UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

WELDING DROPLET SEGMENTATION USING DEEP LEARNING

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

IVÁN IGNACIO GONZÁLEZ PÉREZ

PROFESORA GUÍA:
VIVIANA MERUANE NARANJO

MIEMBROS DE LA COMISIÓN:
PATRICIO MÉNDEZ
RUBÉN FERNÁNDEZ

SANTIAGO DE CHILE
MARZO 2021

RESUMEN DE: TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA Y MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: IVÁN IGNACIO GONZÁLEZ PÉREZ
FECHA: MARZO 2021
PROF. GUÍA:VIVIANA MERUANE NARANJO

WELDING DROPLET SEGMENTATION USING DEEP LEARNING

Recently, deep learning models have had an outstanding performance in tasks such as image classification, anomaly detection and natural language processing. These models thrive when the amount of data is large, which is common nowadays. Coupled with the increasing computing power of GPU, it is possible to solve many of otherwise intractable problems. Nevertheless, the deep learning approach has been mainly used in computer vision while other fields have not adopted it as much. Hence, there is an opportunity to use these models to outperform previous results in the welding field.

In the following work the aim is to use deep learning segmentation models to solve the problem of obtaining relevant features of a Gas Metal Arc Welding (GMAW) process. This is done by segmenting video footage to isolate droplets and then calculating features that are relevant to characterize the process. This problem has been addressed before with computer vision techniques, but the methods used lack automation, and processing large volumes of data becomes unfeasible. Therefore, this thesis' approach allows to achieve results in the segmentation problem itself and the automation process, since results can be obtained faster.

The proposed model is a Fully Convolutional Network approach. Several architectures are considered and compared to then use the best one for further calculations, which by means of supervised training can take a large amount of images and return segmentation masks for each one, isolating the droplets from the background. Furthermore, segmentation masks are used to compute geometric and kinematic properties. This can be helpful to understand and design the welding process, since it would be possible to map the inputs like current, voltage and shielding gas to a measurable output such as position, area, frequency and velocity of droplets.

A literature review is done to understand the problem and how it has been addressed so far and to study the approaches for segmentation in deep learning. Then, data is acquired, consisting of two videos of GMAW processes depicting globular and spray transfer which are manually labeled and augmented. Later, Fully Convolutional Network based architectures are trained with the labeled data, namely U-Net, DeconvUnet and MultiResUnet. Finally, the resulting segmentation masks are processed to compute geometric and physical properties.

The main conclusion of this work is that the U-Net based approach can reliably segment droplets within a frame, achieving similar results to previous attempts, but with the benefit of being able to process thousands of images in seconds or minutes. Furthermore, relevant properties were obtained, namely position, velocity, acceleration, perimeter, area, volume and surface tension, with values mostly in agreement with literature. Hence this work is a successful step towards automation and a better understanding of GMAW.

RESUMEN DE: TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA Y MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: IVÁN IGNACIO GONZÁLEZ PÉREZ
FECHA: MARZO 2021
PROF. GUÍA:VIVIANA MERUANE NARANJO

WELDING DROPLET SEGMENTATION USING DEEP LEARNING

Recientemente, modelos de aprendizaje profundo han obtenido excelentes resultados en tareas como clasificación de imágenes, detección de anomalías y procesamiento de lenguaje. Estos modelos mejoran cuando hay una gran cantidad de datos, lo que es común hoy en día. Sumado al aumento en poder de cómputo de las GPU, es posible resolver problemas que de otro modo serían intratables. No obstante, el enfoque de aprendizaje profundo se ha aplicado principalmente en visión computacional, mientras que en otros campos de la ciencia no se ha aplicado al mismo nivel. De este modo, existe una oportunidad de usar estos modelos y mejorar resultados en el campo de la soldadura.

En el siguiente trabajo el propósito es utilizar modelos de aprendizaje profundo para obtener características relevantes de un proceso de soldadura a gas y arco metálico (GMAW). Lo anterior consiste en segmentar imágenes para separar a las gotas del fondo y luego calcular parámetros. Este problema ha sido abordado con el uso de técnicas de visión computacional, sin embargo las técnicas usadas carecen de automatización, por lo que procesar una gran cantidad de datos se hace inviable. El enfoque de esta tesis permite resolver el problema de segmentación en sí, y el problema de automatización, pues los resultados se pueden obtener rápidamente.

El modelo propuesto se basa en Fully Convolutional Networks, las cuales mediante entrenamiento supervisado reciben imágenes y entregan una máscara de segmentación. Luego, las máscaras se usan para calcular propiedades geométricas y físicas, lo cual es de ayuda para comprender y diseñar el proceso de soldadura, ya que permite establecer una relación entre input como voltaje, corriente o gas protector y output como posición, área, frecuencia o velocidad de las gotas.

Se realiza una revisión de literatura para comprender el problema y cómo se ha abordado hasta ahora, y para estudiar propuestas de segmentación con aprendizaje profundo. Luego, se recopilan datos para un caso de estudio, los datos consisten en videos de modos de transferencia globular y spray que luego son manualmente etiquetados. Posteriormente se entrenan modelos Fully Convolutional con los datos etiquetados, en particular se prueban las arquitecturas U-Net, DeconvNet y MultiResUnet. Finalmente, las máscaras de segmentación se utilizan para calcular características importantes.

La conclusión de este trabajo es que la propuesta de un modelo tipo U-Net es capaz de segmentar correctamente los videos, consiguiendo máscaras similares a propuestas anteriores, pero con la ventaja de poder procesar miles de imágenes en segundos o minutos. Además, se obtiene la posición, velocidad, aceleración, perímetro área, volumen, tensión superficial y tasa de desprendimiento mayormente en concordancia con la literatura. Por lor tanto, este trabajo supone un exitoso paso hacia la automatización y la mejor comprensión de GMAW.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

The optimization of arc welding processes in general has always been an important task since it is necessary to get better results both in the quality of the joints as well as the efficiency of the process in terms of time and materials to produce durable and reliable results. There are constant efforts in improving the welding machines, using better quality materials, physical modeling of the process and increasing the automation of both the welding itself as well as the design in order to achieve an optimized welding process.

A widely used arc welding technique is Gas Metal Arc Welding (GMAW) in which a bare metal wire electrode is consumed while a shielding gas floods the area to avoid external contamination [1]. The study of this process is a complex task, since there are many fields involved such as fluid dynamics, heat transfer and solid mechanics. Also, from a technical standpoint there are several parameters that affect the process as a whole, such as voltage and current waveform, material and diameter of the electrode and shielding gas.

One way to approach the study of GMAW in order to help understand and optimize the process is by using data-driven models. These models have thrived in recent years because no analytical model of the phenomena is required and the high volume of data that can be collected using a wide range of sensor information such as image, acoustic signal, temperature, pressure, among many others.

Specifically, deep learning models have been able to solve problems which would otherwise be difficult or intractable. Deep learning models have reached state of the art results in tasks such as image classification, image segmentation, natural language processing and optimal control taking advantage of the large datasets available [2] and advances in hardware. Also, different scientific fields outside of the typical computer vision applications have successfully adopted this approach, such is the case in bio-medicine, prognostics and health management and the mining industry to name a few [3, 4, 5].

Furthermore, a data-driven approach for the GMAW study poses the challenge of which parameters to measure and how to do it, since the process itself occurs at very high temperatures, it is difficult to use a measuring device directly, because it would need to resist these conditions while functioning optimally as well as not disturbing the process.

Therefore, a suitable approach would be to use image data, since the camera can stay away from the process while capturing important information, namely the movement of the welding

droplets from the wire to the weld pool. For this, the experimental setup would need a high speed video camera to properly record the process and the necessary filters to overcome the plasma glow and other visual effects.

Although deep learning models have been used to address problems like welding machine control, defect detection and weld bead geometry prediction [6, 7, 8], the GMAW segmentation problem has only been tackled using standard computer vision techniques [9, 10] in which the process is recorded and the video is processed to segment each frame and isolate the droplet from the background. Later, several geometric and kinematic properties can be computed. While suitable for the problem, hitherto used methods cannot handle the amount of data that can be retrieved from the process when using high speed video cameras.

Consequently, a data-driven approach using deep learning models would be suitable to solve a task such as droplet segmentation in a GMAW process while also being able to process thousands of images in seconds or minutes after the training is done.

The segmentation problem is not unique to the arc welding study, and has been addressed in other contexts using different techniques, including deep learning, to solve problems such as localizing medical abnormalities [11] like aneurysms, tumors or cancerous elements [12, 13, 14]. Surveillance is also a common application for pedestrian detection and traffic surveillance [15, 16]. Other fields in which segmentation is applied are image detection in forensics [17, 18, 19] and satellite imagery [20].

The main idea in the segmentation task is to have an input image that is then separated into two or more regions. Image segmentation can refer to several related problems, the classic version of the problem is semantic segmentation in which each pixel of an image is labeled from a predefined set of possible classes so that the resulting regions have some kind of visual or semantic relationship. Another kind of segmentation is instance segmentation, in which subjects are not only separated semantically, but also instance-wise, that is multiple occurrences of the same subject are labeled differently [21]. Applied to the GMAW droplet segmentation problem, semantic segmentation would consist in assigning each pixel either a droplet label or a background label. Furthermore, instance segmentation would also assign separate labels to each droplet.

In the deep learning framework, a common family of models used for segmentation tasks are the Fully Convolutional Networks (FCN). These networks consist only of convolutional layers and therefore can output an image from an image input which is not possible when using fully connected layers since they flatten the input [21].

Particularly, the U-Net model, which has had great success in tasks using biomedicine image data, like alveolar segmentation and mitochondria detection [22, 23] is an architecture of interest for this thesis. These datasets have rather simple subjects compared to other tasks in traffic segmentation or face recognition and the problem is mainly subject-background separation, which is the same scenario as in the GMAW segmentation problem. Additionally, the U-Net model also has variants that have been used in similar problems like the DeconvUnet and MultiResUnet [24, 25].

For these reasons, this thesis aims to use Deep Learning models to be able to segment a large

number of images and calculate geometric and kinematic features of droplets from a GMAW process in an automated and reliable manner to then calculate kinematic and geometric properties that characterize the process.

1.2 Motivation

The purpose of this work is to further develop the automated analysis of a welding process, specifically a GMAW setup with globular and spray transfer modes, by using deep learning techniques. As stated before, there is an ever increasing use of deep learning techniques in diverse fields of study because of the large amounts of data available, as well as the outstanding results they can achieve, specially in computer vision with the use of convolutional neural networks. This coupled with necessity of a better understanding of the welding process makes it an interesting project, since segmentation problems have not been addressed for a large volume of high speed video images.

1.3 Aim of the work

1.3.1 General objective

Propose a deep learning supervised segmentation model for high speed video recordings of globular and spray metal transfer for the posterior calculation of relevant properties of the droplets such as position, velocity, acceleration, perimeter, area, volume and surface tension.

1.3.2 Specific objectives

- Retrieve image data from High Speed Videos of globular and spray metal transfer processes to build a dataset.
- Manually label images in order to have ground truth data to train a supervised deep learning model.
- Implement deep learning segmentation models, namely fully convolutional networks, to generate segmentation masks for every image in their respective datasets (globular and spray transfer). Measure error metrics such as Jaccard distance. Then, choose the best performing model and compare obtained masks with original images
- Use the generated masks to calculate properties of the droplet and the process in general: Position, velocity, acceleration, perimeter, area, volume, detachment frequency and surface tension.

1.3.3 Scope

The welding videos are provided by the Canadian Centre for Welding and Joining (CCWJ), so the video files are the starting point for this thesis. Therefore, no experimental setup is studied because this has been previously addressed at the CCWJ laboratories when shooting the videos. The GMAW metal transfer modes considered are globular and spray.

The processing of the videos in this research intends to use the files as separate frames to get a label for each one separating the background from the droplet boundaries using

Fully Convolutional Neural Networks (FCN) for which a small sample of the dataset is labeled manually using the LabelBox platform. A couple of models are to be compared: U-Net, DeconvNet and MultiResUnet. Then, the best performing model will be used to get the definitive predictions. The different transfer modes are trained separately, so there is one model for globular and another for spray. Although it is possible to train one model with all the data, this would trade accuracy for generality which is not convenient when measuring the droplet properties considering that building two models is inexpensive in terms of computation time. Furthermore, the generated labels are used to calculate relevant features of the process which can then be used by an expert to better understand the process.

The code is written in `python` using existing libraries for reading video files, computer vision, data science and deep learning (`pycine`, `numpy`, `scipy`, `cv`, `tensorflow`, etc).

1.4 Structure of the work

In Chapter 2 the methodology of the work is presented. Subsequently, in Chapter 3 background is shown. This includes the problem of welding optimization in general and the attempts made to solve the problem of droplet segmentation specifically. Also, a background of deep learning techniques is given, particularly the approach for segmentation tasks. After, in Chapter 4 the proposed deep learning models are explained in detail. Then, in Chapter 5 study cases are shown, the datasets are explained and the differences and nuances of each GMAW transfer mode are discussed. Also, preprocessing techniques are explained. In Chapter 6 results are shown and analyzed, its implications and relevance are discussed. Finally, in chapter 7 conclusions are made regarding the proposed model and how it can provide important information of the welding process, and the potential of deep learning in the welding field.

Chapter 2

Methodology

This work explores the use of supervised deep learning segmentation models to process GMAW high speed videos for further postprocessing. The process to achieve this is as follows:

1. **Literature review:** The first step consists in broadly studying the GMAW welding process and why it would be important to optimize and understand it better. Furthermore, the specific approach of image segmentation is studied within the field, to know what has been done until now.

   In parallel, a study of the deep learning segmentation models and applications is carried out to know the tools available to solve the task and which kinds of problems have been tackled so far.

2. **Gathering and preprocessing the data:** The dataset used was kindly provided by the Canadian Centre for Welding and Joining (CCWJ). These are high speed videos of GMAW processes for globular and spray transfer modes. The videos are `.cine` files which are read in `python` with the use of the `pycine` library. In this way, each video is stored as a 4D array (time, width, height, channels) into a `.npz` file.

   Additionally, since the proposed model is supervised, a small sample of images is taken to be labeled using LabelBox. The samples are carefully selected so that the whole process is depicted. Also, each transfer mode has its own set of sampled data.

   Finally, the dataset is artificially augmented because the amount labeled images is rather small. Multiple transformations are performed such that each image yields five augmented images.

3. **Develop semantic segmentation models:** A handful of models are used to be compared, these are the U-Net, DeconvUnet and MultiResUnet, they are built in `tensorflow` 2.3.1. The models are trained with image files from the sampled ones that have labels, then the rest of the images without labels are tested. Since there are two different transfer modes considered, globular and spray, they are trained separately, so for each network there are two trained models. Moreover, several validation methods are used to ensure the best results such as k-fold cross validation and grid searching, all of which are described in chapter 3.

4. **Postprocess segmentation masks:** The outputs of the segmentation network are then processed using `opencv` and `scipy` to calculate several relevant characteristics such as position, velocity, area and frequency.

5. **Analysis and conclusion:** The proposed models are compared to discuss about which one would be better for the task. Additionally, the postprocessed features are discussed in terms of their usefulness when optimizing or designing a GMAW process. Finally, future prospects of deep learning techniques in the welding industry are presented.

2.1 Resources available for this Thesis

The computational requirements for this thesis were met by a desktop computer provided by the Smart Reliability and Maintenance Integration (SRMI) Laboratory of the University of Chile with the following specifications:

- Ubuntu 16.04.06 LTS.
- Intel Core i5-7600K CPU @ 3.80GHz x 4.
- 32GB RAM.
- NVIDIA TITAN Xp/PCIe/SSE2.

Additionally, the software and libraries needed include:

- Phantom Camera Control software to view `.cine` files and inspect metadata.
- LabelBox tools for segmentation of images.
- `Python` 3.8 as programming language.
- `Pycine` library to read `.cine` files in `python`[1].
- `Imgaug` for augmenting images.
- `TensorFlow` 2.3.1 as a deep learning framework.
- `Numpy`, `pillow`, `scikit-learn`, `opencv`, `scipy` for preprocessing and post-processing of data.
- `Matplotlib` and `seaborn` for data visualization.
- Dependencies required for the above to work properly.

