could make use of this unlabeled data in some way during training. Approaches to learning that utilize both labeled and unlabeled data are called *semi-supervised learning*, in contrast to *(fully-)supervised* or *unsupervised learning*, where only labeled or no labeled data is used respectively. One example for such a procedure is called the *Mean Teacher approach* [37].

The main idea of the Mean Teacher approach it to use two models, one *student* and one *teacher*, and have the student learn from examples produced by the teacher.

The reason unlabeled data is not directly usable for model training is that no classification cost can be applied to the outputs, as the target is undefined. While there is no way to automatically generate a 100% accurate target, as this is essentially what we try to train our model to do, when we look at it the other way around, we can use a model to approximate the target to a degree. However, using the same model we want to train to simply approximate its own

training samples would not provide any benefit. Two steps are are used in the Mean Teacher method to still make use of these kinds of *pseudolabels*.

As mentioned in 3.3, augmentation and regularization techniques have the purpose of enabling the model to learn the concept of its target function more broadly, since small variances in the input should still produce a similar output. This can be applied not only when comparing the output of the model to the ground truth, but also when comparing the outputs of the model for the same input data, but different levels of noise. If the model has properly learned the correct abstractions, its prediction should be similar for slightly different inputs, even if the outputs are less than perfect. What this means concretely, is that the method uses the teacher to make a prediction on an input without any added noise and then computes student predictions on the same input to which augmentations have been applied. In general, the teacher should have an easier time to predict accurate labels for the sample without the added noise, so a slightly better approximation of the theoretical correct labels should be produced. A *consistency cost* can then be applied between student and teacher predictions which are then used in the same way as the *classification cost* to update student weights. Note that if the augmentations transform the input's geometry the teacher predictions have to be transformed accordingly.

Another way to improve the relative quality of the pseudolabels is to improve the teacher model they are generated by. This is where the second key concept of the method comes in. Instead of using the same weights for both student and teacher models, the *exponential moving average* (EMA) of the student weights are used for the teacher. What this means is if θ_t are the weights of the student at step t and θ'_t are teacher weights at the same point, the teacher weights θ'_{t+1} are updated as follows:

$$\theta'_{t+1} = \alpha\theta'_t + (1 - \alpha)\theta_t$$

where α is a smoothing coefficient called the *EMA decay*.

The EMA of a model has been observed to be slightly better than the model itself on average, as well as being more stable during the training process, since changes to the student model are only adapted slowly. Combining this with the augmented student inputs makes it possible to extract a lot of improvement from unlabeled examples, but can also be applied during training steps with labeled data (see Figure 5.1).

During training, a relative weight λ is applied between classification loss L_{class} and consistency loss L_{cons}, so that the overall loss is given by:

$$L = L_{\text{class}} + \lambda L_{\text{cons}}$$

The weight of the consistency loss should start low and be slowly ramped up over the period of the training. This is necessary since initially, the teacher model may produce very bad outputs. These outputs might also differ a lot from the student outputs, since no concepts haven been learned yet. If the consistency is given too much weight the model will prioritize

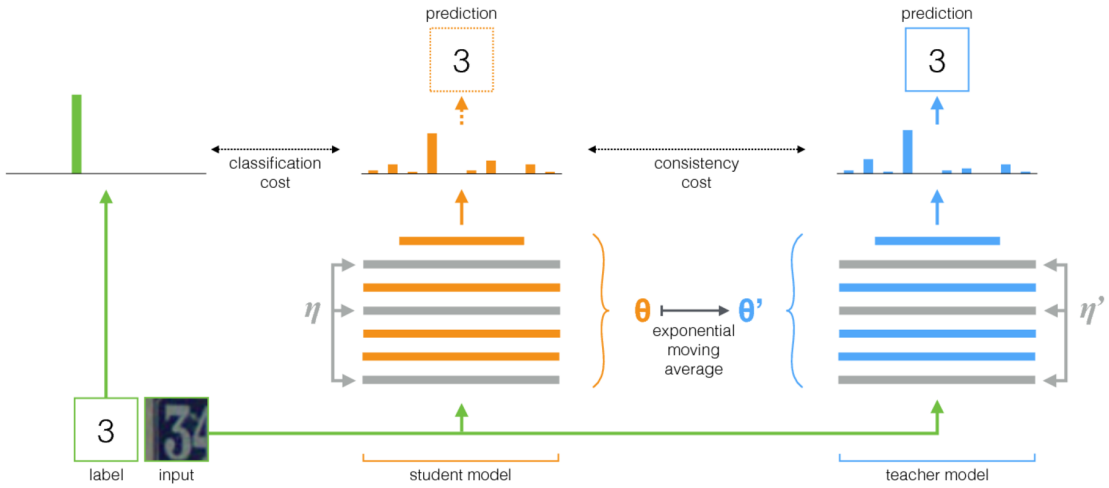


Figure 5.1.: Schematic depiction of the Mean Teacher method. The graphic illustrates a single training step with one labeled sample for the image classification task. The classification loss is applied between the ground truth and the model predictions, while the consistency cost is computed between the soft outputs of both models. After updating the student weights with backpropagation, new teacher weights are computed as the EMA of the student weights. Steps with unlabeled samples simply omit the classification cost. Image taken from [37]

being consistent with the teacher over predicting the correct classes for the labeled examples, potentially hurting or preventing any training progress.

Another key aspect of the method is considering which augmentations are chosen for the student inputs [37]. To produce a large enough difference between outputs, strong augmentations are necessary to achieve the best improvements. The augmentations should also be well tailored to the dataset that is being trained on.

Lastly, as mentioned above, the method introduces the hyperparameter α into the training, which has a big influence over the improvement that can be achieved. If it is too low, the benefits of using the EMA may not be present as much, since the teacher is too similar to the student. If it is too high, the teacher may not be updated quickly enough to incorporate the students improvement. Generally a value of α between 0.99 to 0.999 is used, which has been shown to work well in practice.

The applicability of the mean teacher method will be more closely examined during the experiments of the thesis, with the goal to improve overall accuracy as well as generalization capability of the model produced.

6. Droplet detection

Before discussing the experiments conducted in this work, it is important to understand in which context the final model will be used to process experimental data. The following section also discusses the limitations of the model and the experimental setup.

6.1. Detection and measurement algorithm

The measurement process consists of several steps, each employing different techniques to improve measurement accuracy and reduce computation time.

Step 1: Image preprocessing is done on the raw image data. This step is done to improve visibility of the droplets in the image and make a first guess as to which images actually contain droplets. The preprocessing is done batched by calculating the mean image $\mathbf{m} \in \mathbb{R}^{H \times W}$ as well as the mean greyscale value $\mu \in \mathbb{R}$ of the batch $\mathbf{B} \in \mathbb{R}^{N \times H \times W}$, subtracting $\mathbf{m}$ from each image in the batch, adding μ to each pixel and then mapping the resulting values to the interval $[0,255]$. This process must be done batched because of memory constraints, but batching the images also has the advantage that images in one batch are more likely to be similar to each other in terms of lighting conditions, which makes the output more consistent. This only applies if the images are taken in one measurement run, which is the intended use case.

The mean image subtraction is done to remove camera and lens artifacts such as hot pixels or dust from the images. This is not primarily done to improve the models detection accuracy, but in order to filter out images that are not suitable for further processing. The metric for deciding if an image contains any structure that could be a droplet is the *Michelson contrast*, which is defined as

$$C = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}},$$

where $I_{\max}$ and $I_{\min}$ are the maximum and minimum luminance values in the image. Since droplets have very dark and very bright regions, they produce a high Michelson contrast. Images aren't considered for further processing if C is below a certain threshold. The reason artifacts need to be removed is that the they are very dark compared to the background, which means even images without any droplet structures will have a high Michelson contrast.

Lastly, before applying the model to the images, they are normalized to have a mean of 0 and a standard deviation of 1 using ImageNet mean and standard deviation to match the training augmentations.

Training data underwent the same preprocessing steps.

Step 2: Droplet detection is done by a model trained to identify droplets with dark borders and bright centers as in-focus and segment the images into three classes, *droplet border*, *droplet inside* and *background*. Segmenting the images into three classes instead of two is important in the next step, since it helps to detect errors the model makes in the segmentation process.

Details on model training and evaluation can be found in section 7.

Step 3: Droplet measurement is done by first grouping all adjacent pixels labeled as *droplet border* in the segmented image to distinct instances. This is essentially done by choosing a starting pixel with the correct label and scanning its neighboring pixels to find all pixels that are also labeled as *droplet border*. This step is repeated for this new set of pixels until no more pixels with the correct label can be found. The process starts again for a new pixel that has not been assigned to a droplet yet, until all labeled pixels have been processed. The library *scikit-image*[38] used to accomplish this employs several tricks to speed up this process [39].

As a first filtering step, all areas that are smaller than a certain threshold are now discarded, since no droplets below a certain size can be expected to fit the criteria for being in focus. For each remaining area, the locations of all pixels are averaged to locate the center of the droplet. Since the droplet is approximately circular, the average distance of all pixels from the center is calculated and used as a measure for the radius of the droplet. Because the border of the droplet has a certain width and labels sometimes extend beyond the actual droplet, only the values between the 80th and 95th percentile of the distances are used to calculate the radius.

6.2. Common problems and limitations

The model used to segment the image is not perfect, which means that sometimes areas which do not belong to a droplet are labeled as *droplet border* or *droplet inside* or the model fails to detect a droplet that is actually present in the image. In this section, the most common errors are discussed and possible solutions are proposed, some of which are implemented already.

The model labels an out of focus droplet or other noise partially with border and center labels. This is the most common error observed when applying the model to experimental data, examples for which can be seen in Figure 6.1. The model will assign a border label to the dark pixels of an out of focus droplet or a center label to brighter areas of the image that are not part of a droplet, possibly even both for the same structure. However, this will very rarely happen in a way in which the border region completely surrounds the center region in a closed shape, which can be used to filter out these errors.

Training the model to assign two different labels for the droplets instead of one allows the criterion of droplets having a closed border around a center region to be used not only during

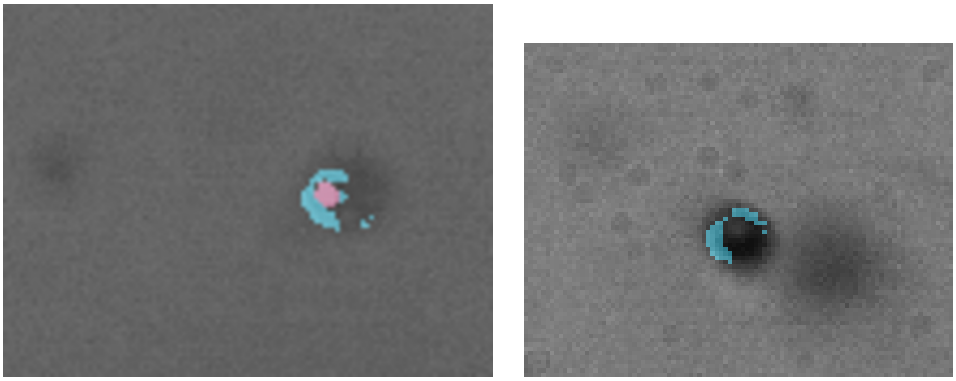


Figure 6.1.: Example images showing a labeling error as described in section 6.2. Blue pixels represent *droplet border* labels, and pink pixels represent *droplet inside* labels.

training, but also after segmentation. To check if a border area such as the ones described in Figure 6.1 fulfills this criterion, the algorithm calculates its *alpha shape*[40] and then checks if any of the pixels inside the shape are labeled as *droplet inside*.