Regarding the dataset, it was provided by the Canadian Centre for Welding and Joining (CCWJ) at the University of Alberta, this collaboration was carried out through the Emerging Leaders in the Americas Program (ELAP) which awarded the author a scholarship to undergo an investigation internship to the University of Alberta for a period of six months.

Finally, version control of the project is carried out using Git. All of the code, results and related documents can be found in GitHub[2].

[1] `https://github.com/OTTOMATIC-IO/pycine`

[2] `https://github.com/igonzalezperez/Welding-droplet-analysis-using-fully-convolutional-networks`.

Chapter 3

Background

In this chapter the welding aspects of the problem are explained and previous work is shown regarding the segmentation problem within welding research as well as the use of machine learning algorithms in the field.

Additionally, theoretical background of the deep learning techniques used is shown. For this, an overview of machine learning is given, detailing the problems it tackles and how this approach works in general. Furthermore, different types of Artificial Neural Networks (ANN) are described, namely fully connected networks, convolutional neural networks (CNN) and fully convolutional networks (FCN).

3.1 Welding

In this thesis the focus is in Gas Metal Arc Welding (GMAW), a welding process in which a consumable electrode is melted by an electric arc. The zone has a shielding gas to prevent external effects in the joint. Figure 3.1 shows a typical GMAW diagram.

The quality of an arc welding operation is highly influenced by the transfer mode. Previous work in GMAW indicates that transfer mode affects weld penetration, weld width, recovery of alloying elements, fume emission, spatter, wetting among others [27, 28]. Furthermore, stable and repeatable transfer modes are achieved by using the correct welding parameters [29], hence it is important to gain knowledge about the welding dynamics to improve automatic welding systems.

In this thesis, globular and spray transfer modes are considered. Globular transfer consists of large droplets that build up in the end of the consumable electrode until it detaches due to the size. This transfer mode is usually not desirable since it produces high heat, poor welding surfaces and spatter. Nonetheless, it is a cost effective mode because carbon dioxide is used as shielding gas which is cheaper than other gases such as argon. On the other hand, spray transfer uses higher voltage and current, producing a fast melting of the electrode, resulting in small droplets. This increases the quality of the weld and spatter is minimized.

The analysis of welding image data can be useful to understand and design the process. Specifically, it is possible to measure kinematic and geometric properties which can provide useful information when designing and optimizing a GMAW process as well as characterizing it. Furthermore, this can help to study mathematical descriptions of droplet flight and better development of automatic welding systems.

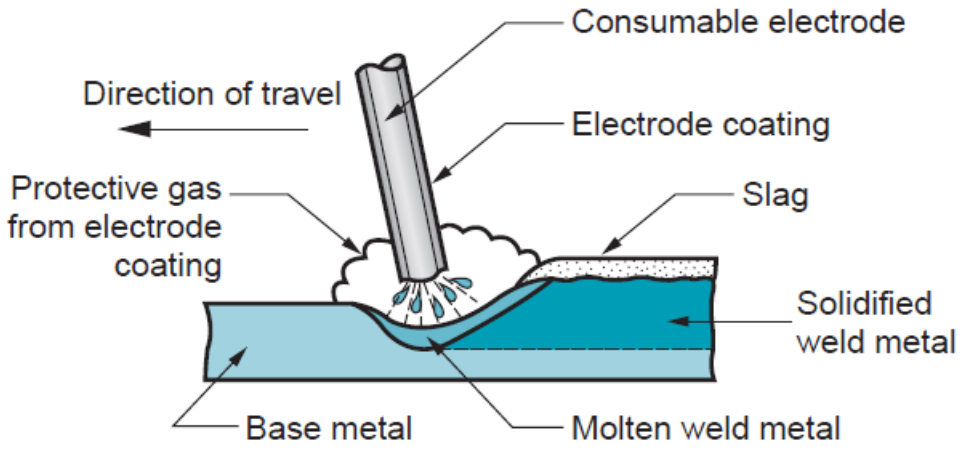


Figure 3.1: GMAW process illustration. Source: *Fundamentals of modern manufacturing : materials, processes, and systems* [26].

3.1.1 Previous work

The use of high speed video welding data has been explored before, using computer vision techniques that can detect the contour of objects within an image, with the aim of obtaining characteristic features of the metal droplets.

In *Ray et al.* [9] an active contour model, also known as *snakes*, was used and was able to accurately outline the border of a droplet. This is a method for outlining an image object and it is widely used in object tracking, shape recognition and edge detection. It is an analytic method in which a set of seeded splines are moved in the image by minimizing an energy function which depends on features of the image, namely lines, edges and terminations; the energy minimum is reached when the points of the spline are able to outline the shapes in the image [30] as shown in figure 3.2.

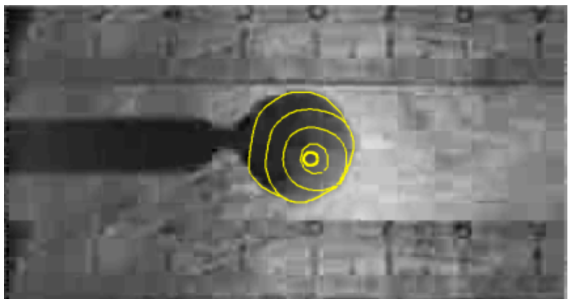
Another case of GMAW droplet segmentation is in *Zhai* [10] in which the problem is solved by using standard techniques of computer vision. Several image enhancement filters were applied to the image in a pipeline, such as grayscaling, thresholding, edge operators (Sobel operator, Laplacian of Gaussian operator, Canny operator among others) and histogram equalization in order to separate the droplet from the background. This approach has a different process for attached and detached droplets.

Furthermore, in the work of *Wang* [31], a similar approach to *Zhai* [10] was used, that is applying successive filters to the images to get a proper segmentation. Then, location and size of the droplet were measured accurately.

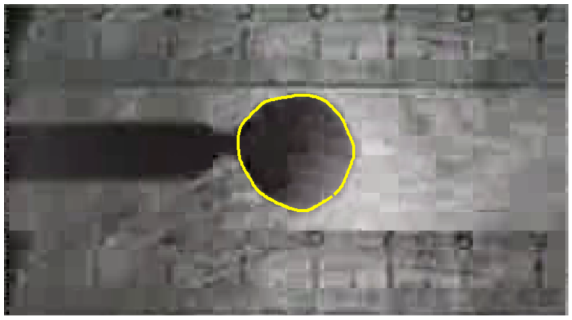
While the droplets were properly segmented in the examples mentioned above, this was done in a per image fashion, this means that to analyze thousands of images one would have to run the model thousands of times which is time consuming[1]. Also, in *Zhai* [10], different models are needed for attached and detached droplets. Therefore it is necessary not only to properly

[1]In the work of *Wang* [31] it is stated that the speed could be greatly improved by migrating from Matlab based code to C, but this is not done in the paper.

(a) Snake seed



(b) Evolution of snake from the seed



(c) Final evolved snake

Figure 3.2: Snake methodology used in *Ray et al.* [9]

segment, but to be able to process a large amount of images in a more general approach.

Although deep learning techniques have been used in welding imaging problems, it has been mainly in a reinforcement learning approach to control the welding machine *in situ* [6]. Similarly, neural networks have been used for defect detection using several sensors, this was achieved by the use of Convolutional Neural Networks to classify the input into one of different degrees of quality of the result, that is to say modes of failure [7]. Recently, deep learning has also been used for weld bead geometry prediction [8].

Nonetheless, to the best of the author's knowledge, segmentation problems have not been addressed with the use of deep learning techniques in the study of welding video data, which makes this thesis a novel approach to help solving a relevant problem in the field.

3.2 Machine Learning

Machine Learning is a field within Artificial Intelligence which aims to solve tasks automatically by learning patterns from data as opposed to hard-coding specific instructions. In an abstract way, the learning characteristic can be defined as follows [32]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

The task (T) is what the computer has to do, and it gets better at it by processing examples of said task, in other words, gaining experience (E) in the task. The examples are the input of the model, and usually come in the form of a vector $x \in \mathbb{R}^n$. For example, if the input data are RGB images, then $x \in R^3$ and has dimensions ($width$, $height$, $channels$) and each

data point in x is a pixel value. Another example is when each column of x represents a specific feature relevant to the task.

Typical machine learning tasks are the following:

- **Classification:** The task is to assign the input to a specific set of outputs by tuning the parameters of a function such that: $f : \mathbb{R}^n \to \{1, \ldots, k\}$ where k is the number of possible classes. An example of this is the handwritten digits problem [33], in which an image of a handwritten digit is given and the output is which class ($\{0, \ldots, 9\}$) the model is assigning to it.
- **Regression:** In this case, the model's task is to learn a function $f : \mathbb{R}^n \to \mathbb{R}$. The output is not a finite set of values, but a continuous value. Examples of this are when the task is to predict some physical measurement in the future with historical data like predicting tomorrow's temperature (continuous output) based on recent climate data trends.
- **Anomaly detection:** This case is similar to classification, since the task is to flag abnormal data from the rest.The difference lies in the fact that there is one class that is larger than the rest and sometimes the large class is the only class present in the training phase. This can be used to prevent fraudulent use of credit cards (abnormal data).
- **Denoising**: For this case the model learns to represent normal or baseline data, so that when a corrupted example is given, the model can reduce the noise in the corrupted example.

The performance measure (P) is important since it is the metric that the model will use to effectively learn from the given examples. In classification tasks, P can be the accuracy, that is how many correct predictions were made out of the total examples. In the case of regression it makes more sense to use a distance metric to measure how far away the predicted value is from the real value. In anomaly detection, since one class is way larger than the other, accuracy can be deceiving and other metrics are used such as the F1-score. However, choosing an appropriate metric is not always straightforward and it is important to understand the task and the nature of the data to decide which metric would be more effective [34].

Another important aspect is that the performance measure that matters is the one computed in new data, since training data has already been seen, it is expected that the error is going to be small when measured using that data, or at least smaller than in the unseen data. Because of that, datasets are usually split into training and testing sets.

Finally, the experience (E) will also determine how to approach the problem. The type of data (experience) can be categorized as follows:

- **Supervised learning:** The data consists of pairs (x, y) in which x is the input and y is the correct output for x. In the handwritten digits examples, a data point would be x, an image with a hand drawn digit $n \in \{0, \ldots, 9\}$, and y, the integer number n.
- **Unsupervised learning:** In this case, no ground truth data is given, only the input x. This can be useful to explore the structure of the data itself by learning representations of it. Also it is used when one does not know how to categorize the data and lets the

model cluster similar examples which can be useful, for example, when categorizing customer preferences or categorizing texts.

- **Reinforcement learning:** In this case the data is constantly being fed to an agent that makes decisions based on the stream of data. This is used for optimal control purposes, in which the agent must react to different scenarios by changing its behavior such as in self-driving cars.

It is important to note that supervised learning is more "expensive" compared to unsupervised, since labels are not always available, and it is often the case that manual labels need to be made in order to train in a supervised approach, which can be slow when the data is large. Nonetheless, supervised learning allows to have more accurate results, since the exact[2] result is given to the model, hence it is possible for the network to reach a high performance.

3.3 Artificial Neural Networks

An Artificial Neural Network (ANN) is a mathematical model that learns a function $y = f(x; w)$ where y is an output (prediction), x is an input and w are the parameters (or weights) of f. The network learns parameters w such that $f \approx f^*$ where f^* is the function that correctly maps input to output.

These kinds of models are said to be layered since the function $f(x; w)$ is usually a composition of several non-linear transformations and each composition is thought of as a layer of the model. Then f can be represented as

$$f(x; w) = f^{(n)}(f^{(n-1)}(\dots f^{(2)}(f^{(1)}(x; w))) \tag{3.1}$$

Thus, the data flows in a feedforward fashion from the first layer or input layer $f^{(1)}$ through the *hidden layers* of the network until the last layer $f^{(n)}$ which is the output layer. The use of deeper composition chains is known as *deep learning*.

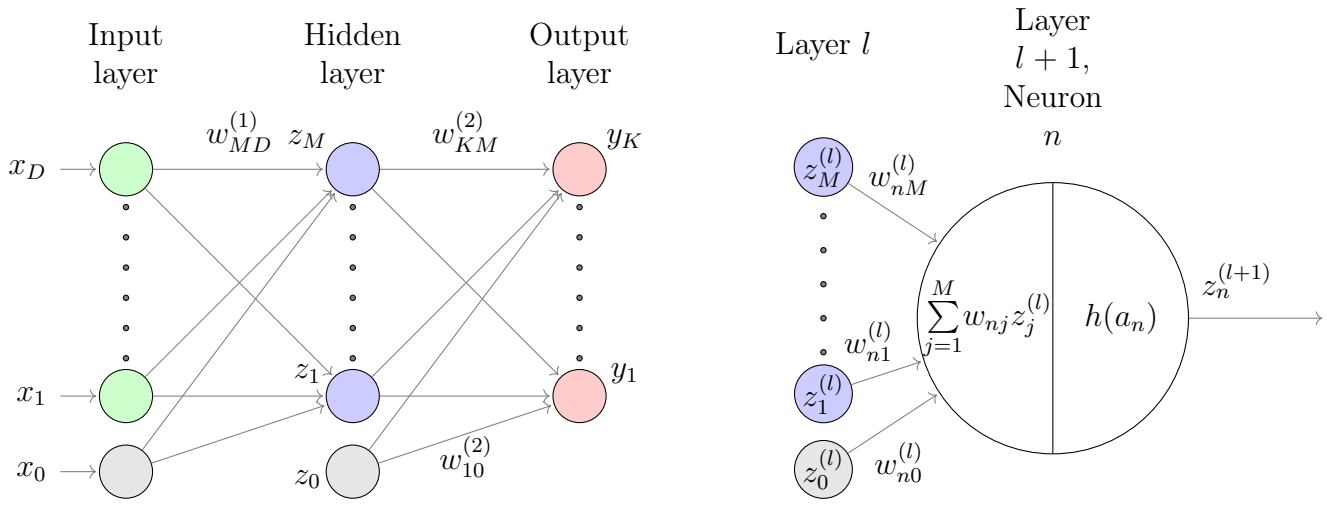
In a supervised context, the model looks for a function $f(x) \approx f^*(x)$ where the training data is a noisy approximation of $f^*(x)$. The data consists in pairs (x, y) where $y \approx f^*(x)$, so the input and output layers are determined, and the model tunes the parameters in the hidden layers in order to get an accurate output.

The "neural" characteristic comes from a loose analogy to how biological neurons work, in which a sensory input is given, then a feedforward chain of neurons fire to propagate the information to the brain so that it can perceive and make sense of the input. Also, neurons have a trigger, namely an action potential which works by differences in voltage in order to be activated. In ANN these triggers are the activation functions which originally resembled this action potential behavior by using logistic functions in which extreme values tend to "collapse" to specific values, emulating the not triggered/triggered states.

In figure 3.3a is a simple ANN of two layers[3] is shown. This kind of network is called a

[2]This will depend on the task at hand. Especially when manually labeling data, the ground truth can have bias of the scientist that is labeling. For example in image segmentation, in which the boundaries between subjects are not a clear pixel-wise separation.