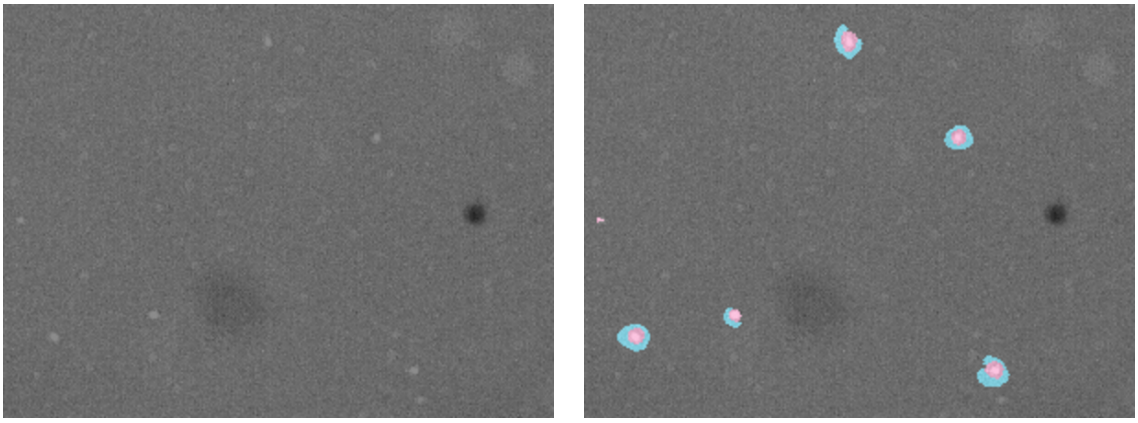
The alpha shape of a set of points is a generalization of their *convex hull* and for a real number α includes all edges between the points for which a *generalized disk* with radius $\frac{1}{\alpha}$ can be drawn so that the points of the edge lie on its border and the disk contains no other points. For $\alpha = 0$, the generalized disk becomes a half-plane and the alpha shape is equivalent to the convex hull. In the code, the alpha shape is calculated using the *alphashape*[41] package, which first calculates the *Delaunay triangulation* of the points and then uses the criterion described above to determine which edges to include in the shape. The parameter $\alpha = 1$ is used since the pixel coordinates are discrete and the pixels are directly adjacent to each other.

This method of filtering works very well for any but the most extreme cases and improves the accuracy of the measuring process significantly.

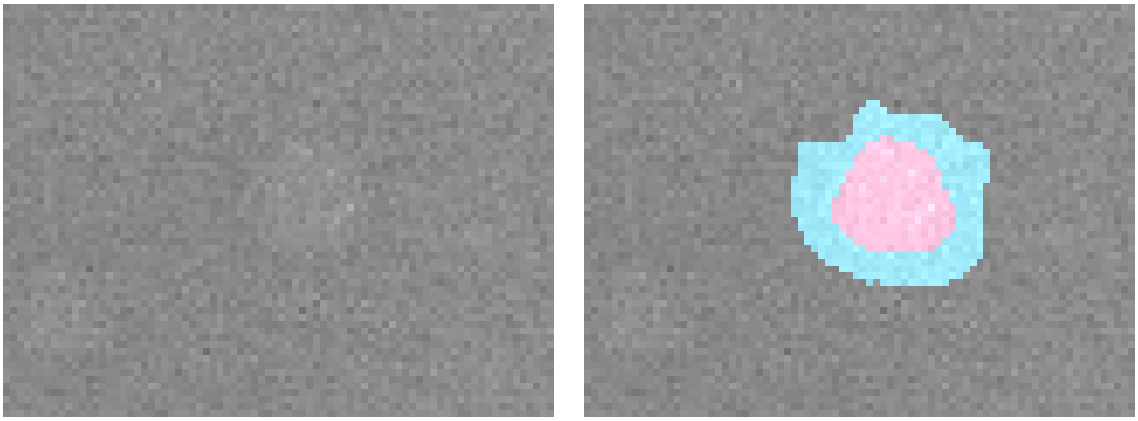
The model labels an out of focus droplet or other noise completely with border and center labels. This is a much rarer error than the one described in the previous paragraph, but it can still happen. It is mostly encountered when the image is very dark overall, which ironically causes the preprocessing step described in section 6.1 to produce bright artifacts in such images, instead of removing dark spots. One example of such an image can be seen in Figure 6.2a. Since the model identifies in-focus droplets by their bright center, this behaviour is not completely unexpected.

However, very rarely the model will assign a false positive even to an area which is not significantly brighter than the background, as can be seen in Figure 6.2b.

This kind of error is much more severe, since there aren't any criteria to differentiate these kinds of false positives from the structure of the actual droplet labels. Thus, the only way to mitigate this that comes to mind are improving image preprocessing and/or training the model differently to not make these kinds of mistakes.



(a) The model identifying bright artifacts in dark images as droplets. Note that only bottom left and top right labels are fully closed shapes, which can't be filtered out.



(b) The model identifying a comparatively brighter spot as a droplet.

Figure 6.2.: Examples for cases where the model confidently labels an out of focus droplet or other noise completely with border and center labels. The left image shows the original image, the right image shows the segmentation mask. Blue pixels represent *droplet border* labels, and pink pixels represent *droplet inside* labels.

One idea to improve the preprocessing is to use a different technique for removing the artifacts. The current method of removing the artifacts by subtracting the background image from the original image is very simple and works well if the images in the batch are similar in brightness, but it is not very robust to outliers. A different approach then, would be to use the background image as a mask for identifying artifact locations instead, and overwrite the pixels in the original image with a more appropriate value. This could either be the mean brightness of each individual image, or the values at the same pixel location in the original image with a gaussian blur applied to it. A process like this could help in minimizing bright artifacts, which seem to be the main cause of this kind of error. For this, *inpainting methods* [42–44] could do a good job, however they were found to be too computationally expensive for the task at hand.

To instead make the model more robust to these kinds of artifacts, some kind of data augmentation could be used that mimics the effect of the artifacts, such as *salt-and-pepper/binary noise*. Another option would be to explicitly use additional samples of images the model seems to find

difficult to classify correctly, such as the ones shown in Figure 6.2. These examples would need to be labeled of course, but could potentially improve the model substantially in this regard.

Very fast droplets that travel a significant distance in the time it takes to expose the image (inv. proportional to *shutter speed*) appear as streaks instead of circles. Since the model is trained to identify droplets that are ellipsoidal in shape, it won't be able to label those streaks correctly. The structure of these streaks is also much less distinct than the structure of the droplets, which makes it harder to say if a streak is in focus or not.

Up to a point, this can be mitigated by increasing the shutter speed during image capturing, however this will also decrease the amount of light that reaches the camera, making the images darker overall, so depending on the lighting conditions, this may not be a viable option.

Using a different model that is trained to identify streaks instead of circles or even training the same model to segment streaks into a different class could be a solution to this problem.

Doing this could have some additional benefits. For example, by finding the minimum angled rectangle that contains the streak we could calculate the ratio of the width and height of the rectangle to estimate the speed of the droplet, since the shutter speed is a known parameter and the longer side of the rectangle corresponds to the distance the droplet traveled in the time it took to expose the image. Since the speed of the droplet is another interesting parameter to measure, this could be a useful addition to the model. One might even be able to deliberately lower the shutter speed to make the droplets appear as streaks for the purpose of measuring the distribution of the droplet velocity.

Due to time constraints, this idea couldn't be explored further in this thesis, but could be an interesting topic for future work.

Droplets that are too small are not reliably identified by the model, since the smaller the droplet, the less distinguishable the different parts of the droplet become. This is an inherent problem of techniques using imaging as a measurement method, since there is always a limit to the scale at which objects can be resolved.

Up to a point this can still be mitigated by increasing the resolution of the camera or the magnification of the lens, but this will also increase the cost of the setup and will only be possible to a certain extent.

Since this problem isn't something that is specific to the algorithm, but rather a limitation of the imaging technique itself, there isn't much that can be done to improve the model in this regard. For the actual usage of the technique, this means human oversight is still necessary for deciding if the approach is applicable to specific set of captured data.

In focus criteria may be unrepresentative of actual in focus droplets. Instead of an error observed in the measuring process, this is a potential problem with the method itself. Which droplets are considered in focus and should be included in the measurement is a key aspect of the process, since data has to be labeled accordingly. Structures with extremely sharp corners are exceedingly rare in the data captured by the experimental setup, which is why Kappl[14] argues that structures with a dark border and a bright center should be considered sufficiently in focus.

The reasoning behind this is that light that passes through the outer areas of the droplet is scattered more strongly, resulting in less light reaching the camera, while light that passes through the center of the droplet passes through straight, with beams near the center even being focused slightly.

The success of the method as a measurement technique depends heavily on the veracity of this assumption. However, if in the future, other criterions are found to be more suitable to differentiate between in focus and out of focus droplets, the method could theoretically be adapted to use those instead.

6.3. Experimental Setup

Athough the detection techniques described in 6 are in principle adaptable to similar data, the image data used in this thesis is captured using a specific setup.

The setup uses the *Fastcam Mini UX* in combination with a *Olympus LMPlan IR 50X / 0.55* microscope objective lens as well as a seperate focussing lens to capture images of the droplets. Together with a parallel light source, this setup forms a microscope with a large magnification factor, which is necessary to capture the droplets in sufficient detail considering their expected size of 1 to 30 µm [14].

The SAW device together with a liquid reservoir is placed next to the light source and the camera so that the droplet droplet stream is illuminated from the side. An image of the setup is shown in Figure 6.3.

Images are taken at a resolution of 1280 px × 1024 px and a frame rate of 50 Hz. The framerate is kept low to enable capturing data over a longer period of time without saturating the memory of the camera, since the droplets are not constantly in the field of view of the microscope.