[3]The way of counting layers can be somewhat arbitrary, since one could include everything and consider a



(a) Two layer neural network diagram

(b) Neuron unit

Figure 3.3: A two layer neural network diagram is shown in (a). Nodes represent input (green), hidden (blue) and output (red) variables while links represent the weights between nodes in which a bias (gray) weight is added. In (b), the feed forward calculation is shown from a given layer l to a neuron in the following layer. First computing the linear combination of inputs and then applying an activation function.

dense or fully connected network, since every node in one layer is connected to every node in the next layer. In a given layer, each neuron will output an activation value to each of the neurons in the next layer through a linear combination of the inputs x_{i} and the respective weights $w^{l}_{j\mathrm{i}}$. Once the activations are calculated, these are fed to a non-linear activation function producing the new values to be received by the next layer[4]. Then, in the first layer with D neurons, the activations of the second layer are given by

$$a_j = \sum_{\mathrm{i}=1}^{D} w^{(1)}_{j\mathrm{i}} x_{\mathrm{i}} + w^{(1)}_{j0}, \quad j \in \{1, \ldots, M\}$$

where M is the number of neurons in the second layer and D is the number of neurons in the input layer. Also, the superscript (1) indicates that the parameters are from the first layer. The parameters $w_{j\mathrm{i}}$ are the weights connecting the neuron j from the second layer and the neuron i from the first layer. The parameters w_{j0} are biases which serve as an offset to the activation. Later, the activations are transformed using an activation function

$$z_j = h(a_j)$$

where $h(\cdot)$ is a nonlinear function such as the sigmoid or hyperbolic tangent.

three layer network, or only include hidden layers, in which case the network has one layer. The convention here is to count the number of layer connections, since this reflects the parameters that the network will optimize. Hence, it is considered a two layer network.

[4]In figure 3.3b, $h(\cdot)$ is shown as single function, but in principle there could be many different functions. The indices $h^{(l)}_n(\cdot)$ that indicate layer (l) and neuron (n) are omitted for simplicity.

Conversely, the output layer can be computed from the previous layer as

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \quad k \in \{1, \ldots, K\}$$

where K is the number of outputs and z_j are the results of the previous layer. Finally, the output activations must be transformed to map the type of the output data. In a regression problem no transformation is needed, so the identity function is used and $y_k = a_k$. In the case of classification it is necessary to use a function to cast the activations into specific classes

$$\begin{aligned} y_k &= \sigma(a_k) \\ \sigma(a_k) &= \frac{\mathrm{e}^{a_\mathrm{i}}}{\sum_{j=1}^{K} \mathrm{e}^{a_j}} \end{aligned}$$

where $\sigma(\cdot)$ is the *softmax* function, a generalization of the sigmoid for the multiple class case, and it measures the probability $p(C_\mathrm{i}|x)$, the probability that the output belongs to class C_i, where $\mathrm{i} = 1, \ldots, K$, given the input x.

Then, the final output of the network can be expressed as

$$y_k(\mathbf{x}, \mathbf{W}) = \underbrace{\sigma_k \left(\sum_{j=1}^{M} w_{kj}^{(2)} \underbrace{h \left(\sum_{\mathrm{i}=1}^{D} w_{j\mathrm{i}}^{(1)} x_\mathrm{i} + w_{j0}^{(1)} \right)}_{f^{(1)}(x)} + w_{k0}^{(2)} \right)}_{f^{(2)}(f^{(1)}(x))}, \quad k \in \{1, \ldots, K\} \tag{3.2}$$

and by using matrix notation we get the more compact version

$$\begin{aligned} \mathbf{z}^{(1)} &= h^{(1)} \left(\mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)} \right) \\ \mathbf{y}(\mathbf{x}, \mathbf{W}) &= \sigma \left(\mathbf{W}^{(2)T} \mathbf{z}^{(1)} + \mathbf{b}^{(2)} \right) \end{aligned}$$

In equation (3.2) it can be seen more explicitly the relationship with the abstract form of expression (3.1) in which the composition of functions represents the layered structure. Furthermore, deeper architectures can be made simply by adding further compositions of the functions representing each layer. In general, a fully connected neural network of L layers can be expressed as

$$\begin{aligned} \mathbf{z}^{(1)} &= h^{(1)} \left(\mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)} \right) \\ \mathbf{z}^{(l)} &= h^{(l)} \left(\mathbf{W}^{(l)T} \mathbf{z}^{(l-1)} + \mathbf{b}^{(l)} \right), \quad l \in \{2, \ldots, L-1\} \\ \mathbf{y}(\mathbf{x}, \mathbf{W}) &= \sigma \left(\mathbf{W}^{(L)T} \mathbf{z}^{(L-1)} + \mathbf{b}^{(L)} \right) \end{aligned}$$

where each equation represents input layer, hidden layers and output layer respectively.

3.3.1 Activation functions

An important aspect of the neural network's architecture is that it needs to have a non-linear transformation to be able to approximate functions that can be extremely complex [35, 36].

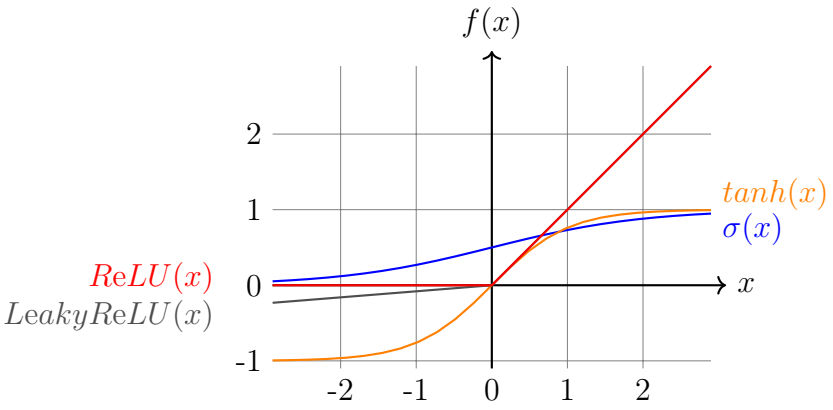


Figure 3.4: Commonly used activation functions. Sigmoid, hyperbolic tangent, ReLU and Leaky ReLU. ReLU and Leaky ReLU overlap when $x \geq 0$.

The most commonly used activation functions are the rectified linear unit (ReLU), leaky ReLU, sigmoid and hyperbolic tangent. These functions are shown in figure 3.4. In practice, the most used are the ReLU and leaky ReLU because the derivative of the sigmoid and hyperbolic tangent tends to zero on the edges, which means that the deeper the network, the closer the gradients are to zero and then it is not possible to update the weights since gradient computation is a fundamental step to optimize the weights. This is known as the vanishing gradients problem and is especially present when using deep learning models [37, 38].

These functions are defined as

$$\begin{aligned} sigmoid(x) = \sigma(x) &= \frac{1}{1+\mathrm{e}^{-x}} \\ tanh(x) &= \frac{\mathrm{e}^{x}-\mathrm{e}^{-x}}{\mathrm{e}^{x}+\mathrm{e}^{-x}} \\ ReLU(x) &= max(0,x) \\ LeakyReLU(x) &= max(\alpha x, x), \quad 0<\alpha<1 \end{aligned}$$

3.3.2 Training

In the previous sections the way in which the network computes an output from a given input has been shown. However, for the network to be useful it is necessary to tune the weights so that the model can accurately make predictions. This process of tweaking parameters is called the training phase. A dataset is usually split into at least two groups: training set and testing set. So that the model can optimize parameters by receiving the training data and then using the testing data to measure some performance metric, thus measuring how good the model is at generalization.

In the training phase, it is necessary to optimize the weights such that some error function[5]

[5]Also called loss or cost function.

is the smallest possible. That is solving the problem

$$\mathbf{W}^* = \underset{\mathbf{W}}{arg\,min}\, E(\mathbf{W}; \mathbf{X})$$

Although it is possible in theory to solve the minimization problem by calculating the gradient of $E(\cdot)$ analytically and then solving for $\nabla E(\mathbf{W}; \mathbf{X}) = 0$, it is often too computationally expensive, and iterative methods are preferred.

In order to achieve this iterative minimization, the *backpropagation* algorithm is used which efficiently computes the gradient of an error function with respect to the weights of the network for an input-output example. Since the error is calculated in the last layer, after the feedforward pass, the gradient is calculated backwards, from the output layer to the input layer, propagating the error using the chain rule at each layer. A more detailed discussion of the backpropagation algorithm can be found in *Goodfellow* and *Bishop* [39, 40].

After backpropagation it is possible to use optimization methods to find weights that minimize the error $E(\cdot)$. One way to solve this problem is to use gradient descent methods. Specifically, the Stochastic Gradient Descent (SGD) method is a common approach. In SGD, weights are initialized randomly, then each iteration tunes the weights such that the value of the error is updated in the opposite direction of the gradient, thus reducing its value.

If the weight matrix is $\mathbf{W} \in \mathbb{R}^P$, then the partial derivatives of $E(\cdot)$ are

$$\frac{\partial E}{\partial \mathbf{W}} = \left[\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_P}\right]^T$$

subsequently, the weights are updated as follows

$$w_{\mathrm{i}} \to w_{\mathrm{i}} - \alpha \frac{\partial E}{\partial w_{\mathrm{i}}}, \quad \forall \mathrm{i} \in \{1, \ldots, P\} \tag{3.3}$$

where α is the learning rate, which determines how large the steps are between iterations. A smaller value will produce a slower but more accurate algorithm. Conversely, a larger learning rate will increase speed, but minima are harder to find. The weights are updated until the error is close to zero by using some tolerance measure.

Equation (3.3) implies that a single pair (x, y) is processed by the network, this is often not optimal because datasets can have a large number of data points, so a variant of SGD is used called mini-batch gradient descent in which the data is split into many small bacthes and the gradient descent is done over the average of the gradients within the batch. Finally, the updating rule for a batch is

$$w_{\mathrm{i}} \to w_{\mathrm{i}} - \frac{\alpha}{N} \sum_{j=0}^{N} \frac{\partial E_j}{\partial w_{\mathrm{i}}}, \quad \forall \mathrm{i} \in \{1, \ldots, P\}$$

where the sum is done over all the examples in the batch. Furthermore, there are many optimizers aside from SGD which incorporate more levels of sophistication such as using momentum or decaying learning rate. Some of them are AdaGrad, Adadelta and Adam, the latter being particularly common.

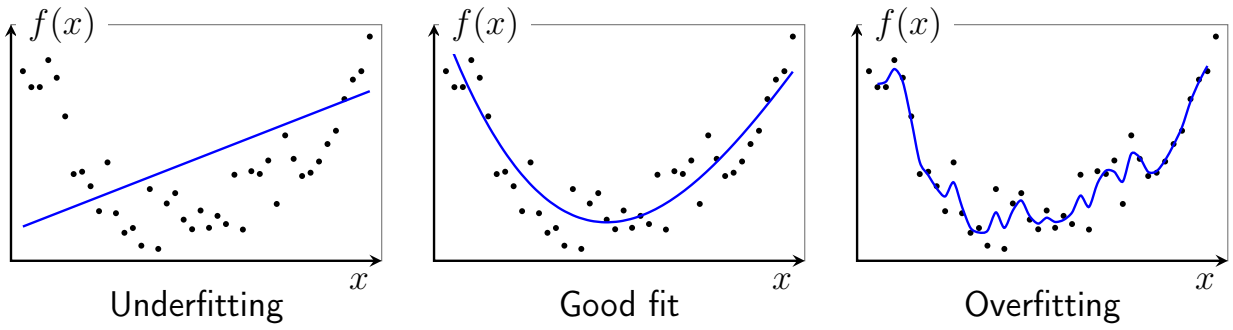


Figure 3.5: Illustration of the underfitting/overfitting phenomena. On the left, the linear model is underfitting the non linear data, the model is too simple. In the middle, a good model is able to capture the overall trend of the data while not fitting specific data points. On the right, an overly complex model is overfitting the data, passing through most of the data points.

Then, to optimize the network this process needs to be repeated for a number of epochs that yields good results. An epoch is defined as the number of times the model has processed the training data, since it is usually necessary to process the training data many times before achieving acceptable results.

3.3.3 Regularization

An important trade-off when training neural network is underfitting-overfitting. Briefly, these concepts convey the idea of how much does the network learn specifically the training data as opposed to generalizing to unseen data. Underfitting means that the network lacks the ability to perform well in the task, but has similar results both in training data as well as testing data. On the other hand, overfitting occurs when the model is too complex and it is able to have really good results in training data but at the cost of not being able to generalize, thus yielding bad results when testing. In figure 3.5 an illustration of the underfitting-overfitting phenomena is shown.

In general, overfitting is more common in deep learning because the models are usually large and contain millions of parameters. The techniques used to address these problems are referred to as regularization techniques.

A widely used technique to prevent overfitting when dealing with complex models is to use dropout, which at every step gives each neuron a probability of being "turned off". Then, the network can be complex enough but it cannot "memorize" the training examples because the weights needed to produce each output are not the same. Figure 3.6 shows how dropout works.

Furthermore, the *hyperparameters* of the network need to be optimized too. Hyperparameters are any parameters that are not the network's weights such as learning rate, number of epochs, number of layers, among many others. These are not optimized when training the network and therefore they must be set by the scientist, usually through some heuristic. A common approach to do this is to use a grid search in which different values for each hyperparameter are

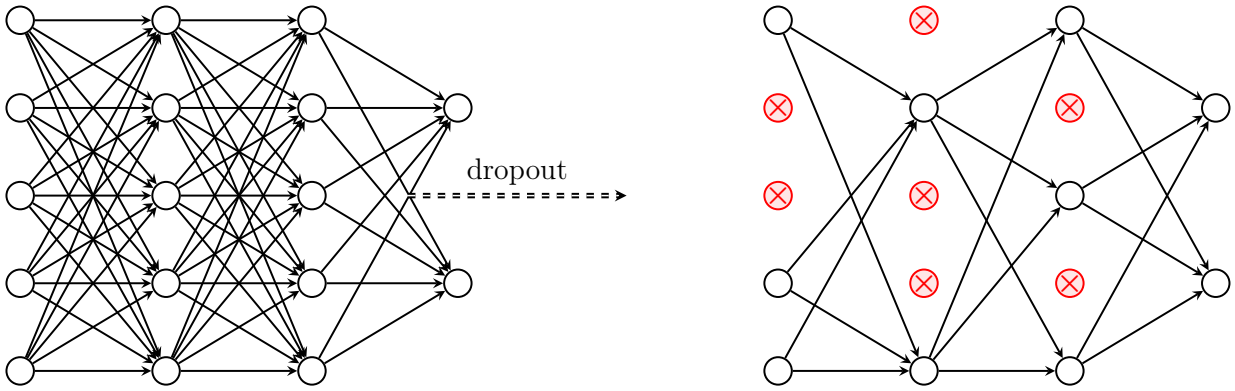


Figure 3.6: Illustration of the dropout technique with $p = 0.5$. At every forward pass of the network, neurons have a probability $p = 0.5$ of being "turned off" and no values are propagated through them.

set and then the network is trained using all of the possible hyperparameters' combinations. Then, results between each point in the grid are compared and the best one is chosen.

As stated earlier, the data used is split into training and testing where the testing portion should not be used to tune any parameter because that can cause the model to overfit the testing data instead of being able to generalize previous information to unseen examples. Since the hyperparameters need to be tuned as well as the network's weights, the training set is subdivided into training and validation, where training is to tune the network's weights while the validation set can be used to optimize hyperparameters.

Moreover, a common practice to ensure a robust model is to use *k-fold cross validation*. This is to subdivide the training set into k subsamples where one is for validation and the rest $k-1$ subsamples are for actual training. Then, the model is run over all the splits every time taking a different one for validation to finally take an average of each run. These averages can then be used to compare other models in a statistically validated way. In figure 3.7 a diagram of the k-fold cross validation process is shown.

Finally, an additional measure to prevent overfitting is the use of early stopping. Since the model has to run a number of epochs to be determined, it is possible that up to a certain number of epochs the model stops improving. Notice that a halt in improvement does not necessarily mean that the training loss does not improve, rather, it could be the case that the training set is performing better at each epoch but the validation gets worse which is a clear sign of overfitting. Hence, early stopping consists of setting a maximum number of epochs M as well as a patience parameter P, which is the number of consecutive epochs that the model will check for improvement, stopping the process if validation loss does not improve in P consecutive epochs.

3.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a type of ANN that has been used widely when dealing with grid-like data. These kinds of networks work just as a regular ANN in the sense that they receive an input, perform a matrix operation with learnable weights and biases

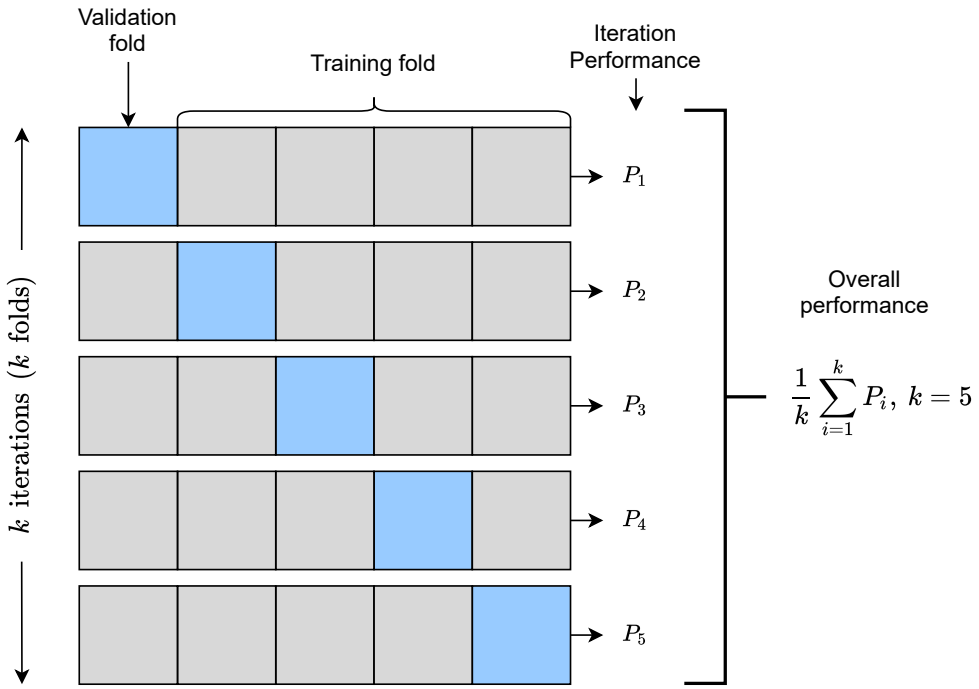


Figure 3.7: k-fold cross validation diagram for $k = 5$ folds.

and then apply a non-linear activation. The difference lies in that CNN make use of the spatial relationships of the input which would be lost if the input were flattened to use a fully connected layer.

In figure 3.8 a typical CNN architecture is shown. Three distinctive parts can be identified:

- **Convolutional layer:** This layer performs a convolution operation between the input and a kernel[6] with weights that the network can learn, these kernels have a width and height that define the window of the convolution and depth which defines the number of different outputs of the operation. The transformed data from the convolution process is called a feature map. Specifically, figure 3.10 shows that a number of kernels is defined, as one would define the number of neurons in a fully connected layer. These kernels have a fixed size and slide through the input image or feature map and perform a convolution operation. After the convolution is done, an activation function is applied such as a ReLU, as in a regular ANN.

  The one dimensional convolution is defined as

  $$s(t) = (x * w)(t) = \int x(\tau)w(t-\tau)\mathrm{d}\tau$$

  In the context of CNN, $s(t)$ is the resulting feature map, the input is x and the kernel is w. In practice, the input is a multidimensional array of data points, so a discrete convolution is used

  $$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

[6] Also called filter.

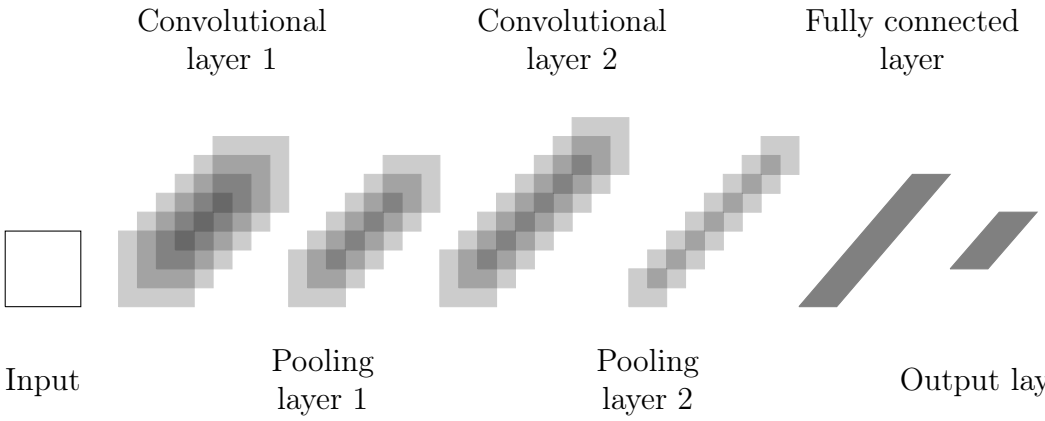


Figure 3.8: Diagram of a CNN. The input is expected to have a grid-like structure. The architecture consists of two convolutional blocks which include a convolutional and pooling layer. After the subsequent convolutional blocks, a fully connected layer is used to finally produce the output. Source: `https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/` [41].

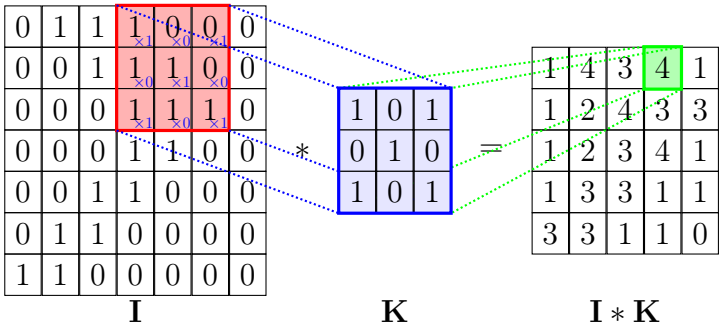


Figure 3.9: Two dimensional discrete convolution example where I is the input, K the kernel and $I * K$ is the convolution that yields the resulting feature map. Source: `https://github.com/PetarV-/TikZ`.

It is assumed that these functions are zero in all their domain except for the actual values of the data, so the infinite sum can be computed. Furthermore, it is common to process image-like data and kernels, so a 2D version is defined as

$$S(\mathrm{i}, j) = (I * K)(\mathrm{i}, j) = \sum_m \sum_n I(\mathrm{i} + m, j + n) K(m, n)$$

This is called *cross-correlation* and is equivalent to the convolution operation except that it is non-commutative, $S(\mathrm{i}, j) = (I * K)(\mathrm{i}, j) \neq (K * I)(\mathrm{i}, j)$, but this is not an issue when implementing the network as long as the application is consistent. Figure 3.9 shows how this works in practice for a $2D$ input and a 3×3 kernel.

In practice, it is necessary to define some parameters to apply a convolutional layer, namely the kernel size, strides and padding. The strides are the amount of pixels the kernel moves at each step while the padding defines the behavior in the borders, when part of the kernel lies outside of the input. In figures 3.11 and 3.12 examples

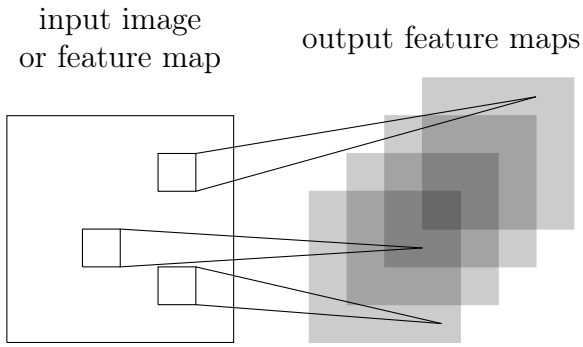


Figure 3.10: Illustration of a convolutional layer. A number of filters with learnable parameters are applied to the input or previous feature map by using the filters as a sliding window and computing the dot product at each step. Source: `https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/` [41].

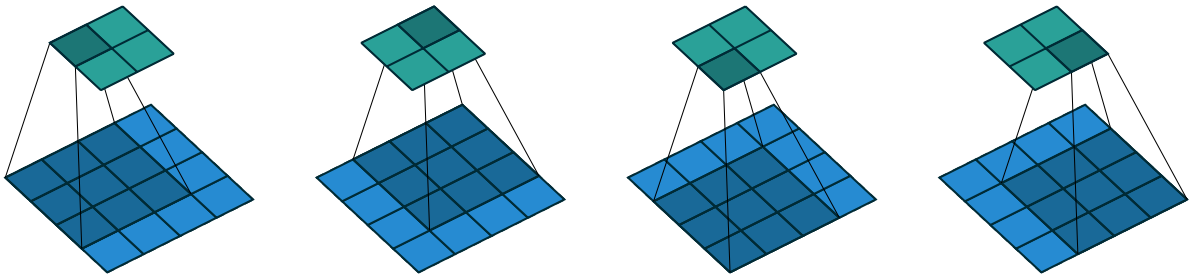


Figure 3.11: Convolving a 3×3 kernel over a 4×4 input using unit strides and no padding. Source: `https://github.com/vdumoulin/conv_arithmetic` [42].

with specific input, kernel, strides and padding are shown. Notice that all of these parameters will affect the shape of the output.

Furthermore, it is useful to define the input size (i_j), kernel size (k_j), stride size (s_j) and padding size (p_j) where each parameter is considered along axis j[7]. Then, a general expression [42] for the output shape is

$$o_j = \left\lfloor \frac{\mathrm{i}_j + 2p_j - k_j}{s_j}, \right\rfloor + 1 \quad \forall \mathrm{i}, k, s, p \in \mathbb{N}$$

- **Pooling layer:** The pooling layer downsamples the previously convolved input, i.e. feature maps, thus reducing the dimension of the data. A commonly used type of pooling layer is the max pooling, in which a window slides through the data and takes the maximum value within the window. The way it works is very similar to a convolution in the sense that a kernel size is defined, and this kernel scans the input and applies a transformation at each step. The difference is that the transformation is not a convolution, but a downsample through the use of some statistical feature within the window (average, maximum, etc). A diagram of a pooling layer is shown in 3.13. Similar to the convolution, the kernel size and strides affect the size of the downsampled output.

[7] In the case of square input, strides and padding this distinction is not necessary.

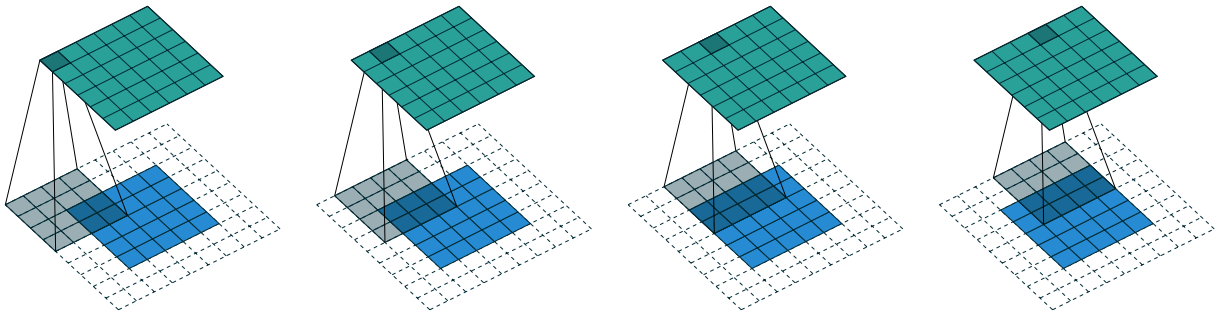


Figure 3.12: Convolving a 4×4 kernel over a 5×5 input padded with a 2×2 border of zeros using unit strides. Source: `https://github.com/vdumoulin/conv_arithmetic` [42].

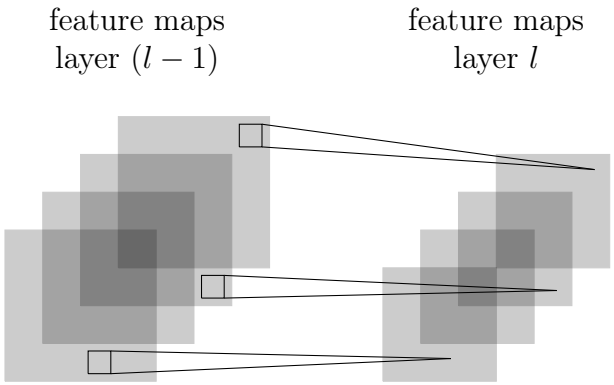


Figure 3.13: Illustration of a pooling layer. Layer l summarizes the information in layer $l-1$ through some statistical feature (maximum, average, etc). This is accomplished by defining a fixed size window and then scanning the previous layer, applying the pooling operation at each step. The number of feature maps remains the same. Source: `https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/` [41].

An expression for the output size is[8]

$$o = \left\lfloor \frac{\mathrm{i} - k}{s} + 1, \right\rfloor + 1 \quad \forall \mathrm{i}, k, s \in \mathbb{N}$$

- **Fully connected layer:** A feed forward network is used to produce the final output. Since the convolutional and pooling layers already have done the work of extracting relevant spatial features, the fully connected layers can then receive the transformed data.

Furthermore, CNN have properties that benefit their application [43]:

- **Sparse interactions:** The number of parameters is reduced compared to fully connected layers since kernels are usually much smaller than the input and this is more efficient in terms of memory usage.
- **Shared weights:** Weights are used multiple times in the model because kernels slide through the input while dense layers have each weight associated with only one feature.

[8] Axis subindex is omitted for simplicity.

- **Equivariance to translation:** The representation of the input's features is equivariant to the location of features, that is, if the input is translated, the feature maps in each layer will be translated in the same way [39].

3.5 Segmentation models

An important problem in computer vision is the analysis of images or videos for segmentation and it has been addressed using different techniques, including deep learning models which are discussed in this section.

First, the segmentation task encompasses several different but related problems. The most common are:

- **Object localization:** An image with different subjects is processed and the model is expected to localize them, this is usually done using bounding boxes around every subject.
- **Semantic segmentation:** In this case, each pixel of the image is labeled as a subject, so the specific geometry is recognized as opposed to the bounding box localization.
- **Instance segmentation:** This expands the semantic segmentation by assigning labels to each instance of a subject within an image. This allows to separate subjects that would have the same label in semantic segmentation.

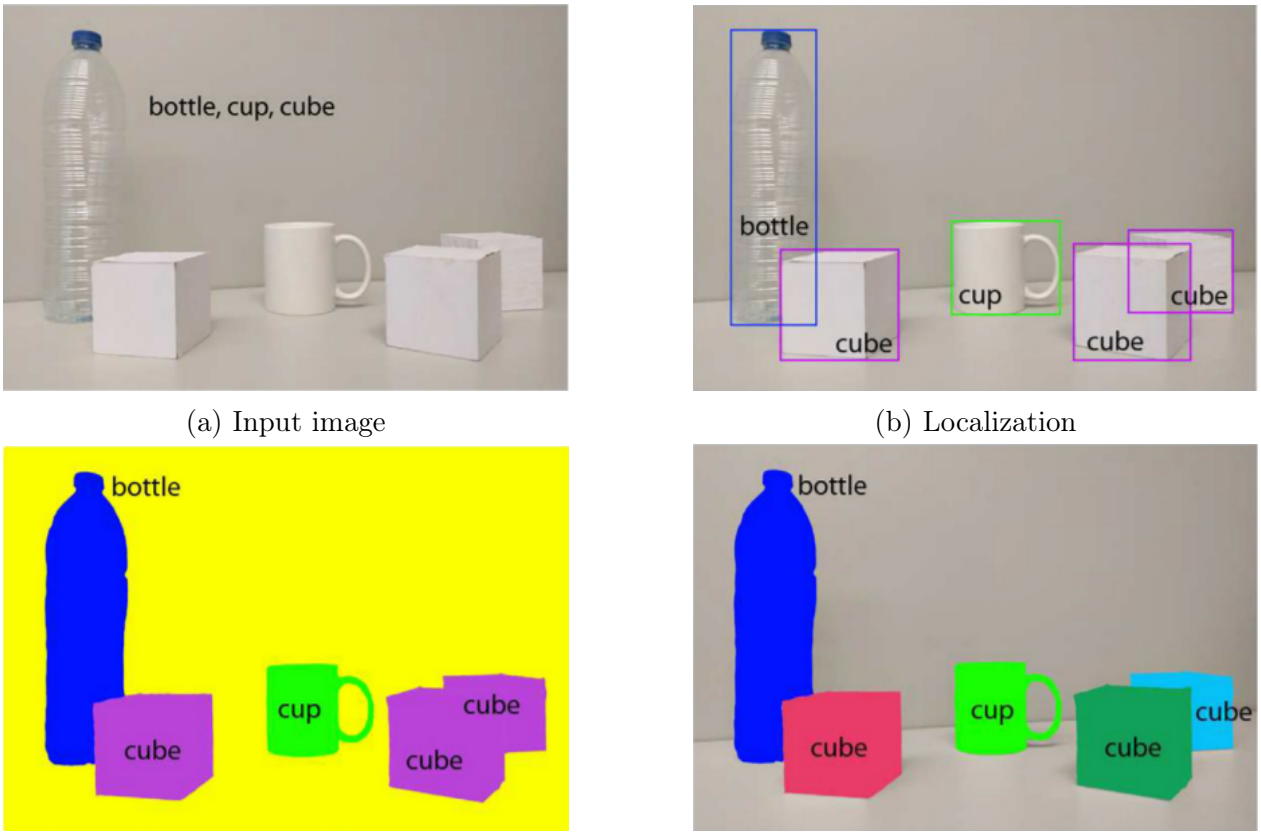
In figure 3.14 the different types of segmentation are depicted. This thesis is concerned with the application of supervised semantic segmentation models, so the models studied aim to solve that task.