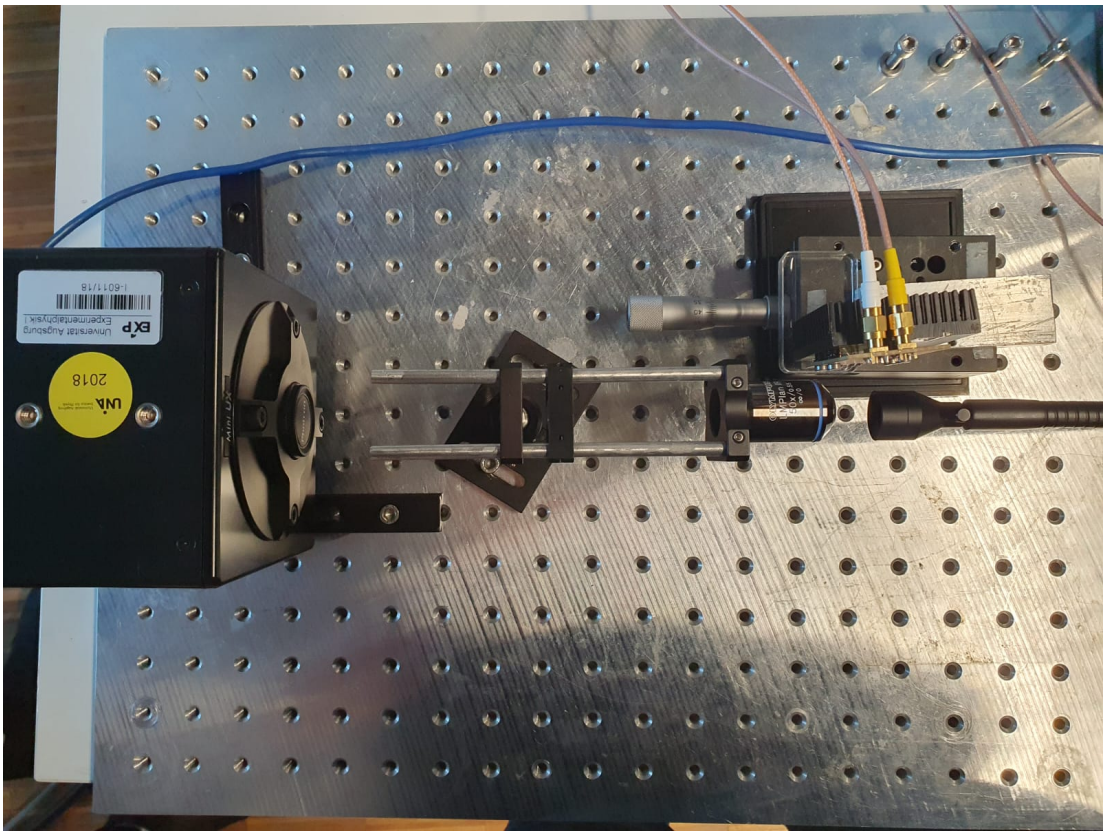


Figure 6.3.: Image of the measurement setup. From left to right the high speed camera, the focusing lens and objective lens, the SAW device and the liquid reservoir as well as the light source are visible.

7. Experiments and Results

Several techniques are explored during model training in the hopes of improving the detection and measurement performance as well as how the model generalizes. This section evaluates the results of these experiments and discusses their effectiveness.

Before discussing specific experiments, the general training and evaluation environment will be described.

Training and Evaluation Setup All learning is done using the *pytorch*[45] framework.

Where applicable, techniques were tested on the *Cityscapes*[26] dataset, which contains 3000 images with pixel-level annotations of 30 different classes, 19 of which were used during training and evaluation. Since the annotations for the Cityscapes test set are not public, models trained on this dataset are only evaluated based on their performance on the validation data.

The training data for the target dataset contains image data recorded using the setup described in section 6.3 with three different sets of imaging and droplet generation parameters. The data is annotated with three classes, namely *droplet inside*, *droplet outside* and *background*. Annotation was performed with the Software *CVAT*[46].

The test set for the vapour data contains 35 images from different experiments, which were not used during training. The images in the test set are carefully selected to cover a wide range of droplet sizes and clarity as well as different overall lighting conditions. It also deliberately includes images with a lot of artifacts and noise that are observed to commonly cause problems for the model (see 6.2) as well as images with no droplets at all.

The loss function used for training is the *cross entropy loss* which is defined as:

$$\ell_{i,j} = -\log \hat{y}_{i,j,y_{i,j}} \quad ,$$

where $\hat{y} \in \mathbb{R}^{M \times N \times C}$ are the predicted class probabilities, $y \in \mathbb{N}^{N \times M}$ are the ground truth class labels and C is the number of classes. The loss is averaged over all pixels [47].

$$\mathcal{L}(y, \hat{y}) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \ell_{i,j}$$

The class probabilities are calculated from the logit outputs o of the model using the *softmax* function:

$$\hat{y}_{i,j,c} = \frac{\exp(o_{i,j,c})}{\sum_{k=1}^{C} \exp(o_{i,j,k})} \quad .$$

The main metric used for evaluating the performance of the model is the *mean intersection over union* (mIoU, see section 3.2), which is calculated by accumulating the intersection and union for each class over the whole data set, calculating the class-wise IoU and then averaging the results over the number of classes.

For training on the vapour dataset, additional metrics can be calculated which may better reflect the models performance in its intended application. To do that, the measurement algorithm is applied to both the predicted and ground truth segmentation masks. Predicted droplets are considered to be correctly detected if their center is within the area of a ground truth droplet, which is precise enough because the droplets are generally very sparsely distributed. The additional metrics calculated from this information are

- *Precision*: The fraction of detected droplets (P) that are actually present in the image (TP).

$$\frac{\mathrm{TP}}{\mathrm{P}}$$

- *Recall*: The fraction of actual droplets (= T) that are detected.

$$\frac{\mathrm{TP}}{\mathrm{T}}$$

- *Relative Mean Radius Error* (RMRE): The relative difference between the predicted and ground truth average droplet radius (ADR).

$$\frac{\mathrm{ADR}_{\mathrm{pred}}}{\mathrm{ADR}_{\mathrm{truth}}}$$

  This is calculated considering only the droplets that are detected in both the ground truth and the prediction ($\mathrm{RMRE}_{\mathrm{c}}$) as well as all droplets that are present in the ground truth ($\mathrm{RMRE}_{\mathrm{t}}$).

Unless stated otherwise, all metrics are calculated for the test set of the vapour dataset.

All training uses *early stopping* to determine the end of training. The model is trained for as long as the improvement in the evaluation metric (mIoU) is significant. Improvement is considered stagnant if the metric doesn't surpass its best value by at least a certain *minimum delta* within a certain number of *patience* epochs. Patience and minimum delta are generally set to 30 and 1×10^{-4} respectively.

Additionally, the model is trained using a *learning rate scheduler* which reduces the learning rate by a certain factor if the improvement in the evaluation metric is not significant for a certain number of epochs. The idea behind this is that initially a higher learning rate is beneficial to avoid getting stuck in local minima. However, in later stages of training the model may already be close to a global minimum and a higher learning rate may cause it to overshoot the minimum, so lowering the learning rate helps to avoid this. The scheduling process used in this

thesis is called *ReduceLROnPlateau*[48]. There are other approaches to scheduling, however the ReduceLROnPlateau scheduler achieves good results in this case and is easy to implement without knowing the exact number of epochs that will be needed for training. The minimum delta is set to 1×10^{-4} for all experiments, with the patience varying between 5 and 30 epochs depending on the dataset and training approach.

Unless otherwise stated the model architecture is always a U-Net with a ResNet34D pretrained on ImageNet data as its encoder.

7.1. Oversampling to overcome class imbalance

Class imbalance is a common problem in many machine learning tasks. This is especially true in the case of the underrepresented classes being the most relevant ones, as is the case for the droplets in the vapour dataset. The background label makes up 99.914 % of all labeled pixels in the dataset, while the droplet border and droplet inside labels only make up 0.063 % and 0.023 % respectively.

Initially, this extreme unbalance made it impossible to train a model that could detect droplets and the mIoU would stagnate around 0.33, which is the accuracy of a model that always predicts the background class.

One way to combat this imbalance is weighting the loss function of each class inversely proportional to the number of samples in the dataset that belong to that class, to prioritize predicting the underrepresented classes correctly during training. However, perhaps due to the unusually large imbalance, this method did not improve the performance of the model. Using smaller weights than the inverse distributions didn't achieve better results either.

The method that is applied in the thesis to overcome this problem is a form of *oversampling*[49], where the occurence of the underrepresented classes is artificially increased by showing the model samples that do contain these classes more often. Although there are more sophisticated approaches to oversampling [50], the solution used in this thesis is a simple one that is easy to implement, but has proven to be effective.

Since droplets are distributed very sparsely in the data, when splitting the images into smaller samples, a lot of the splits will not contain any droplets. Instead of including these empty regions in the training set, they are discarded when the split images are created. Doing this changes the label distribution to 99.84 % background, 0.116 % droplet border and 0.044 % droplet inside, which doesn't appear to be much better than the original distribution, but roughly doubles the occurrence rate of the droplet classes. This alone is enough to allow the model to learn to detect droplets with good accuracy.

To take this further, instead of splitting the image dimensions by two they could also be split by a larger factor, which would increase the relative density of droplets in the samples even more.

The results of this are shown in each experiment separately, since the same split factor might not work well for all approaches. No split factors larger than 4 were tested, since images would become very small and split droplets into different samples more often, which might also be detrimental.

No further comparative studies were conducted to determine the optimal split factor or the best way to apply oversampling, since the results of this simple approach were already good enough to be used in the thesis. Oversampling is used in all other experiments, since training on the vapour dataset is impossible without it.

7.2. Mean Teacher for general improvement

As described in 5.2, the mean teacher approach should enable us to utilize unlabeled data to improve the performance of our model. To verify the effectiveness of this approach, the method is first tested on the Cityscapes dataset before it is applied to the vapour dataset.

In this experiment, the only concern is to improve the general performance of the model measured by the metrics described above. Further experiments will verify the effectiveness of the method for improving the generalization capabilities of the model on the vapour dataset.

Verification on Cityscapes For the Cityscapes dataset training is done on images downsampled to a 256 px × 512 px resolution. The initial learning rate is set to 1×10^{-3} and the batch size to 16 containing 4 labeled and 12 unlabeled samples. The patience for the scheduler is set to 5 epochs and the reduction factor is set to 0.1. The weight of the consistency loss is ramped up to its maximum value of 1.0 over the first 15 epochs using a sigmoid ramp. Consistency loss is calculated using the *cross entropy loss* between the predictions of the student and the hard predictions of the teacher (meaning after applying *argmax*). The teacher is updated using an EMA with a decay α of 0.996.

For the baseline model, the best results are achieved by using an initial learning rate of 1×10^{-4} with a batch size of 16, a scheduler patience of 20 epochs and a reduction factor of 0.5.

The baseline model is trained on 100 labeled samples from the training set, while the mean teacher model is trained on the same 100 labeled samples plus all other samples from the training set as unlabeled samples. A training epoch when using MT is considered to be finished when the model has seen all unlabeled samples once, with the labeled samples being seen as many times as necessary to fill the batch size. This means epochs for MT training contain considerably more samples than epochs for baseline training, which is why the patience value for baseline training is higher.

For the baseline model, the augmentation pipeline consists of a random 224 × 224 crop, a random translation, scaling and rotation with a probability of 0.5, a RGB-shift with a probability of 0.5 and a random variation of the image brightness and contrast with a probability of 0.5.

The augmentation pipeline for the MT training employs a different set of augmentations, since strong asymmetrical augmentations between teacher and student are important for the MT approach to properly function. For this, the augmentation pipeline is inspired by Scherer et al.[51]. It contains the same shift, scale, rotation and crop operations as the baseline augmentations, but also applies a strong dropout of 0.5 to the student model, as well as using a *CowMask* technique to compose new samples by combining two images as well as their teacher predictions. For this technique a mask such as the one shown in Figure 7.1 is generated and applied to an image and its pseudo label. The same is done for a second image and its pseudo label with the inverted mask and both are combined.

Finally, the images are normalized using the mean and standard deviation of the ImageNet dataset in both baseline and MT training.

Figure 7.1.: Example for a cow mask used in the training process. The white areas mark regions that use the first image, while the black areas use the second image. From [51]

Results on Cityscapes With the MT approach, the model is able to achieve a mean IoU of 0.426 on the validation set, which is a significant improvement over the baseline model with a mean IoU of 0.349.

However, this improvement is only achievable within a very small space of hyperparameters, as all training runs with different hyperparameters than described above yielded worse results than even the baseline model.

This deterioration in performance can likely be attributed to very poor pseudo label quality. As Scherer et al.[51] discovered in their analysis of *pseudo label filtering* techniques, pseudo labels with poor quality have a detrimental effect on final model performance. Since the setup used in this thesis doesn't achieve very high performance on the Cityscapes dataset as is, even when using all labeled samples (mIoU of 0.475), it stands to reason that label quality may hurt the training process if hyperparameters are suboptimal.

Nonetheless, the results of this experiment show that the MT approach is able to improve the performance of the model. With the supervised performance on the target dataset being

significantly better than on the Cityscapes dataset, it is likely that the MT approach will be able to improve the performance of the model on the vapour dataset as well.

For a more comprehensive overview of training results, see Table 7.1.

MT	#lbls	lr	bsize	lrsf	lrsp	Dropout	α	Loss	mIoU
no	all	0.0001	16	0.5	10	-	-	0.216	0.475
yes	100	0.001	16	0.1	5	0.5	0.996	0.644	**0.426**
yes	100	0.001	16	0.1	5	0.4	0.996	0.559	0.412
no	100	0.0001	16	0.5	30	0.2	-	0.444	0.349
no	100	0.0001	16	0.5	30	-	-	0.469	0.331

Table 7.1.: The best training results of the Mean Teacher approach on the Cityscapes dataset, as well as the supervised baseline performance along with important hyperparameters. For abbreviations refer to Table A.1.

Setup on vapour dataset The training setup for the vapour dataset is very similar to the Cityscapes setup, with the images being split in half along the vertical and horizontal axis and these 4 segments being used as separate samples. The decision to split the images instead of resizing them was made because the relevant objects in the images are already quite small, so resizing them might make it impossible to detect some of them. This leaves us with 640 px × 512 px images, which necessitated lowering the batch size to 12 for baseline training and 8 (2 labeled, 6 unlabeled) for MT training. Learning rate was generally found to be optimal around 1×10^{-3}.

Another difference in the setup is the augmentation pipeline, which is similar to the Cityscapes pipeline, but replaces the RGB-shift with gaussian noise, since the vapour dataset contains greyscale images. The gaussian noise might also be able to mimic some of the noise the camera produces, so it seems like an adequate replacement.

Furthermore, we also explore using *CutMix*[52] for mixing pseudo labels instead of the CowMask technique used for Cityscapes. CutMix cuts a rectangular region of the image instead of the varying sizes of blobs in the CowMask technique. The reason this may be more suitable to the vapour dataset is that the objects in the images are generally quite small, so the CowMask technique, which masks smaller regions might not be effective in combining relevant features from the two images. On Cityscapes, CutMix is not explored since Scherer et al.[51] already concluded that the CowMask technique is superior.

Results on vapour dataset The results of the training on the vapour dataset are shown in Table 7.2. Unfortunately, it does not appear that the MT approach is able to outright improve the performance of the model on the vapour dataset regarding the mIoU.

On average, the the MT approach produces a worse mIoU than the supervised training on the test dataset, although not by a large margin. It is worth noting that the MT approach does consistently achieve a higher mIoU than the supervised approach on the validation dataset during training, but this doesn't seem to translate to the test dataset. However, there are some

metrics where semi supervised learning does appear to be beneficial, as it achieves the overall best recall and RMRE (CowMask sf=4 and CutMix sf=2 in Table 7.2).

There are several conjectures as to why the MT approach doesn't produce the desired results.

As mentioned in section 7.1, images that do not contain any droplets after splitting have to be filtered out in the supervised training setup to achieve any meaningful results. Meanwhile, unlabeled samples that don't contain any droplets are used as normal in the MT setup, which might degrade the performance of the model.

Another possible reason could be that the number of unlabeled samples is too small, which might not be enough to improve the performance of the model in the same way as it does on the Cityscapes dataset.

However, since semi supervised learning is able to improve the performance of the model in several key metrics, it is still beneficial to the overall measurement process. If the aforementioned flaws are addressed, further improvement might also be possible.

As for the comparison between the CowMask and CutMix techniques, it appears that CutMix outperforms CowMask for a split factor of 2, but becomes significantly worse for a split factor of 4 (mIou 0.76 vs 0.68). The reason for this degradation could be that the borders of the mask are more likely to intersect droplets, which take up a larger portion of the image for a higher split factor. These kinds of dissected droplets could be too difficult to detect for the model, which would explain the worse performance.

Generally a split factor of 4 seems to produce better results when using the MT + CowMask, while not having a large impact on supervised training.

MT	sf	Loss	mIoU	recall	precision	$RMRE_c$	$RMRE_t$
no	2	0.00090	**0.77832**	0.85333	**0.98462**	-0.11194	-0.11199
	4	0.00074	0.77217	0.94667	0.97260	-0.10405	-0.10232
yes (CowMask)	2	0.00114	0.73377	0.84667	0.91367	-0.16528	-0.15719
	4	0.00100	0.75464	**0.99333**	0.87135	**-0.06566**	-0.05721
yes (CutMix)	2	**0.00080**	0.75884	**0.99333**	0.88690	-0.07457	**-0.05647**
	4	0.00152	0.68242	1.04000	0.74641	-0.13009	-0.06471

Table 7.2.: Results for the Mean Teacher approach on the vapour dataset. MT indicates wether the MT approach was used and which mixing technique was employed. The table only contains results that weren't pretrained. Bold numbers indicate the overall best performance for the metric. For abbreviations refer to Table A.1.

7.3. Influence of Pretraining on model performance

As mentioned in section 5.1, besides using a pretrained encoder, the thesis also explores pretraining the whole model on the Cityscapes dataset as a starting point for training on the droplet dataset.

Because the model has already learned some useful features, this could potentially speed up the training process and improve the performance of the model.

For this experiment, the weights of the best performing model on Cityscapes (mIoU 0.475) is loaded when starting training, except for the last layer, whose number of output channels is changed to match the number of classes in the droplet dataset.

Results are displayed in Table 7.3. The comparison is made with the results in Table 7.2.

Pretraining the model on Cityscapes seems to have the largest impact on the performance of supervised training with a split factor of 2, while only improving the results of semi supervised training by a small margin.

For supervised training with a split factor of 4 as well as for MT + CutMix, the performance is even worse than the non pretrained model.

Since the majority of the computational complexity of the model comes from the encoder, it is perhaps not surprising that pretraining on Cityscapes doesn't improve the performance a lot. Even if the whole model isn't pretrained, the encoder still uses ImageNet weights, which are already quite good.

During Cityscapes pretraining, the decoder might even learn features that are not useful or even detrimental for training on the droplet dataset. However, if this is the case, it is unclear as to why this only happens for supervised training with a split factor of 4 and MT + CutMix.

MT	sf	Loss	mIoU	recall	precision	$RMRE_c$	$RMRE_t$
no	2	**0.00068**	**0.78370**	0.88667	**0.97080**	**-0.09096**	-0.08740
	4	0.00092	0.75218	0.83333	0.96154	-0.11241	-0.11096
yes (CowMask)	2	0.00082	0.75358	**0.98667**	0.91358	-0.09818	**-0.08481**
	4	0.00083	0.75835	0.94667	0.93421	-0.10643	-0.11527
yes (CutMix)	2	0.01043	0.65301	0.90000	0.95745	-0.10438	-0.10716

Table 7.3.: Results of the pretraining experiment. All training is done by starting training with weights pretrained on Cityscapes. For comparison with non pretrained models refer to Table 7.2.

7.4. Mean Teacher for generalization

Another hope for the MT approach is that it will be able to improve the generalization capabilities of the model. Since different imaging parameters or different SAW devices will produce slightly different images, being able to adapt the model to new parameters without having to label new data would be a huge advantage.

To explore this, two subsets (S1 and S2) of the vapour datasets with different parameters are used to create two new datasets (DS1 and DS2). DS1 contains the labeled samples from S1 as well as samples from S2 with the labels removed. DS2 is built in the same way, but with S2 as the labeled dataset and S1 as the unlabeled dataset.

The model is then trained on one of the two datasets (DS1 or DS2) and evaluated on the validation set of the subset that was not included as labeled data (S2 or S1). Training is done once with only the labeled samples and once with the unlabeled samples using MT. This is repeated for both datasets, so that one model is trained on either dataset.

To avoid adapting the model purely to one subset, the original test set is used for validation during training.

A similar number of labeled (~ 30) and unlabeled (~ 150) samples are used for each dataset, so that the model is trained on the same number of samples in total.

If including the unlabeled samples from S2 helps the model improve on the test set of S2 compared to only having seen the labeled samples from S1 (and vice versa), it would indicate that the MT method does indeed help the model generalize without new labeled data.

Training parameters are the same as described in section 7.2.

Results The evaluation results are shown in table 7.4. Unfortunately, the results aren't as conclusive as hoped. While using the unlabeled samples from S1 did help the model improve its mIoU on S1 (results for DS2 in Table 7.4), the opposite isn't true, with the supervised training on DS1 performing better than the MT training, although not as significantly. Additionally, in both cases the supervised model performs better in all measurement specific metrics except for the recall.

The reason for this could be that the data from S1 is generally more difficult to segment than the data from S2. This could mean that including the easier data from S2 as unlabeled data doesn't really help the model learn new information, while including S1 as unlabeled data (in the case of DS2) actually introduces new concepts to the model. As evidence for the higher difficulty of S1, we can point to the mIoU on the Validation set during training being higher for DS1 than for DS2, where S1 samples were labeled.

To further investigate this in the future, we could construct another pair of datasets that achieve more similar scores on the original test set which could mean they are more similar in difficulty.

As for now, evidence for the MT method's ability to help generalization without new labeled data is inconclusive.

This doesn't discount the value of the MT method for improving the model in general, but could mean that including new data in the training means having to label a small fraction of it to

enable the model to grasp its basic structure, which is still a huge advantage over having to annotate all of it.

Set	MT	Validation mIoU	Test mIoU	recall	precision	$RMRE_c$	$RMRE_t$
DS1	no	0.76072	**0.71120**	0.70330	**0.94118**	**-0.05730**	**-0.05524**
	yes	**0.76570**	0.70605	**0.74725**	0.89474	-0.11600	-0.14195
DS2	no	0.72521	0.67748	0.62963	**0.94444**	**-0.18807**	**-0.17170**
	yes	**0.72534**	**0.71552**	**0.66667**	0.85714	-0.26960	-0.25511

Table 7.4.: Training Results for the generalization experiments. The Set Column indicates which dataset was used for training and evaluation and refers to the sets described in section 7.4. MT indicates whether the MT method was used. Bold values indicate the better result for each metric.

7.5. Binary vs. multi-class segmentation

In section 6.1 it's stated that training the model to segment the image into three different classes helps to filter out common model errors in later stages of the measurement algorithm. However, introducing a third class also makes the problem more difficult to learn. It could be that training the model on only two classes (background and in-focus droplet) would enable the model to avoid these errors in the first place, since it could learn the criterion for a droplet being in focus more easily.