3.5.1 Fully convolutional networks

Fully Convolutional Networks (FCN) are ANN that only make use of convolutional and pooling layers. This type of architecture is suitable for semantic segmentation because it is possible to have a grid-like output, as opposed to a single value or array of values when fully connected layers are used in the output layer. As stated before, in semantic segmentation it is necessary to assign a label to each pixel in an image to generate groups of pixels that have some semantic or spatial relationship.

A generic FCN architecture is shown in figure 3.15. The structure is composed of a contracting convolutional path in which subsequent convolutional and pooling layers are applied so the number of feature maps increases (depth) and the size of the input decreases (width and height). Then there is an expanding path in which transposed convolutions[9] are applied and followed by regular convolutions that reduce the number of feature maps. The purpose of transposed convolutions is to upsample the feature maps so the original input shape is recovered. The output is the segmentation map which is an image of the same shape as the input and each pixel has a specific label. In a supervised approach, these labels come from whatever labels have been assigned to the training examples.

[9]The term "deconvolution" is sometimes used, but this is discouraged because a deconvolution is defined as the inverse of a convolution, which is different from a transposed convolution [42].



(a) Input image

(b) Localization

(c) Semantic segmentation

(d) Instance segmentation

Figure 3.14: Typical segmentation tasks. Source: *A survey on deep learning techniques for image and video semantic segmentation* [44].

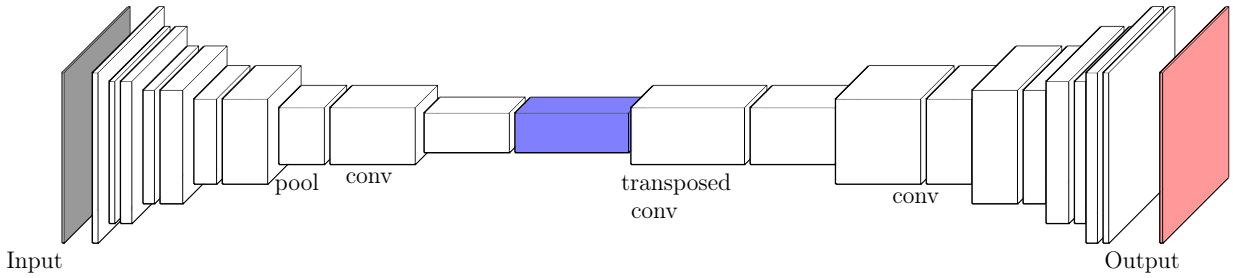


Figure 3.15: Illustration of a FCN. First, subsequent convolution and pooling layers are applied. In the convolutions the feature maps are added (more length in the figure) while maintaining height and width, then the pooling layer reduces the size while keeping the same number of feature maps. In the expanding path, a transposed convolution is applied which upsamples the feature maps, then a normal convolution is applied reducing the number of feature maps. Source: Adapted from `https://github.com/jettan/tikz_cnn`.

The main feature of a FCN is the upsampling layer in which the transposed convolution is applied, because without such an operation, it would no be possible to recover the shape of the input. This could be solved by not using pooling layers and maintaining the shape throughout the network, but this would be computationally expensive because the number of parameters would be really large, and one of the benefits of CNN is the possibility to reduce the number of parameters by using small filters compared to the input size.

3.5.2 Transposed convolution

The transposed convolution operation functions in the opposite way of a pooling layer, thus upsampling the input. For this, a kernel with trainable parameters is used. Note that since this is not the inverse of a convolution, the result is just a new collection of feature maps that recovers the shape of the previous layer, but does not have the same parameter values.

As an example, let us consider the convolution shown in figure 3.11. If we write the convolution as a vector-matrix multiplication, the kernel can be written as

$$\mathbf{C} = \begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

where w_{ij} is the weight of the kernel located in row i and column j. Then, the convolution can be computed by flattening the input so that it becomes a $16-$dimensional vector. The result is going to be a $4-$dimensional vector which can be reshaped to 2×2 to get the matrix form. With this in mind, notice that in the forward pass, the convolution consists in multiplying by $\mathbf{C}$ while in the backward pass to propagate the error it is necessary to multiply by $\mathbf{C}^T$. Hence, the weight matrix defines both the forward and backward pass of the network.

Now consider the idea behind transposed convolution which is to take a small matrix, applying a transformation and obtaining a larger matrix maintaining the number of feature maps. This is precisely what happens when in the backward pass $\mathbf{C}^T$ is applied. Then, multiplying by $\mathbf{C}^T$ is a transposed convolution by definition if $\mathbf{C}^T$ is applied in the forward pass instead of the backward pass.

In general, the process can be thought of as having an input matrix in which every data point is multiplied by the kernel and then summed over the corresponding values. This is equivalent[10] to a regular convolution over a fully padded input[11] as shown in figure 3.16 which is a 2×2 input in which a transposed convolution is applied with a 3×3 kernel to produce a 4×4 output or equivalently, a 6×6 (padded) input convolved with a 3×3 kernel.

3.5.3 Residual connections

As of now, the networks discussed have all been connected using adjacent layers which means every node in one layer is connected with every node in the next layer, no connections are

[10]This is useful as a concept, but is not used when programming because many of the operations are useless zero multiplications.

[11]This means the input is padded such that the kernel can overlap its last value (bottom-right) with the first of the input (top-left).

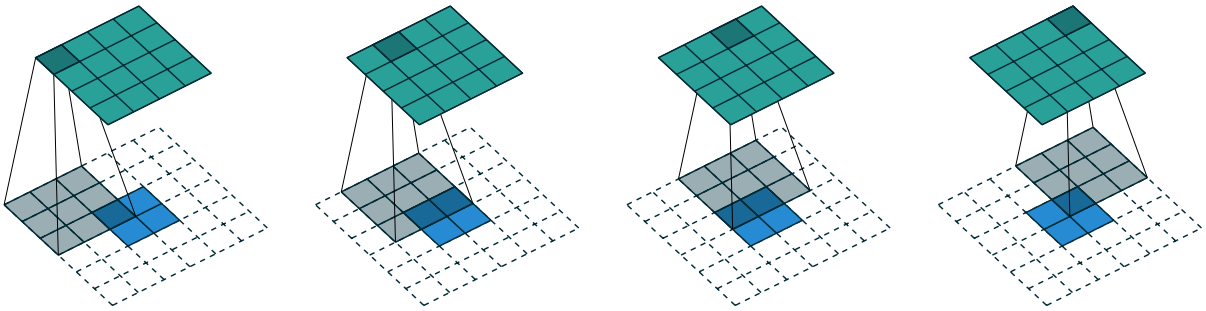


Figure 3.16: Transposed convolution of a 2×2 input with a 3×3 kernel to produce a 4×4 output. This is equivalent to a regular convolution of a 6×6 input with a 2×2 padding with the same kernel. Source: `https://github.com/vdumoulin/conv_arithmetic` [42].

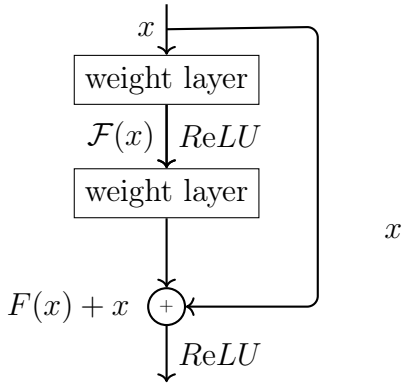


Figure 3.17: Residual connection diagram. Source: Adapted from *Deep residual learning for image recognition* [45].

missing and no connections with other layers are added. Nonetheless, a trick that has proven to be very effective when training ANN is the use of residual connections, allowing to achieve state of the art results in a wide range of tasks. These connections consist in having a connection with deeper layers of the network, so a neuron n in layer l can be connected with layer $l + 1$ as usual and with other layers $l + r$, $r \geq 1$.

Before residual connections were developed, some deep networks performed worse than shallow ones despite the fact that in theory neural networks are universal approximators [35, 36], which means that it is possible to approximate any function in a compact domain with an arbitrary small error given enough neural units. Particularly, there was a saturation in accuracy, that is the accuracy improved with depth up to a certain point at which accuracy began to decrease. This is called the degradation problem [45].

Specifically when a shallow network performs better than a deep one, an intuitive approach would be to allow the deep network to skip some layers so that it can mimic the behavior of a shallow network if needed. Then, if a layer activation is small, an identity mapping can be added so that the gradient does not vanish and the network can make use of that layer's level of abstraction. In figure 3.17 a residual connection is shown.

In the context of FCN, these residual connections are called *skip layers*, which also skip

connections to further layers in the network, but do so by concatenating layers as opposed to the sum shown in 3.17. On a semantic level, what the network does when convolving and pooling is extracting more high-level representations of the input. Then, what the skip layers do is recover the layers' low-level information which is the context needed to reconstruct when upsampling [3].

Chapter 4

Proposed approach using Fully Convolutional Networks

In this chapter the proposed models are described. These consist in FCN based deep learning architectures which are compared in the same datasets to ensure validated results and also to choose the best performing model for further analysis. The architectures are described using baseline parameters that will then be optimized using a grid search. In figure 4.1 a flowchart of the proposed framework is shown.

4.1 Segmentation architectures

The architectures used are FCN based, so they share they building blocks shown in figure 3.15. There is a contracting path of successive convolution and pooling layers and then an expanding path that mirrors the shape of the contracting one by applying transposed convolutions (upsampling) instead of pooling. Also, in the contracting path the number of feature maps increases while in the expanding path it decreases until reaching the output's number of channels. The segmentation architectures used are:

- **DeconvNet:** This network is an FCN which uses a VGGNet [46] as a backbone for the contracting path. The expanding path mirrors the VGGNet architecture. Since it is based in a network that has already proven to have good results with image data it is appropriate to extend its application [46, 24]. In figure 4.3 the contracting path is shown.
- **U-Net:** The U-Net is an FCN that was first implemented in the biomedical field for microscopy images. The architecture is shown in figure 4.2 and makes use of skip layers which are taken from the contracting path and then concatenated in the expanding path layers. This allows the network to localize specific data by reducing dimensionality and also have the context of the general image through the skip layers [3].
- **MultiResUnet:** This is an improvement of the U-Net, than can process subjects which have large differences in scale within an image. This is done with several convolutions with different kernels so that the features are learned at different resolutions, this is borrowed from Inception architectures such as GoogLeNet [47]. The structure is the same as in figure 4.2 but the convolution blocks are changed with residual blocks that apply different convolutions which are then summed. Also, skip connections have a residual residual path in between. Figure 4.4 shows the residual block used and figure 4.5 depicts the residual connections used in the skip layers.

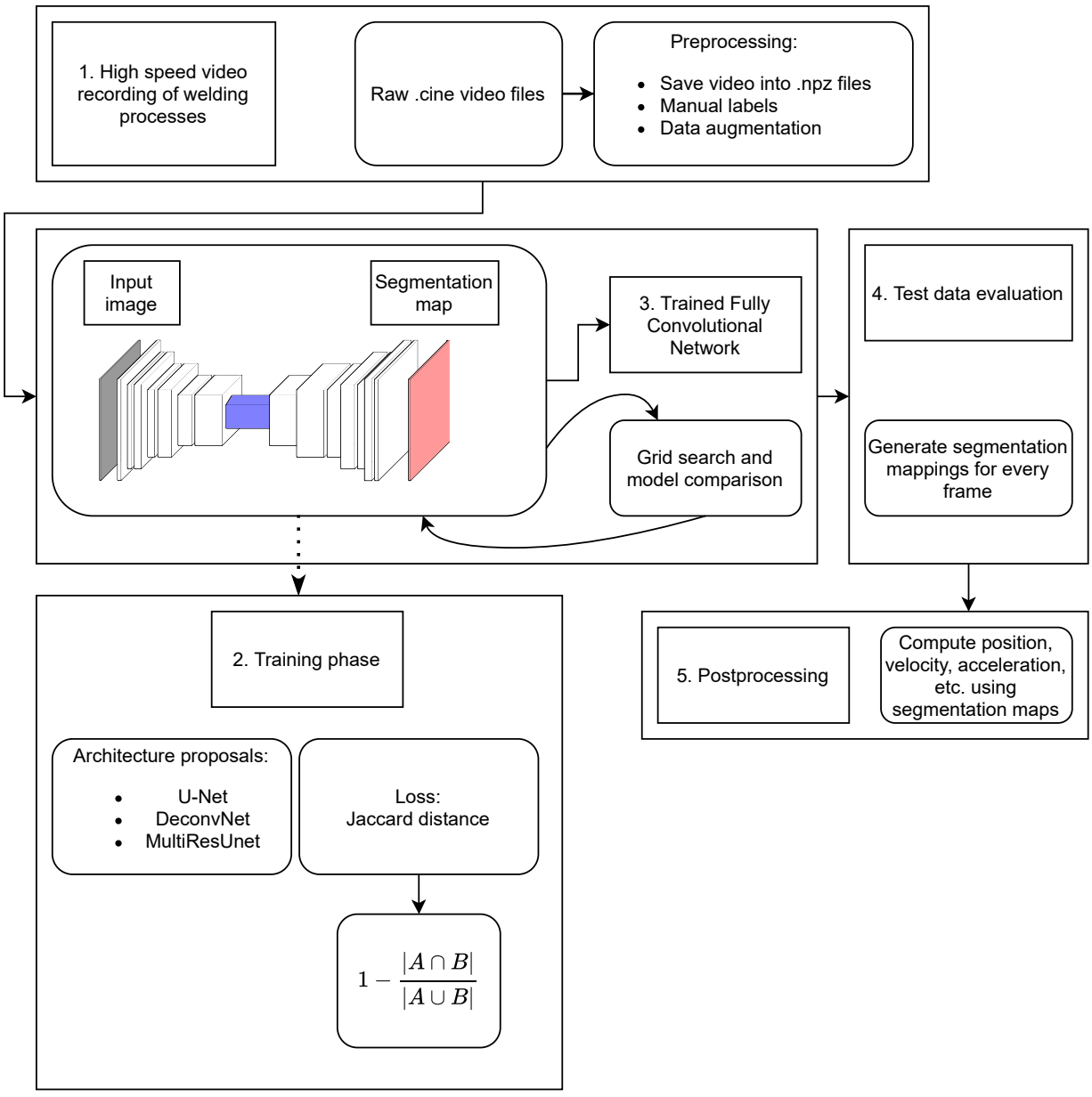


Figure 4.1: Proposed framework flowchart.