To test this, the model is trained on the vapour dataset, but with all different droplet labels combined into a single class. The model is then evaluated on the test set that underwent the same transformation.

During the evaluation process, the filtering step performed by the droplet measurement algorithm is skipped, since the inside labels don't exist anymore.

Training parameters are the same as described in section 7.2. The split factor for training is 2.

Results are shown in table 7.5.

Surprisingly, when training without MT, binary segmentation doesn't seem to perform worse than multi-class segmentation. Since the mIoU is is averaged over two classes instead of three and the IoU for the background class is close to 1, a slightly higher mIoU is expected, but this level of difference is still significant. Supervised binary segmentation also performs reasonably in the other metrics.

The story seems to be different when looking at the results of semi supervised trainings. Instead of improving the performance, the mIoU drops significantly in the case of binary segmentation. This degradation is most heavily reflected in the precision metric, which drops from 0.89 to 0.39, while recall stays roughly the same. This makes semi supervised learning unusable for binary segmentation.

Since better results for recall and precision have been achieved by supervised and semi supervised multi-class training, approaching the problem as a multi-class segmentation task seems to be the better choice and confirms the assumption that additional information for the post processing step is more valuable than a simpler problem.

#Classes	MT	Loss	mIoU	recall	precision	$RMRE_c$	$RMRE_t$
2	no	0.000607	0.87815	0.88889	0.88889	-0.08282	-0.04464
	yes	0.001975	0.73203	0.89506	0.39189	-0.18260	0.12074
3	no	0.000914	0.75570	0.80666	0.96800	-0.14852	-0.14507
	yes	0.000798	0.75884	0.99333	0.88690	-0.07457	-0.05646

Table 7.5.: Comparison of binary vs. multi-class segmentation. The multi-class examples represent the top results in the same category (MT/no MT). Split factor for training is 2.

7.6. Training with reduced layers

The architecture of the model described in chapter 4 has five distinct downsampling and upsampling stages in its encoder and decoder paths respectively. The stated purpose of the lower layers is to be able to utilize more contextual information since neurons in lower layers have a larger *receptive field* than neurons in higher layers.

However, for our specific problem, objects are generally quite small and the problem itself isn't very complex, so the contextual information provided by the lower layers might not be as useful as it is for larger objects. Additionally, including the lower layers may not only be unnecessary, but also detrimental to the performance of the model, since it increases the number of parameters and with that the computational complexity, which makes it harder to train. This also plays a role in the demand for computational resources during training and inference, since even only omitting the deepest layers reduces the trainable parameter count from 24.5 M to 9.1 M.

To explore this hypothesis, the model is trained on the vapour dataset with the encoder and decoder paths reduced to only four/three stages each (final encoder stride is 16/8 instead of 32). The model is then evaluated as normal.

Because of time constraints, pretraining the reduced model on the Cityscapes dataset was not attempted, however since the Cityscapes dataset is much more complex than our own, it is unlikely the model would have achieved good results on it in the first place.

The experiment compares the performance with and without semi supervised learning, with a split factor of 2.

Training parameters are the same as described in section 7.2.

Results of the experiment are shown in Table 7.6. Comparison is done with the results of the complete model seen in Table 7.2.

While the complete model still takes the top score when it comes to the mIoU metric (0.784 for supervised training with sf = 2, see Table 7.2), the top result for the reduced model only lags behind slightly with 0.775 for the same training parameters (4 stages supervise sf=2, see Table 7.6).

For the semi-supervised training, the reduced model consistently outperforms the complete model at the mIoU metric, with the only exception being the case where the 4-stage model is trained with sf = 4.

What is most surprising is that this is true not only for the 4-stage model, but also for the 3-stage model, which only contains 1.7 M trainable parameters. While there is a slight drop in performance for 3 stages compared to 4, considering the difference in size between the two models, this is quite impressive.

Unfortunately, it seems the reduced models don't benefit from the MT approach for the RMRE metrics, as was observed in some cases for the complete model.

The best overall result is achieved by the 4-stage model during supervised training with sf = 2, which achieves a mIoU of 0.775 as well as precision and recall values of over 0.95, which is the highest combined accuracy observed.

Since the precision should reflect the actual performance of the model in the application very well, using this configuration for inference seems to be a good choice.

#Stgs./Param.	MT	sf	Loss	mIoU	recall	precision	$RMRE_c$	$RMRE_t$
3 / 1.7 M	no	2	0.00103	0.75167	0.84667	0.97692	-0.14081	-0.13211
		4	0.00091	0.76490	0.86667	0.97744	-0.12805	-0.13039
	yes	2	**0.00068**	0.76703	0.85333	**0.99225**	**-0.09556**	-0.10277
		4	0.00123	0.76038	0.91333	0.93197	-0.10090	-0.10861
4 / 9.1 M	no	2	0.00077	**0.77552**	0.95333	0.97279	-0.10466	**-0.09549**
		4	0.00088	0.76513	0.86000	0.94161	-0.11796	-0.10821
	yes	2	0.00084	0.76889	0.88000	0.97059	-0.11428	-0.11925
		4	0.00123	0.73325	**0.96000**	0.87805	-0.10355	-0.10079

Table 7.6.: Evaluation results for models trained with a reduced number of encoder/decoder stages. CowMask is used as the mixing method. For comparison to full scale results refer to Table 7.2. For abbreviations refer to Table A.1.

7.7. Employing a classical algorithm

While the main focus of this thesis is on the use of machine learning to detect droplets, it also begs the question whether a classical algorithm could be used to achieve similar results without the drawbacks that come with a deep learning approach.

The idea is to use the Hough transform to detect the circular droplets in the image. The Hough transform is used on the image after it has been preprocessed the same way as in the actual method and applying a gauss filter to smooth out the image [53].

We can compute the same measurement accuracy metrics as for the actual method on the test set, the results of which are shown in Table 7.7.

Looking at example images, the Hough transform is indeed able to recognize some of the droplets in the images, however at several points, even very clear droplets are not detected, even though the transform is able to identify very similar instances. Another problem is that the transform also detects a lot of circular structures that do not conform to the in-focus criteria. This is to be expected, as the transform itself does not have a way to distinguish between a droplet that is in focus and one that is not.

To help with this, we can again make use of the high brightness contrast of in focus droplets, by calculating the *Michelson contrast* for only the pixels that are part of the circle and discarding any that are below a certain threshold. The higher this threshold is, the more likely it is to also discard in-focus droplets so it is important to strike a balance when choosing this parameter. This also only helps with the precision of the method, as the recall is still limited by the fact that the transform is not able to detect all droplets.

Still, the Hough transform method is not able to surpass a recall of 0.7, in which case the precision drops to 0.17, which is not good enough to be used in practice. The RMRE is also not very good, but since the control in this test is still performed by applying the original algorithm to the test ground truth, it is best not to pay too much attention to this metric.

contrast threshold	recall	precision	$RMRE_c$	$RMRE_t$
-	0.68667	0.17311	-0.19577	-0.31010
0.5	0.46667	0.66667	-0.26924	-0.30075
0.4	0.55333	0.57639	-0.22304	-0.27240
0.3	0.58667	0.43564	-0.21101	-0.28273

Table 7.7.: The results for applying the Hough Transform method to the test data. The contrast threshold is the number the Michelson contrast in a detected circle must surpass to be considered a droplet.

7.8. Summary

To summarize the results of the experiments, the best overall results are achieved by supervised training a model with the number of stages reduced to four on multi-class labels (see section 7.6).

While semi supervised training does not demonstrate a clear advantage over supervised training when it comes to overall accuracy of the model or the ability to generalize to new data, it does seem to improve the performance of the model in the case of the RMRE metric and shows promise for future improvements if key issues can be resolved (see sections 7.2, 7.4).

Binary labeling does provide a boost in mIoU for supervised training, but this is not the case for semi supervised training, where the model performs worse than compared to using multi-class labels (see section 7.5). Multi-class training also beats binary training when it comes to droplet detection accuracy.

Almost all models predict the droplet radius as slightly too small. Since this is seems to be a general trend, this error could perhaps be addressed on the post-processing side of the algorithm, for example by raising the threshold for including pixel distances in the droplet radius.

Applying a classical image processing algorithm to the problem is able to achieve some results, but ultimately fails to perform well because of its inability to differentiate between in-focus and out-of-focus droplets in a nuanced way as well ad detect droplets that don't have a clear boundary.

When it comes to applying the model to real data, the results can be significantly improved by supplying the model with high quality data, as a lot of the more difficult examples that were included in the test set are produced from highly inconsistent lighting conditions in the images. By slightly curating which data is used for measurement, the algorithm can be used to measure droplet size in a reliable way. Although doing this means human interference can not be eliminated completely from the process, the application still results in a significant time saving compared to manual measurement.

To demonstrate the effectiveness of the technique, some of the instance of the techniques's performance on example image data can be seen in the appendix in Figure B.1.

8. Conclusion

This thesis explored the use of deep learning for the segmentation of droplets in images of fluids vaporized using surface acoustic waves. The overall goal was to develop a model that can be used to detect the presence of droplets in such images and to employ it in a usable system that measures the size of the droplets.

It gave an overview to the physical background of the atomization techniques and mentioned some of the limitations of other measurement techniques. Furthermore the basics of neural networks and deep learning as well as the concepts and structure of the employed architectures were explained.

An algorithm for the measurement of droplets from segmentation mask was developed and its shortcomings and strengths as well as possible improvements were discussed.

The thesis also explored the use of transfer learning and self-supervised learning to improve the performance of the model in its application. While some of the techniques weren't as successful as expected, the results of the experiments still showed promise and possible future improvements were suggested.

As it stands, the thesis achieved its goal of building a tool that could find use in current research and development projects, while also providing a basis for further investigation and refinement.

The use of SAW for the application of atomization is a promising technique that has the potential to be used in a variety of applications and the author hopes that the work presented in this thesis will contribute to the development of this technology.

9. Bibliography

[1] C. Alippi, A. Ferrero, and V. Piuri, "Artificial Intelligence for Instruments and Measurement Applications", *Instrumentation & Measurement Magazine, IEEE* **1**, 9–17 (1998).

[2] Z. Ballard, C. Brown, A. M. Madni, and A. Ozcan, "Machine Learning and Computation-Enabled Intelligent Sensor Design", *Nature Machine Intelligence* **3**, 556–565 (2021), https://www.nature.com/articles/s42256-021-00360-9 (visited on 02/01/2023).

[3] D. Mandal and S. Banerjee, "Surface Acoustic Wave (SAW) Sensors: Physics, Materials, and Applications", *Sensors* **22**, 820 (2022), https://www.mdpi.com/1424-8220/22/3/820 (visited on 02/08/2023).

[4] J.-F. Li, *Lead-Free Piezoelectric Materials* (Wiley-VCH, Weinheim, 2021).

[5] M. Kurosawa, T. Watanabe, A. Futami, and T. Higuchi, "Surface Acoustic Wave Atomizer", *Sensors & Actuators: A. Physical* **1–2**, 69–74 (1995), https://www.infona.pl//resource/bwmeta1.element.elsevier-58e6f205-d865-39d8-8bd8-bfebfab7fa9f (visited on 02/09/2023).

[6] A. Qi, J. R. Friend, L. Y. Yeo, D. A. V. Morton, M. P. McIntosh, and L. Spiccia, "Miniature Inhalation Therapy Platform Using Surface Acoustic Wave Microfluidic Atomization", *Lab on a Chip* **9**, 2184 (2009), http://xlink.rsc.org/?DOI=b903575c (visited on 02/04/2023).

[7] N. Murochi, M. Sugimoto, Y. Matsui, and J. Kondoh, "Deposition of Thin Film Using a Surface Acoustic Wave Device", *Japanese Journal of Applied Physics* **46**, 4754 (2007), https://iopscience.iop.org/article/10.1143/JJAP.46.4754/meta (visited on 02/24/2023).

[8] M. Alvarez, J. Friend, and L. Y. Yeo, "Rapid Generation of Protein Aerosols and Nanoparticles via Surface Acoustic Wave Atomization", *Nanotechnology* **19**, 455103 (2008), https://dx.doi.org/10.1088/0957-4484/19/45/455103 (visited on 02/24/2023).

[9] A. Winkler, S. M. Harazim, S. B. Menzel, and H. Schmidt, "SAW-based Fluid Atomization Using Mass-Producible Chip Devices", *Lab on a Chip* **15**, 3793–3799 (2015), http://xlink.rsc.org/?DOI=C5LC00756A (visited on 02/08/2023).

[10] D. J. Collins, O. Manor, A. Winkler, H. Schmidt, J. R. Friend, and L. Y. Yeo, "Atomization off Thin Water Films Generated by High-Frequency Substrate Wave Vibrations", *Physical Review E* **86**, 056312 (2012), https://link.aps.org/doi/10.1103/PhysRevE.86.056312 (visited on 02/10/2023).

[11] Q.-Y. Huang, Y. Le, H. Hu, Z.-j. Wan, J. Ning, and J.-L. Han, "Experimental Research on Surface Acoustic Wave Microfluidic Atomization for Drug Delivery", *Scientific Reports* **12**, 7930 (2022), https://www.nature.com/articles/s41598-022-11132-9 (visited on 02/10/2023).

[12] A. Qi, L. Y. Yeo, and J. R. Friend, "Interfacial Destabilization and Atomization Driven by Surface Acoustic Waves", *Physics of Fluids* **20**, 074103 (2008), https://aip.scitation.org/doi/full/10.1063/1.2953537 (visited on 02/10/2023).

[13] Aisha Qi, J. R. Friend, and L. Y. Yeo, "Investigation of SAW Atomization", in 2009 IEEE International Ultrasonics Symposium (Sept. 2009), pp. 787–790, http://ieeexplore.ieee.org/document/5441556/ (visited on 02/11/2023).

[14] L. Kappl, *Akustisch induzierte Vernebelung verschiedener Flüssigkeiten*, 2022.

[15] N. R. Labiris and M. B. Dolovich, "Pulmonary Drug Delivery. Part I: Physiological Factors Affecting Therapeutic Effectiveness of Aerosolized Medications", *British Journal of Clinical Pharmacology* **56**, 588–599 (2003).

[16] R. M. Drake and J. E. Gordon, "Mie Scattering", *American Journal of Physics* **53**, 955–962 (1985), https://aapt.scitation.org/doi/abs/10.1119/1.14011 (visited on 02/24/2023).

[17] T. Wriedt, "Mie Theory: A Review", in *The Mie Theory: Basics and Applications*, edited by W. Hergert and T. Wriedt, Springer Series in Optical Sciences (Springer, Berlin, Heidelberg, 2012), pp. 53–71, https://doi.org/10.1007/978-3-642-28738-1_2 (visited on 02/24/2023).

[18] W. Holl and G. P. G. Morawietz, "PARTICLE DEPOSITION VELOCITIES",

[19] R. Sijs, S. Kooij, H. J. Holterman, J. van de Zande, and D. Bonn, "Drop Size Measurement Techniques for Sprays: Comparison of Image Analysis, Phase Doppler Particle Analysis, and Laser Diffraction", *AIP Advances* **11**, 015315 (2021), http://aip.scitation.org/doi/10.1063/5.0018667 (visited on 02/13/2023).

[20] B. Macukow, "Neural Networks – State of Art, Brief History, Basic Models and Architecture", in Computer Information Systems and Industrial Management, edited by K. Saeed and W. Homenda, Lecture Notes in Computer Science (2016), pp. 3–14.

[21] m0nhawk, *Answer to "TikZ: Diagram of a Perceptron"*, Mar. 2013, https://tex.stackexchange.com/a/104376 (visited on 02/24/2023).

[22] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", in Proceedings of the 27th International Conference on Machine Learning (ICML-10) (2010), pp. 807–814.

[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators", *Neural Networks* **2**, 359–366 (1989), https://linkinghub.elsevier.com/retrieve/pii/0893608089900208 (visited on 02/02/2023).

[24] S. Park, C. Yun, J. Lee, and J. Shin, *Minimum Width for Universal Approximation*, June 2020, http://arxiv.org/abs/2006.08859 (visited on 02/02/2023).

[25] *Att_00028.Png (PNG Image, 1564 × 381 Pixels) – Scaled (69%)*, https://fastai.github.io/fastbook2e/images/att_00028.png (visited on 02/24/2023).

[26] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, *The Cityscapes Dataset for Semantic Urban Scene Understanding*, Apr. 2016, http://arxiv.org/abs/1604.01685 (visited on 08/09/2022).

[27] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, Jan. 2017, http://arxiv.org/abs/1412.6980 (visited on 09/19/2022).

[28] P. V. C. Hough, "Machine Analysis of Bubble Chamber Pictures", *Conf. Proc. C* **590914**, edited by L. Kowarski, 554–558 (1959).

[29] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures", *Communications of the ACM* **15**, 11–15 (1972), https://doi.org/10.1145/361237.361242 (visited on 02/27/2023).

[30] H. K. Yuen, J. Princen, J. Dlingworth, and J. Kittler, "A Comparative Study of Hough Transform Methods for Circle Finding", in Procedings of the Alvey Vision Conference 1989 (1989), pp. 29.1–29.6, http://www.bmva.org/bmvc/1989/avc-89-029.html (visited on 02/27/2023).

[31] P. Kierkegaard, "A Method for Detection of Circular Arcs Based on the Hough Transform", *Machine Vision and Applications* **5**, 249–263 (1992), https://doi.org/10.1007/BF01212714 (visited on 02/27/2023).

[32] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, May 2015, http://arxiv.org/abs/1505.04597 (visited on 08/09/2022).

[33] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, Dec. 2015, http://arxiv.org/abs/1512.03385 (visited on 09/26/2022).

[34] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, Mar. 2015, http://arxiv.org/abs/1502.03167 (visited on 08/10/2022).

[35] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, *Bag of Tricks for Image Classification with Convolutional Neural Networks*, Dec. 2018, http://arxiv.org/abs/1812.01187 (visited on 09/26/2022).

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database", in 2009 IEEE Conference on Computer Vision and Pattern Recognition (June 2009), pp. 248–255.

[37] A. Tarvainen and H. Valpola, *Mean Teachers Are Better Role Models: Weight-averaged Consistency Targets Improve Semi-Supervised Deep Learning Results*, Apr. 2018, http://arxiv.org/abs/1703.01780 (visited on 11/11/2022).

[38] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "Scikit-Image: Image Processing in Python", *PeerJ* **2**, e453 (2014), https://peerj.com/articles/453 (visited on 02/24/2023).

[39] K. Wu, E. Otoo, and A. Shoshani, "Optimizing Connected Component Labeling Algorithms", (2005), https://escholarship.org/uc/item/7jg5d1zn (visited on 02/24/2023).

[40] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the Shape of a Set of Points in the Plane", *IEEE Transactions on Information Theory* **29**, 551–559 (1983).

[41] K. Bellock, N. Godber, and P. Kahn, *Bellockk/Alphashape: V1.3.1 Release*, Zenodo, Apr. 2021, https://zenodo.org/record/4697576 (visited on 02/24/2023).

[42] *OpenCV: Image Inpainting*, https://docs.opencv.org/3.4/df/d3d/tutorial_py_inpainting.html (visited on 02/24/2023).

[43] A. Telea, "An Image Inpainting Technique Based on the Fast Marching Method", *Journal of Graphics Tools* **9**, 23–34 (2004), http://www.tandfonline.com/doi/abs/10.1080/10867651.2004.10487596 (visited on 02/24/2023).

[44] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting", in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Vol. 1 (2001), pp. I-355-I–362, http://ieeexplore.ieee.org/document/990497/ (visited on 02/24/2023).

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Dec. 2019, http://arxiv.org/abs/1912.01703 (visited on 02/20/2023).

[46] CVAT.ai Corporation, *Computer Vision Annotation Tool (CVAT)*, Sept. 2022, https://github.com/opencv/cvat.

[47] *CrossEntropyLoss — PyTorch 1.13 Documentation*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html (visited on 02/21/2023).

[48] *ReduceLROnPlateau — PyTorch 1.13 Documentation*, https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html (visited on 02/28/2023).

[49] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results", in 2020 11th International Conference on Information and Communication Systems (ICICS) (Apr. 2020), pp. 243–248.

[50] "A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique", *International Journal of Recent Trends in Engineering and Research* **3**, 444–449 (2017), http://www.ijrter.com/papers/volume-3/issue-4/a-review-on-imbalanced-data-handling-using-undersampling-and-oversampling-technique.pdf (visited on 02/22/2023).

[51] S. Scherer, R. Schön, and R. Lienhart, *Pseudo-Label Noise Suppression Techniques for Semi-Supervised Semantic Segmentation*, Oct. 2022, http://arxiv.org/abs/2210.10426 (visited on 12/02/2022).

[52] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*, Aug. 2019, http://arxiv.org/abs/1905.04899 (visited on 02/22/2023).

[53] *OpenCV: Feature Detection*, https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d (visited on 02/27/2023).