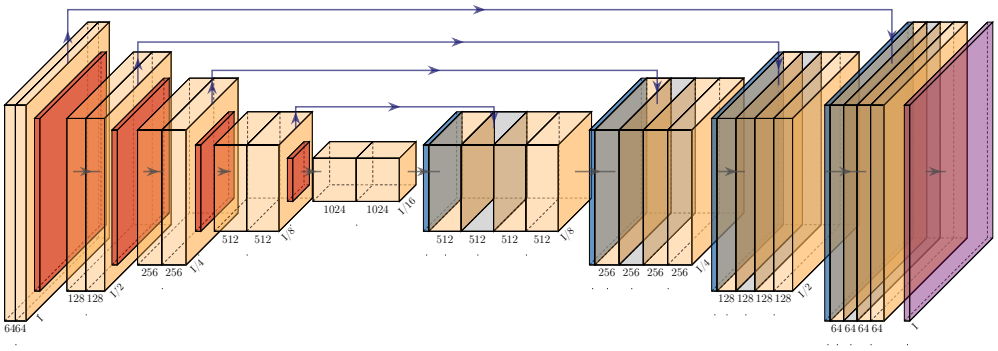


Figure 4.2: Illustration of a U-Net architecture. The architecture is divided in a contracting path and an expanding path. The contracting path has conv (yellow)/conv/pool (red) blocks. The expanding path has transposed-conv (blue)/conv/conv blocks. Dimensions in the expanding path mirror the contracting path. Each convolution before pooling is used as a skip layer and concatenated with the corresponding shape in the expanding path. The final layer reduces the feature maps to one and applies a softmax function for pixel-wise classification. Convolution kernels are 3×3 while pooling kernels are 2×2 so the size is halved at each block. Source: Adapted from `https://github.com/HarisIqbal88/PlotNeuralNet`.

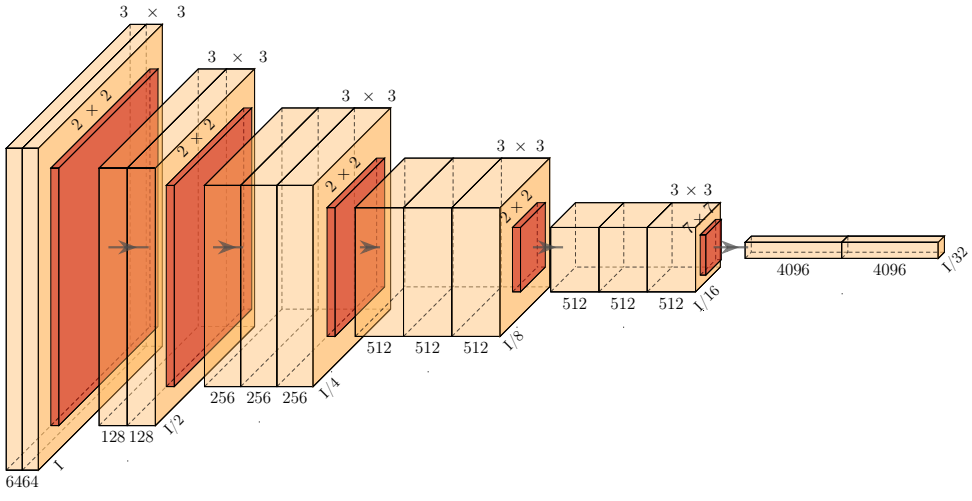


Figure 4.3: Illustration of a DeconvNet contracting path. A VGGNet architecture is used and then mirrored in the expanding path. At the top of each block the kernel shape is shown, below are depicted the number of feature maps per convolution. Finally, I is the input size which is halved after each pooling layer. Source: Adapted from `https://github.com/HarisIqbal88/PlotNeuralNet`.

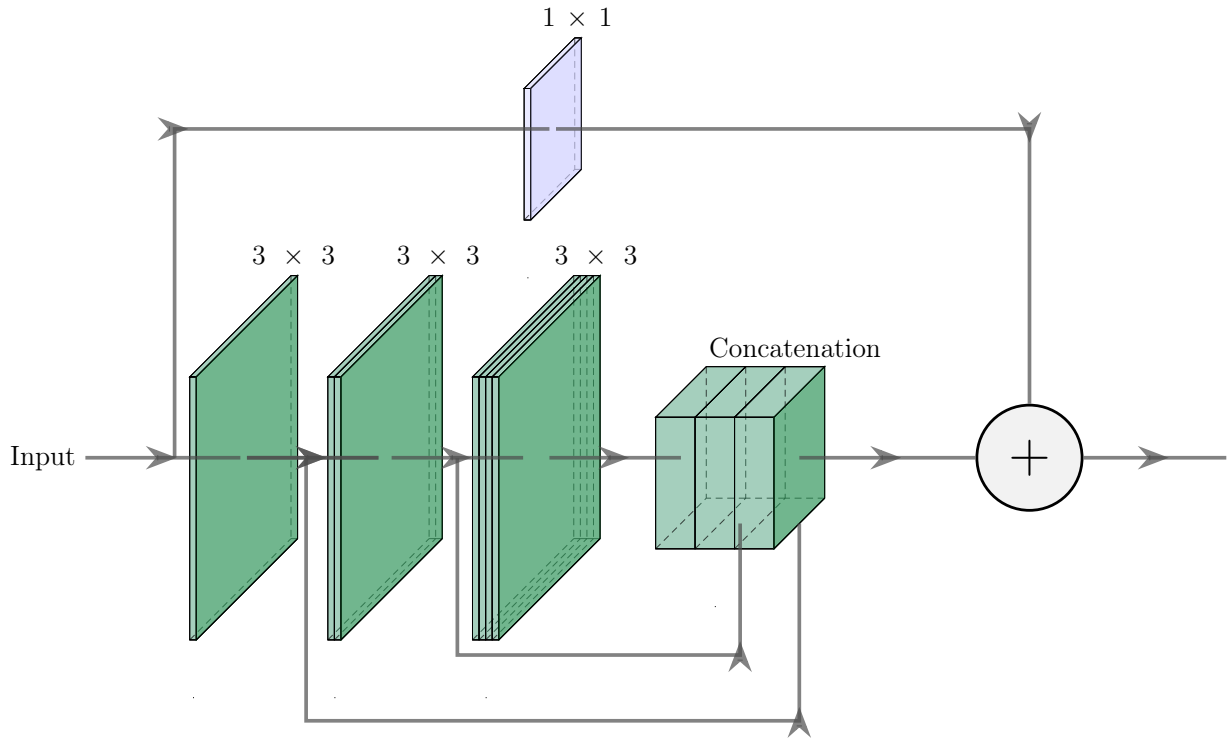


Figure 4.4: Illustration of a Multi Resolution block. The block consists of successive convolutions which are then stacked via skip layers and then summed using a residual connection. The second and third convolutional blocks are stacked convolutions of 3×3 kernels which are factorized operations of 5×5 and 7×7 convolutions respectively [48]. This approach is more computationally efficient than using parallel higher resolution kernels. Source: Adapted from `https://github.com/HarisIqbal88/PlotNeuralNet`.

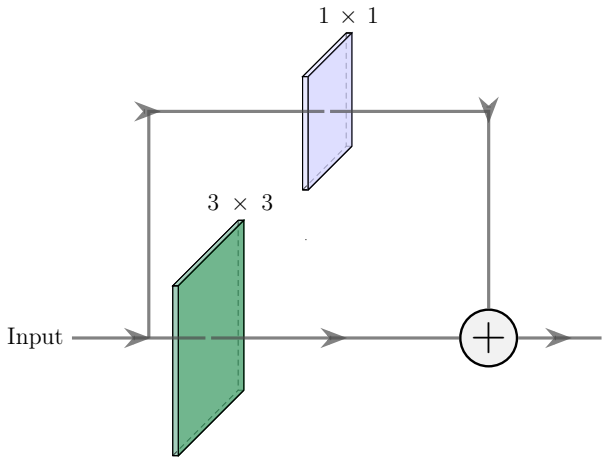


Figure 4.5: Illustration of a residual path. This type of block is used in series in the MultiResUnet in between skip layers, so the skipped features are processed before being concatenated to the expanding path. Source: Adapted from `https://github.com/HarisIqbal88/PlotNeuralNet`.

4.2 Segmentation loss function

As stated earlier, to train a supervised neural network it is necessary to have some measure of error so that it can be minimized with respect to the weights. This measure is the loss function which depends on the task. In the case of semantic segmentation, it is necessary to use a loss function that can capture pixel-wise comparisons. Specifically, for supervised segmentation, the input is an image, the output is a 2D mapping of each pixel of the image to a certain class (e.g. pixel= 0 if background and pixel= 1 if foreground) and the output is compared through the loss function with a previously labeled image with the same kind of segmentation mapping corresponding to the input image. A widely used loss metric for supervised semantic segmentation is the intersection over union metric, also known as Jaccard index. This metric is given by

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where $|\cdot|$ is the cardinality of the set. Hence, if the overlap is perfect $|A \cap B| = |A \cup B| \rightarrow J = 1$. Conversely, if there is no overlap $|A \cap B| = 0 \rightarrow J = 0$. In a segmentation mapping, $|A \cap B|$ can be computed as the element-wise multiplication of A and B, and then sum the result. Furthermore, $|\cdot|$ is computed as the sum of the matrix' elements. The score is computed over all classes and then averaged.

In order to minimize a loss function using the Jaccard index, the Jaccard distance is used and is given by

$$\mathrm{d}_j(A, B) = 1 - J(A, B)$$

4.3 Validation

The proposed architectures have to be validated to ensure that the best results are obtained. Then, every model is subject to a grid search. The parameters used for grid search are shown in table 4.1. Also, each model (i.e. each point in the grid) is run using k-fold cross validation with $k = 4$. That is, each model is run four times with a different and disjoint 25% of validation data each time. The average loss in the validation set is used to compare the models' performances and choose the best one.

Table 4.1: Hyperparameters used for grid search.

Dataset	(Globular, Spray)
Epochs	Early stopping Max: 200 epochs Patience: 20 epochs
Loss function	Jaccard distance
Batch size	(8, 16, 32)

Number of filters	(8, 16, 32)
Learning rate	(.01, .005, .001)
Optimizer	Adam
Architecture	(U-Net, DeconvNet, MultiResUnet)

Chapter 5

Dataset

In the following chapter the case studies considered in this thesis are detailed, specifically the datasets used and the preprocessing that later leads to the input of the proposed model.

5.1 Video files

The study case consists in analyzing high speed video data from GMAW processes, namely globular and spray transfer. The videos are raw `.cine` files which can be inspected using the *Phantom Camera Control* (PCC) software. Since the processes are characterized by their transfer mode, one video for each is used. In tables 5.1 and 5.2 the parameters of each video are shown.

After inspecting the metadata of the videos in PCC, they are read in `python` using the `pycine` library[1]. Then, each video is stored into an `.npz` file. In figures 5.1 and 5.2 frames of each transfer mode can be seen, particularly the main characteristic of larger droplets in globular and smaller droplets in spray[2].

Notice that later, to calculate physical properties of the process, there has to be a relationship between the pixel and real distance. This can be done using the wire's diameter, since that is a fixed length that is always visible in the image. Using that relationship, the pixel to distance conversion is

$$\begin{aligned} \text{Wire's diameter} = 0.045\ in = 26\ px = 1.143\ mm \\ 1\ px = 0.0439615\ mm = 4.39615 \cdot 10^{-5}\ m \end{aligned} \tag{5.1}$$

5.2 Labeling

In order to train a supervised deep learning model, manual labels are made using the images as input. This is done using the LabelBox platform which allows to make such segmentation and then download the pairs of input image and corresponding mask. A number of 150 images were segmented for each transfer mode. Some of the resulting masks are shown in figures 5.3 and 5.4 for globular and transfer respectively.

[1] `https://github.com/OTTOMATIC-IO/pycine`

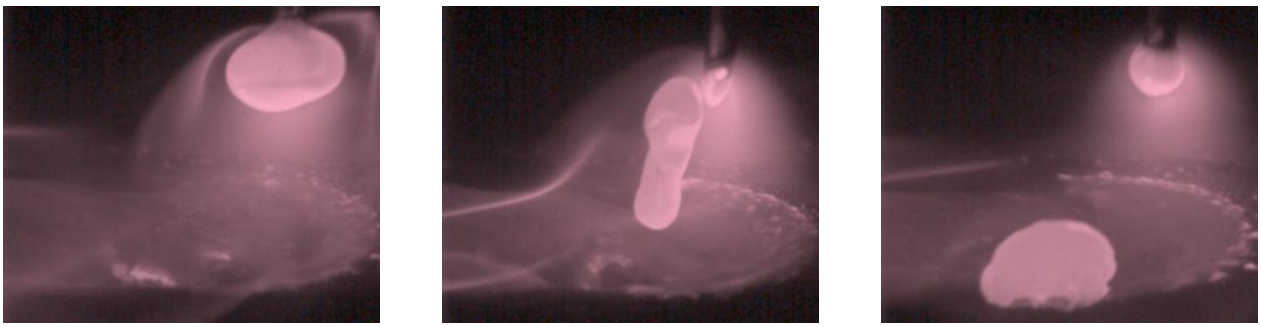
[2] Similar videos found at `https://sites.ualberta.ca/~ccwj/videos/pages/Intro%20High%20Speed/`

Table 5.1: Globular transfer video parameters.

Electrode	$ER70S-6$	Shielding gas	85% Ar, 15% CO_2
Size	$0.045''$ (1.2 mm)	Number of frames	9947
Wire feed speed	175 ipm (4.4 m/min)	Resolution	352×296
Voltage	33 V	Frame rate	3000
Travel speed	20 ipm (0.5 m/min)	Period	333.3 μs
CTWD	$0.7''$ (18 mm)	File size	2.01 Gb

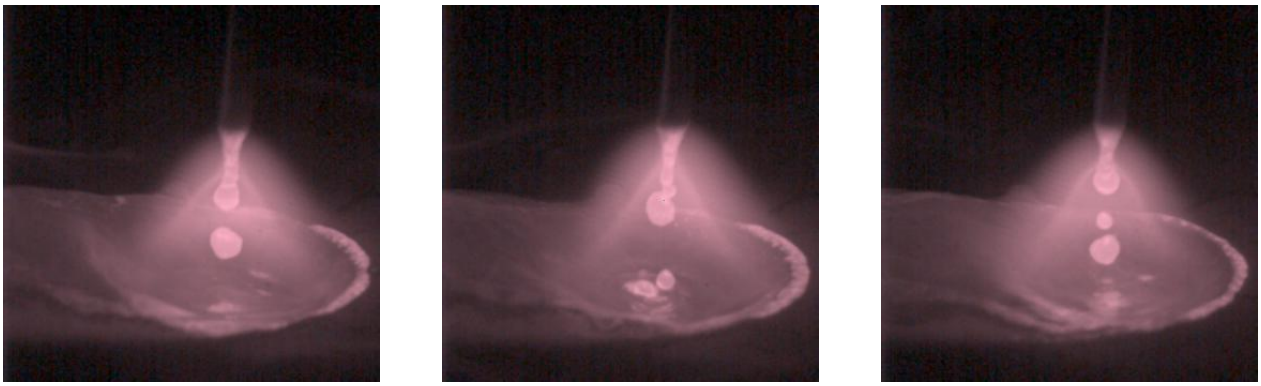
Table 5.2: Spray transfer video parameters.