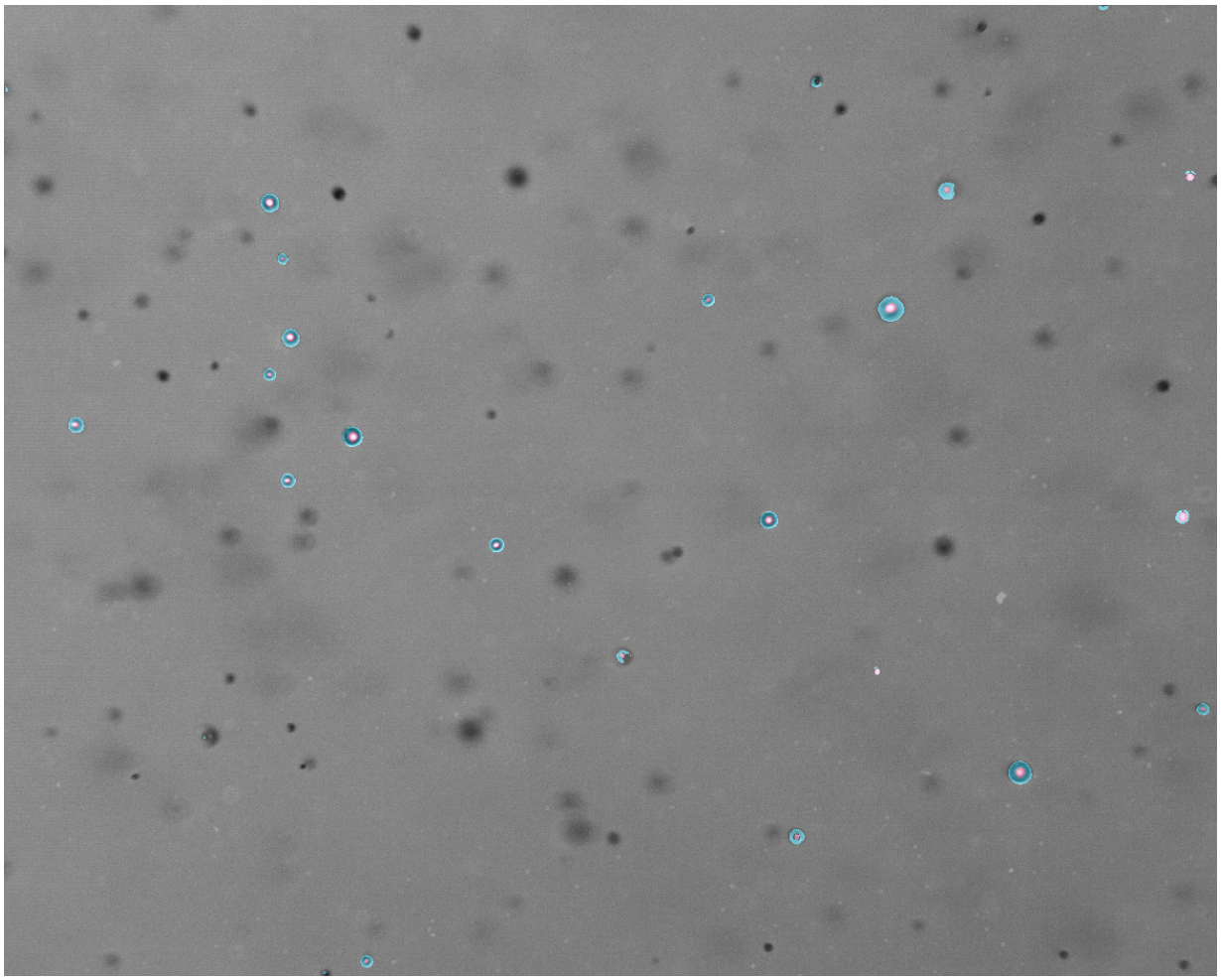
Appendices

A. Abbreviations

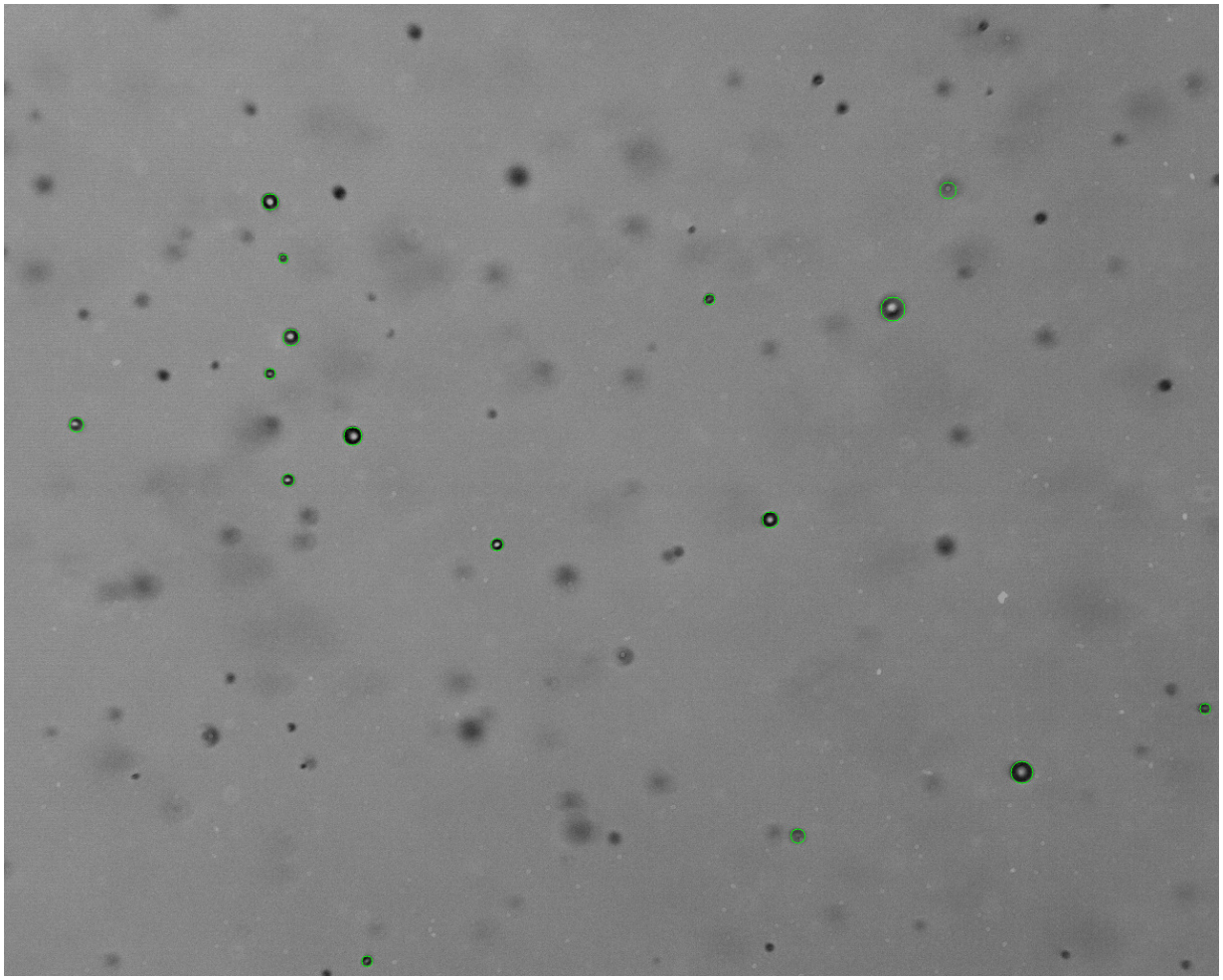
Abbreviation	Meaning
(D/F)CNN	(Deep-/Fully-) Convolutional Neural Network; a (deep) neural network that (only) includes convolutional layers in its main path
SAW	Surface Acoustic Wave
IDT	Interdigital Transducer
lr	learning rate
lrsp	learning rate scheduler patience
lrsf	learning rate scheduler factor; the number the learning rate is multiplied by in case the scheduler detects stagnant progress
bs, bsize	batch size
lbls	labeled samples
MT	Mean Teacher
EMA	Exponential Moving Average
sf	split factor; the number the height and width of the images and masks in the dataset are divided by for splitting them into smaller samples

Table A.1.: A list of abbreviations used in this thesis. The table explains the full meaning of each abbreviation as well as a short description if it is not obvious.

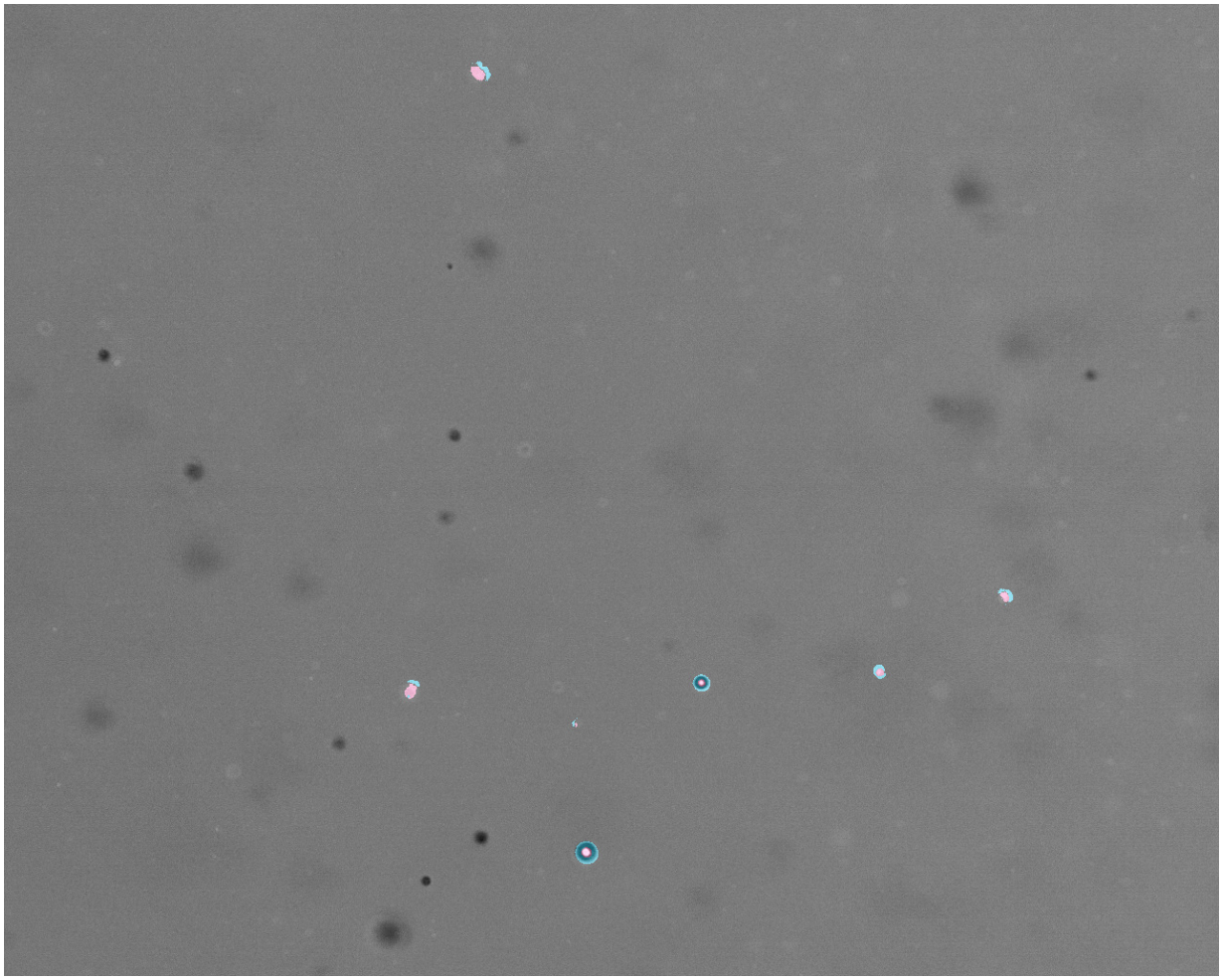
B. Example Images



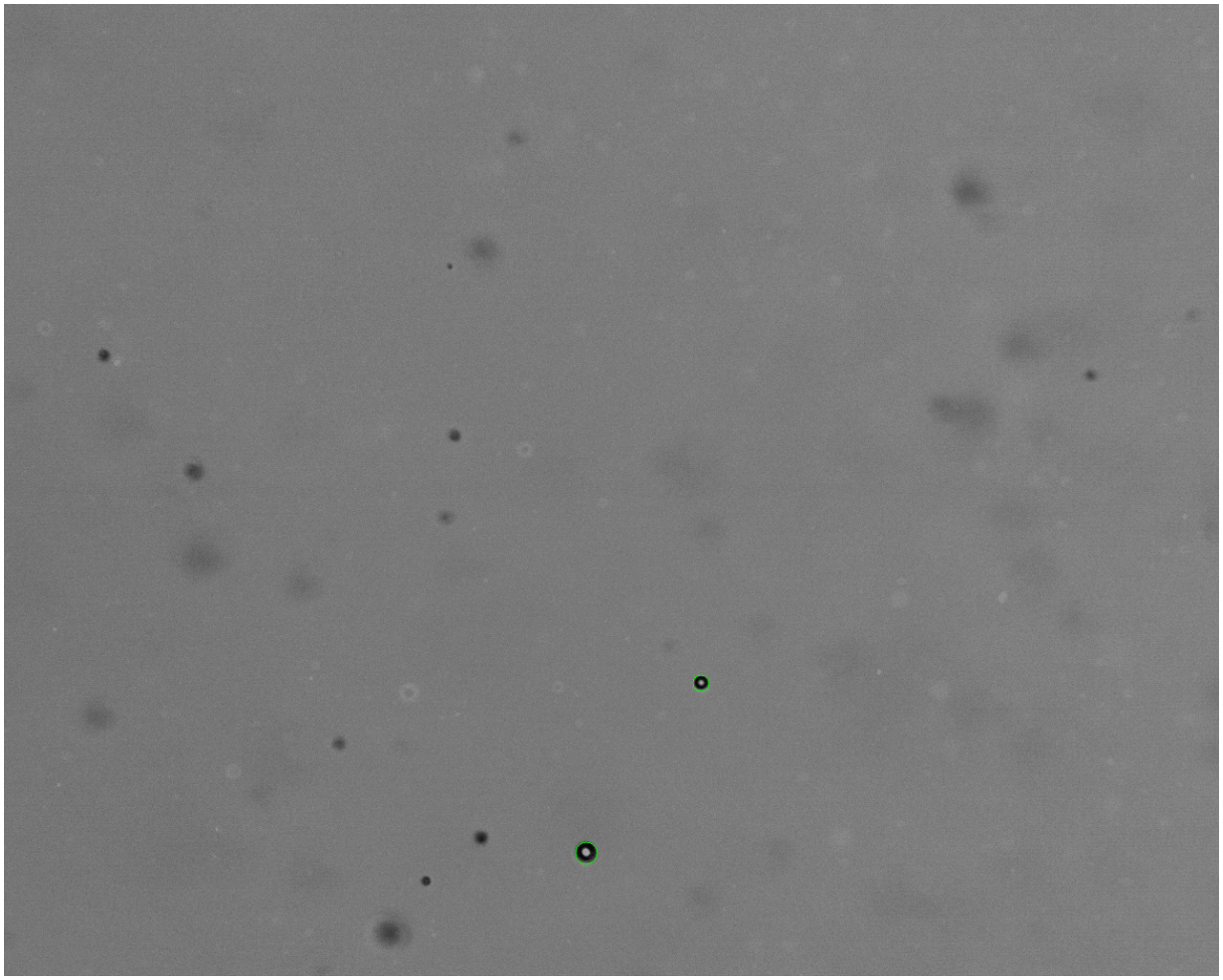
(a) Mask overlay for example image 1.



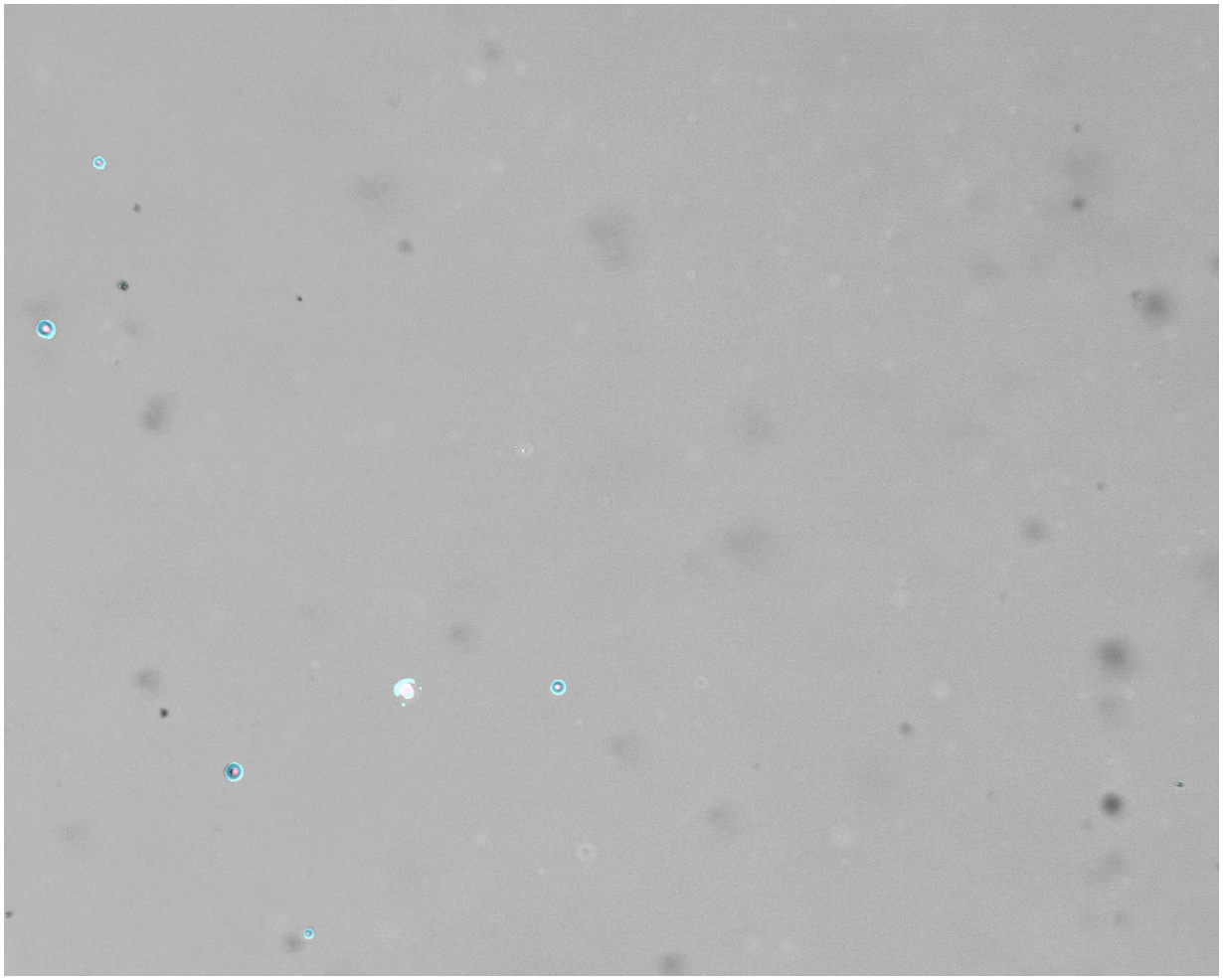
(b) Detected droplets for example image 1.



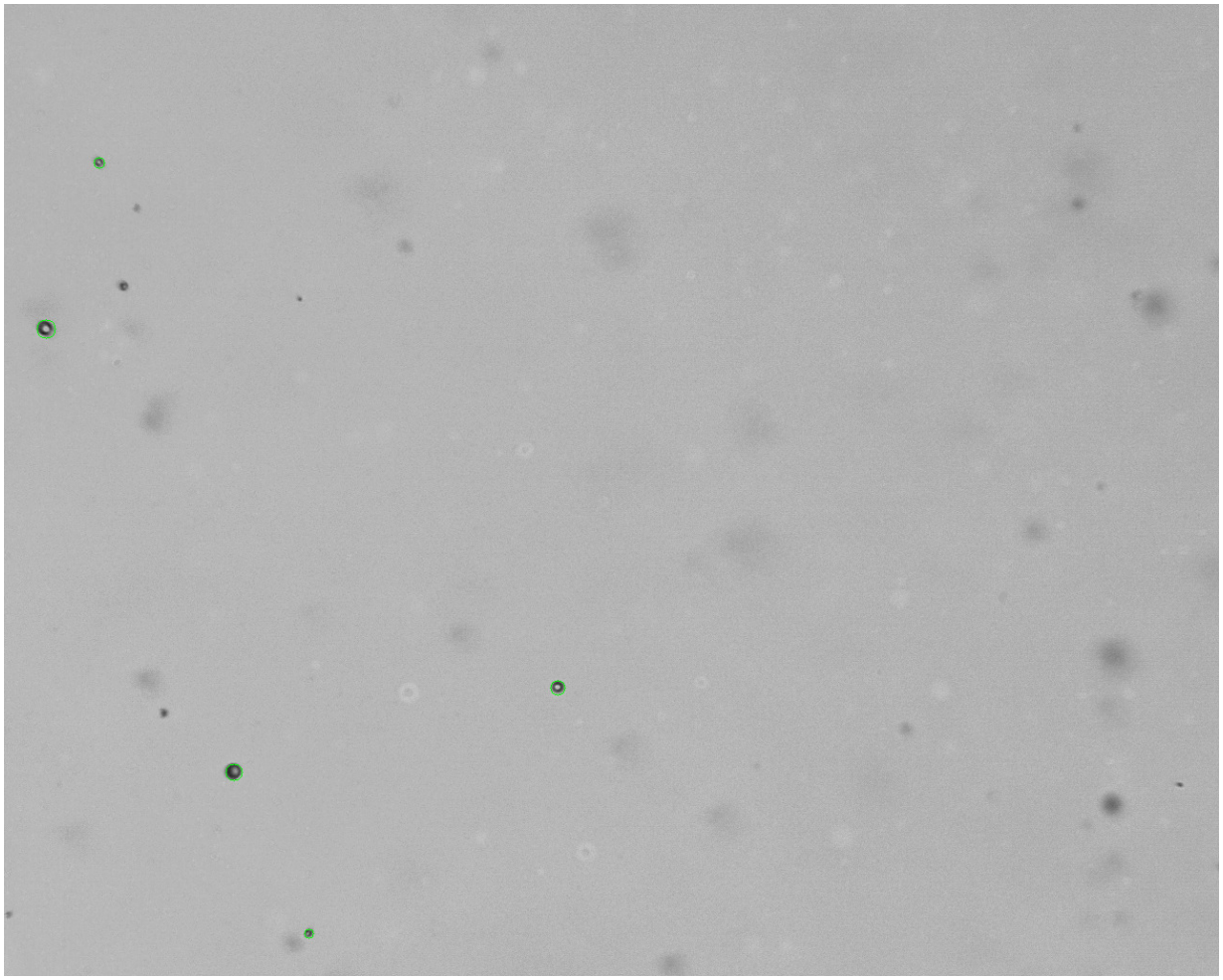
(c) Mask overlay for example image 2.



(d) Detected droplets for example image 2.



(e) Mask overlay for example image 3.



(f) Detected droplets for example image 3.

Figure B.1.: Examples for some of the results from applying the measurement technique developed in the thesis. The images show the overlayed segmentation masks with blue for the droplets border labels and pink for the droplet center labels, as well as the droplets detected after filtering as green circles.