Electrode	$ER70S-6$	Shielding gas	85% Ar, 15% CO_2
Size	$0.045''$ (1.2 mm)	Number of frames	6691
Wire feed speed	400 ipm (10 m/min)	Resolution	352×352
Voltage	35 V	Frame rate	3000
Travel speed	20 ipm (0.5 m/min)	Period	333.3 μs
CTWD	$0.7''$ (18 mm)	File size	1.61 Gb



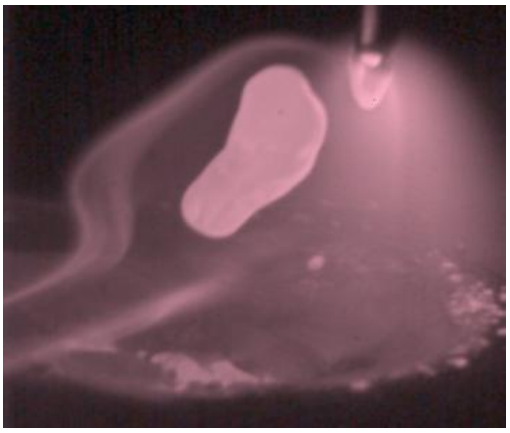
(a) (b) (c)

Figure 5.1: Globular transfer mode frames.



(a) (b) (c)

Figure 5.2: Spray transfer mode frames.

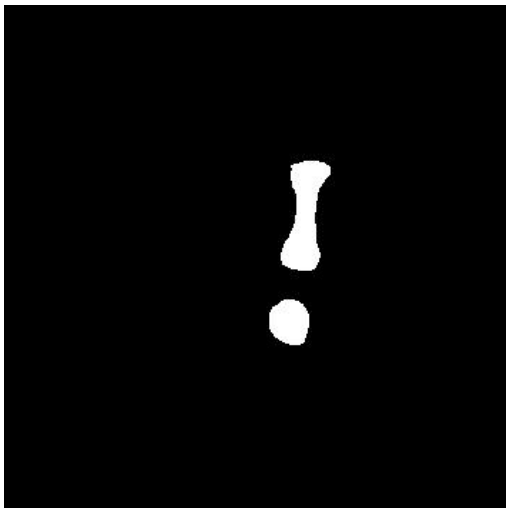


(a) Original frame

(b) Manual segmentation mask

Figure 5.3: Examples of manually segmented frames using LabelBox for globular transfer.

(a) Original frame



(b) Manual segmentation mask

Figure 5.4: Examples of manually segmented frames using LabelBox for spray transfer.

The selection of the training images is not random, frames are chosen to have as much of the variety of the process as possible, that is the droplet formation, growth and release as well as some deviations such as irregular droplet shape (see figure 5.1b), horizontal droplet movement (as opposed to vertical), among others. Since most of the frames are from the growth process because it takes more time, it would be counterproductive to choose random frames because one would run into the risk of choosing most, if not all, images from the growth process, inadvertently ignoring the rest.

Notice that although the images have color, it is mostly a gradient from white to black through different intensities of magenta so the images are used as grayscale because the same information can be captured with a third of file size.

5.3 Data augmentation

A total of 150 images were labeled for each transfer mode. Since the number of images is low, data augmentation is used applying several alterations to the original images and labels. Each example is augmented such that it produces 5 different training examples. Therefore, the augmented dataset has 750 images for each transfer mode. In figure 5.5[3] an example of an augmented input is shown and in table 5.3 a summary of the dataset information is shown.

The augmentations were made in `python` using the library `imgaug` and are listed as follows:

- Pixel dropout between 0% and 5%, which means a probability is sampled uniformly such that $p \in [0, 0.05]$ for each augmented image so that every pixel has a probability p of being turned to zero. In practice, every image will lose random pixels up to a maximum of 5% of the whole image.
- Rotations between $-45°$ and $45°$.
- Elastic transformations with $\alpha \in [20, 50]$ and $\sigma \in [4, 5]$ sampled uniformly from their respective intervals for each augmented image, where α is the strength of the distortion field. Higher values mean that pixels are moved further with respect to the distortion field's direction. σ is the standard deviation of the Gaussian kernel used to smooth the distortion fields. This gives the images a ripple effect.
- Additive Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma \in [0, 15]$ sampled uniformly for each augmented image.

Notice in figure 5.5 that the sampling of parameters within intervals for each image makes it so that some images are slightly changed while others are more heavily augmented. Also, the labels are only affected by the transformations that do not affect pixel intensity, such as elastic transformations and flips, and they are not affected by pixel dropping or Gaussian noise.

[3]Image 5.5 has different colors compared to images in previous figures, this is due to the color map used when plotting the grayscale image in `python`. Nonetheless, pixel values are the same.

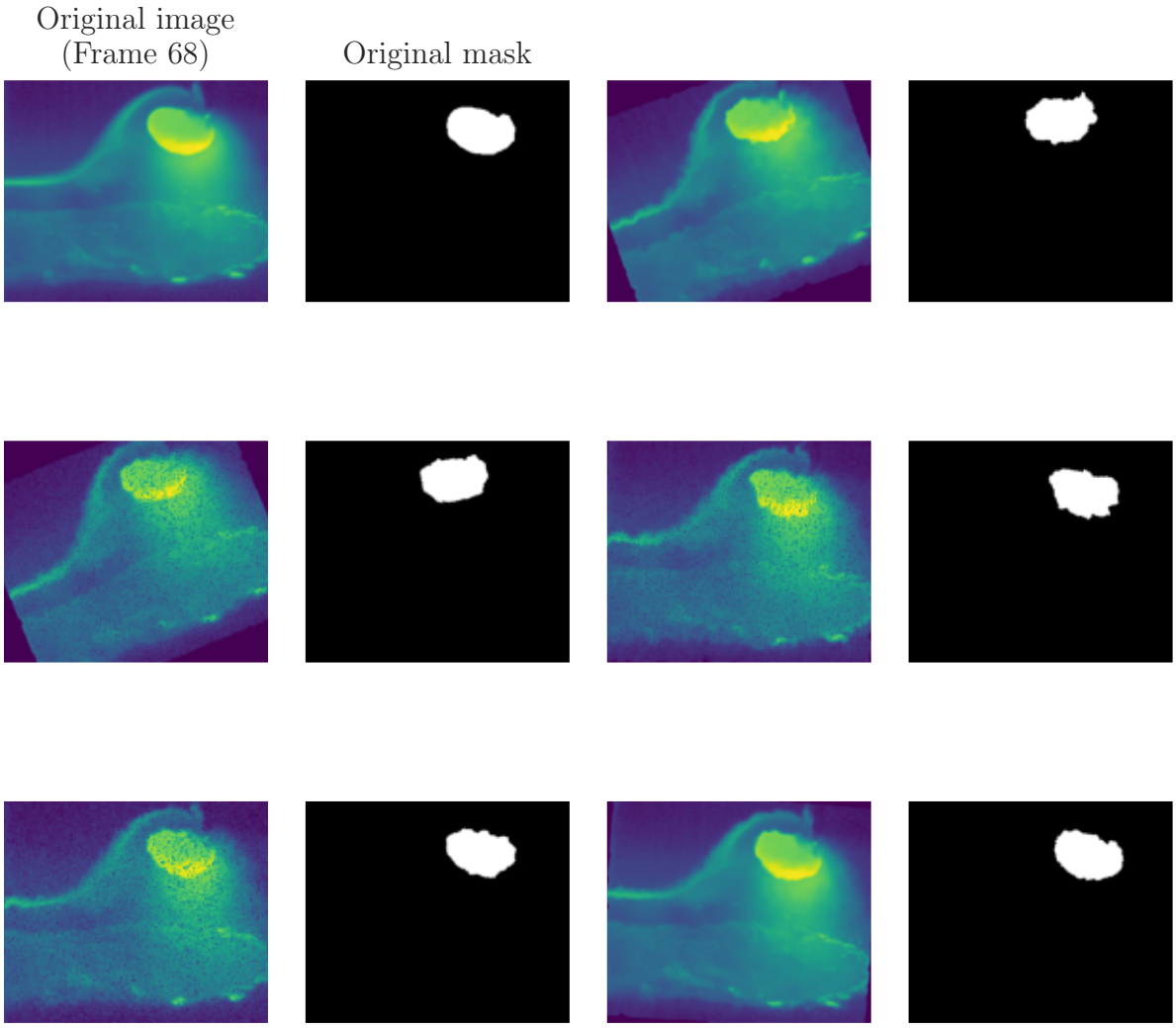


Figure 5.5: Sample of an augmented image and respective mask (top left) using pixel dropout, rotation, elastic transformation and additive Gaussian noise. All the effects are applied simultaneously.

Table 5.3: Dataset summary per transfer mode.

Manually segmented images	150
Augmented images	750
Shape	352×288 (globular) 352×352 (spray)
Channels	1 (grayscale)

Chapter 6

Results and analysis

In this chapter, the results are shown and discussed covering the model selection process, training, testing and post processing. It is worth mentioning that since this thesis is focused on analyzing videos, there is a great value in having video results as well but because of the format, only sampled images can be shown. Nevertheless, in the GitHub project[1] are all the video results which generate the images shown in this chapter. These are specially useful for analyzing the post processing results.

6.1 Grid search

The three proposed architectures' hyperparameters are optimized through grid search (see table 4.1). Also, 4-fold cross validation is performed so that every model is run over four different and disjoint combinations of training and validation set. Therefore, every model yields four training loss and validation loss values which can then be averaged and used to compare between models.

In tables 6.1 and 6.2 the three best models for each proposed architecture are shown for globular and spray transfer respectively[2]. The results show that the best model is the U-Net in every case, while the MultiResUnet has the worst performance and the DeconvNet is between the other two, closer to the MultiResUnet. Hence the U-Net architecture seems suitable to solve the task at hand.

The low performance of the MultiResUnet can be explained by the fact that it is considerably more complex than the alternatives and the problem of droplet detection is relatively simple if compared to other computer vision tasks such as traffic image segmentation in which a wide variety of subjects (in terms of geometry, color, scale, etc.) can be found. Moreover, in table 6.1 the first two MultiResUnet models have a low number of epochs, especially considering that the patience is of 20 epochs when applying the early stopping criteria. Then, a higher patience could improve those results but this is not clear since the third best model does have higher number of epochs and performs similarly. Furthermore, the better performance of U-Net over DeconvNet is to be expected, since the U-Net incorporates skip layers while DeconvNet is a vanilla FCN. Also, notice that the standard deviation is always one order of magnitude below the average, which means the individual results do not differ significantly. Hence, the models are robust since they perform similarly across different sets of validation data.

[1] `https://github.com/igonzalezperez/Welding-droplet-analysis-using-fully-convolutional-networks`.
[2] All of the results of the grid search are shown in Appendix A

Table 6.1: Grid search results for globular dataset. The best three models for each architecture are shown based on the average validation loss over 4-fold cross validation.

Architecture	Batch size	Filters	Learning rate	Last epoch				Train loss		Val. loss	
				1	2	3	4	Mean	Std. dev.	Mean	Std. dev.
U-Net	16	32	0.001	95	82	118	67	0.1	0.015	0.19	0.028
U-Net	32	8	0.01	155	95	83	68	0.13	0.025	0.2	0.006
U-Net	32	8	0.001	93	106	82	85	0.12	0.007	0.2	0.02
DeconvNet	32	8	0.001	151	52	75	89	0.21	0.065	0.35	0.029
DeconvNet	8	16	0.001	68	90	89	105	0.2	0.025	0.37	0.013
DeconvNet	8	8	0.001	154	74	94	74	0.2	0.03	0.37	0.038
MultiRes	8	8	0.01	24	41	46	63	0.5	0.008	0.52	0.117
MultiRes	16	16	0.01	49	34	41	39	0.5	0.009	0.52	0.018
MultiRes	8	32	0.001	111	106	94	116	0.48	0.006	0.55	0.025

Table 6.2: Grid search results for spray dataset. The best three models for each architecture are shown based on the average validation loss over 4-fold cross validation.

Architecture	Batch size	Filters	Learning rate	Last epoch				Train loss		Val. loss	
				1	2	3	4	Mean	Std. dev.	Mean	Std. dev.
U-Net	8	8	0.005	74	91	105	122	0.23	0.017	0.28	0.008
U-Net	8	8	0.001	113	97	74	114	0.2	0.019	0.28	0.006
U-Net	8	16	0.005	144	119	93	111	0.19	0.01	0.28	0.009
DeconvNet	16	16	0.001	85	101	80	69	0.27	0.013	0.45	0.008
DeconvNet	8	8	0.001	63	90	76	80	0.32	0.021	0.47	0.016
DeconvNet	16	8	0.001	76	94	86	105	0.27	0.024	0.47	0.018
MultiRes	8	16	0.005	94	43	64	47	0.35	0.013	0.43	0.021
MultiRes	8	32	0.005	76	68	61	75	0.32	0.007	0.43	0.038
MultiRes	16	8	0.005	53	107	78	100	0.34	0.017	0.45	0.049

6.2 Training

After the grid search, the best models are selected based on the average validation loss and then these models are saved to make predictions. In figures 6.1a and 6.1b the learning curves for the globular and spray models are shown respectively. These graphs show that the training is indeed successful, steadily reaching a low loss value both in training and validation. The gap between the two is to be expected and the fact that the two curves show similar behavior as opposed to having the validation go up while training loss goes down suggests that there is no significant overfitting in neither of both cases.

Additionally, validation loss is monitored through the training process so that only parameters in which that metric improves are saved. Then, the best model is saved rather than the model at the end of training. In practical terms, since the early stopping has a patience of 20, when the training is done, the last model saved is 20 epochs prior to the last one which corresponds to the model with the smallest validation loss in the whole curve as seen in the dashed vertical lines in figures 6.1a and 6.1b.

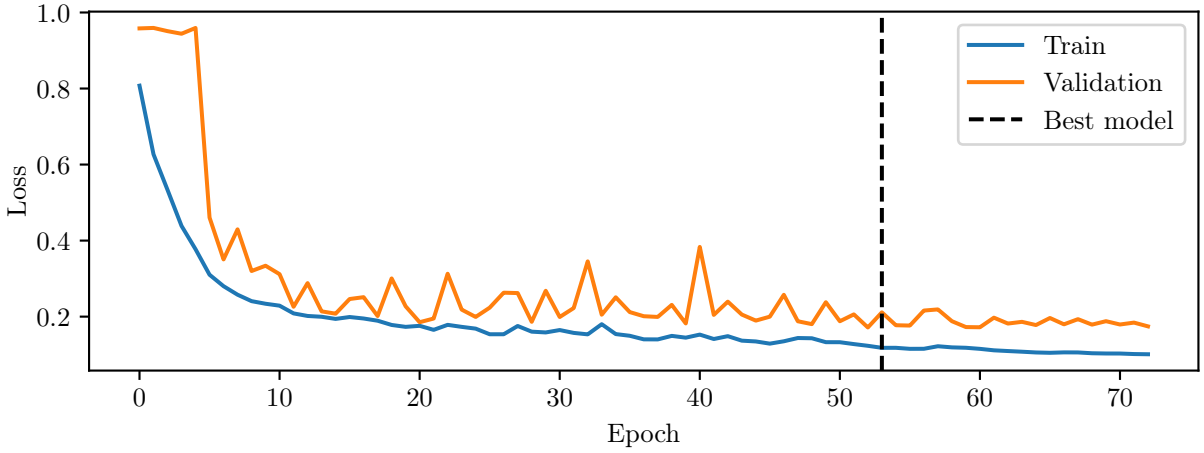
6.3 Testing

The best models obtained via grid search are trained and then used to make predictions over all of the image data, that is new masks are generated by the model. The globular model reaches a test loss of 0.02 and in spray the test loss is of 0.01. These results are low compared to those obtained in the training phase which can be explained by a couple of factors as follows.

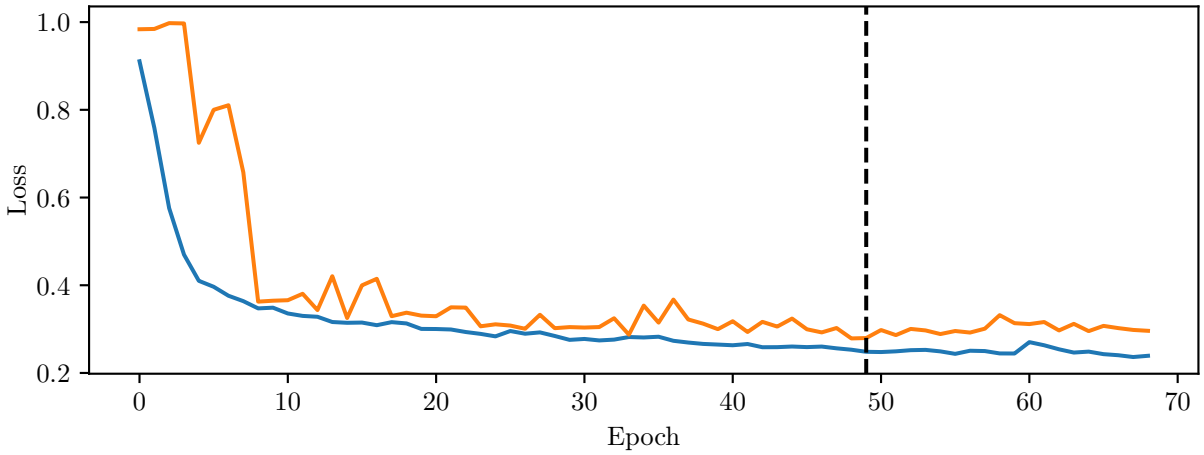
First, when training, dropout is applied to prevent overfitting but, when testing, dropout is not necessary since the network has already been trained and the weights are fixed. Therefore, the network is more powerful when testing which can yield lower loss values in comparison. Second, a lower test loss can be explained by the fact that the training examples where chosen to have a high variance in them so the network can learn the more complex patterns which can appear in the process. Finally, the training examples were augmented, hence the examples that were already diverse in geometry, get more complex by dropping pixels, adding elastic transformations and noise. Then, the testing set has a majority of examples that are rather simple compared to training examples which consequently reduces the loss value.

In figures 6.2a and 6.2b a histogram of the obtained loss for every example in the test set is shown for each transfer mode. As stated before, an overwhelming majority of examples have a small loss value and a small amount are on the higher end in both cases. A different pattern appears in globular and spray although the loss values remains small in all of the spectrum.

Examples of input and mask are sampled from some of the bins in each histogram, shown in figures 6.3 and 6.4. In the globular case, most of the images are similar to figure 6.3a reaching a low loss value and having a noticeably accurate segmentation. In the other examples in which the loss is higher, the network does not output a segmentation mapping and predicts zero for most of the pixels. This is because of the nature of the process of globular transfer, since all of those frames occur when a droplet has been released and submerged in the welding

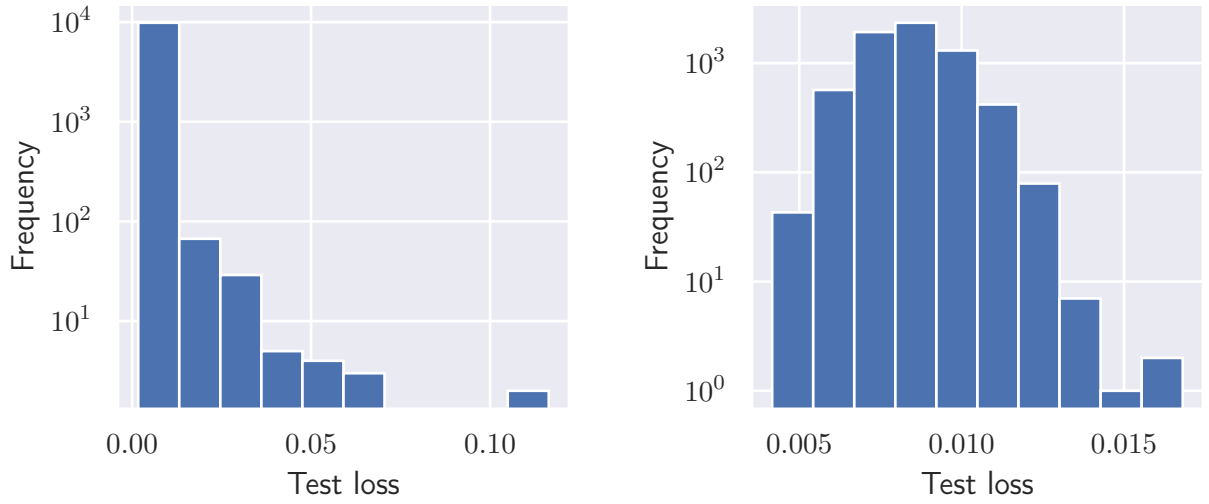


(a) Globular



(b) Spray

Figure 6.1: (a) Globular learning curve using the best model according to table 6.1. U-Net model with 16 batch size, 32 filters, 0.001 learning rate, Adam optimizer, Jaccard distance loss and early stopping of 200 epochs with patience 20 stopping at 53 epochs. Training loss: 0.12, validation loss: 0.17. (b) Spray learning curve using the best model according to table 6.2. U-Net model with 8 batch size, 8 filters, 0.005 learning rate, Adam optimizer, Jaccard distance loss and early stopping of 200 epochs with patience 20 stopping at 49 epochs. Training loss: 0.25, validation loss: 0.27.



(a) Globular

(b) Spray

Figure 6.2: Log histogram of the test loss obtained for every test example.

pool, and a new droplet is being formed. At this point when the droplet is just beginning to form, the network does not recognize a droplet in the frame, which can be useful since the droplet detachment frequency can be inferred from this behavior.

On the other hand, figure 6.4 shows that most of the loss values in spray transfer are low even on the higher end of the histogram. Nevertheless, figure 6.4d is from the worse performing examples and clearly the mapping is not accurate, this may be because the geometry is significantly different from the other examples, making it harder to predict. The fact that most of the examples in spray transfer have a similar loss while in globular transfer there is a wider gap happens due to the fact that the droplet's process in spray transfer is not separated by frames, that is, in most frames there are multiple droplets in the image, so the process of formation, detachment and landing of the droplet into the welding pool cannot be separated in a frame by frame basis, so spray transfer frames look more alike.

The previous results are to show some of the outliers with bad results. However, the majority of segmentation mappings are accurate both qualitatively since they look like they separate the droplet and quantitatively since the loss is small. In figures 6.5 and 6.6 some randomly sampled predictions are shown.

Moreover, an important aspect of this work is that the labels were manually generated, so the results are subject to a human bias, specifically the selection of the boundary of the droplet because at a pixel level there is not an exact distinction between droplet and background and a gradient is seen between the brighter pixels of the droplet and the darker pixels of the background.

Hence, to make more evident where the network decides to make the droplet-background division which is influenced by the labeling, figures 6.7 and 6.8 show the pixel values of a single horizontal strip, specifically the one that goes through the centroid of the droplet segmentation. There, one can see that there is a transition between droplet and background,

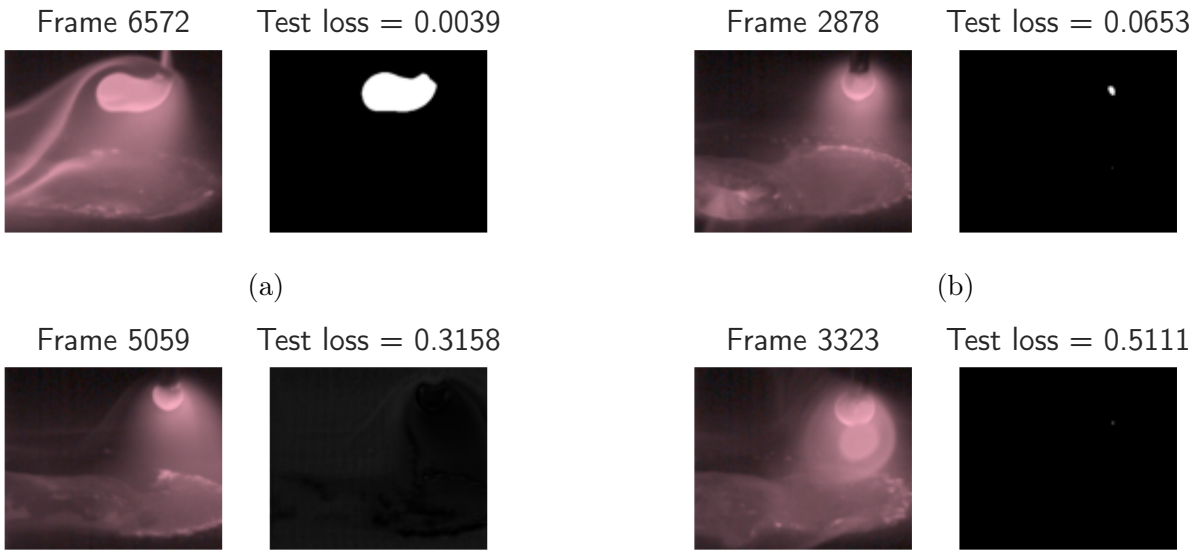


Figure 6.3: Image samples for globular transfer mode based on test loss.

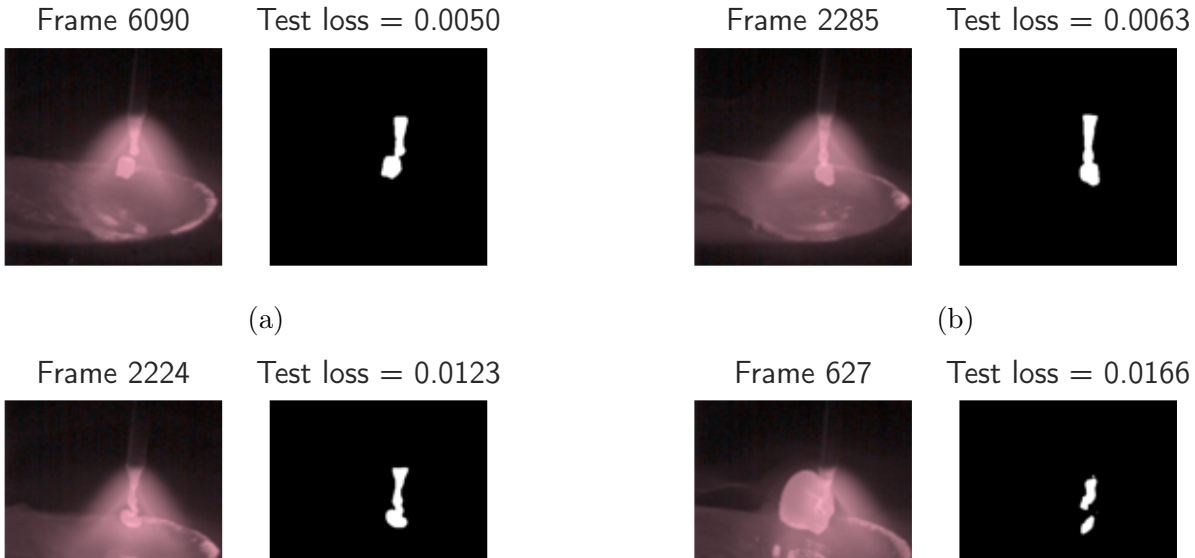
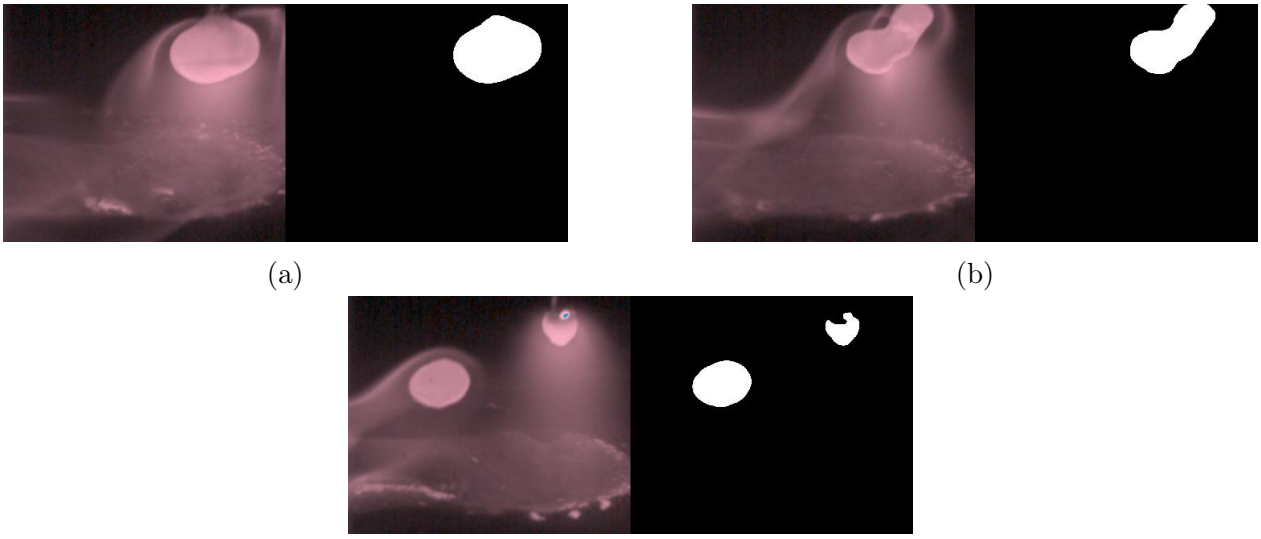


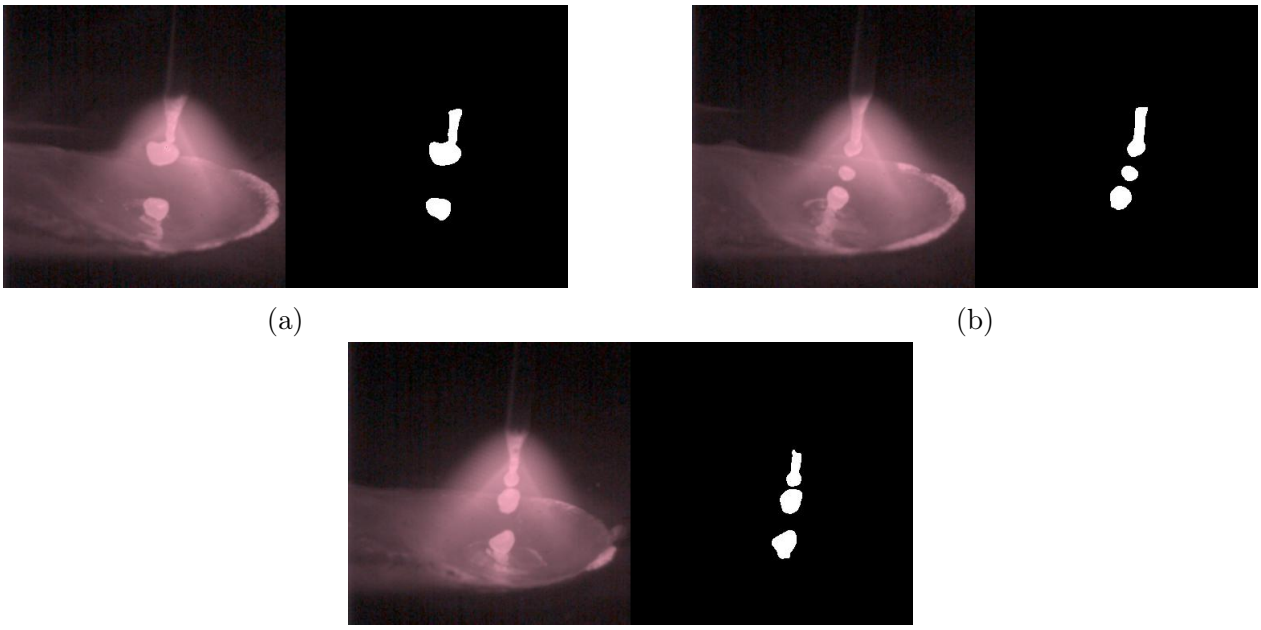
Figure 6.4: Image samples for spray transfer mode based on test loss.



(a) (b)

(c)

Figure 6.5: Globular transfer mode predictions.



(a) (b)

(c)

Figure 6.6: Spray transfer mode predictions.

and vertical lines show where the model cuts and makes the segmentation. The figures show that in the image there there is a steep transition between background and droplet, and the segmentation cuts around the middle of that transition.

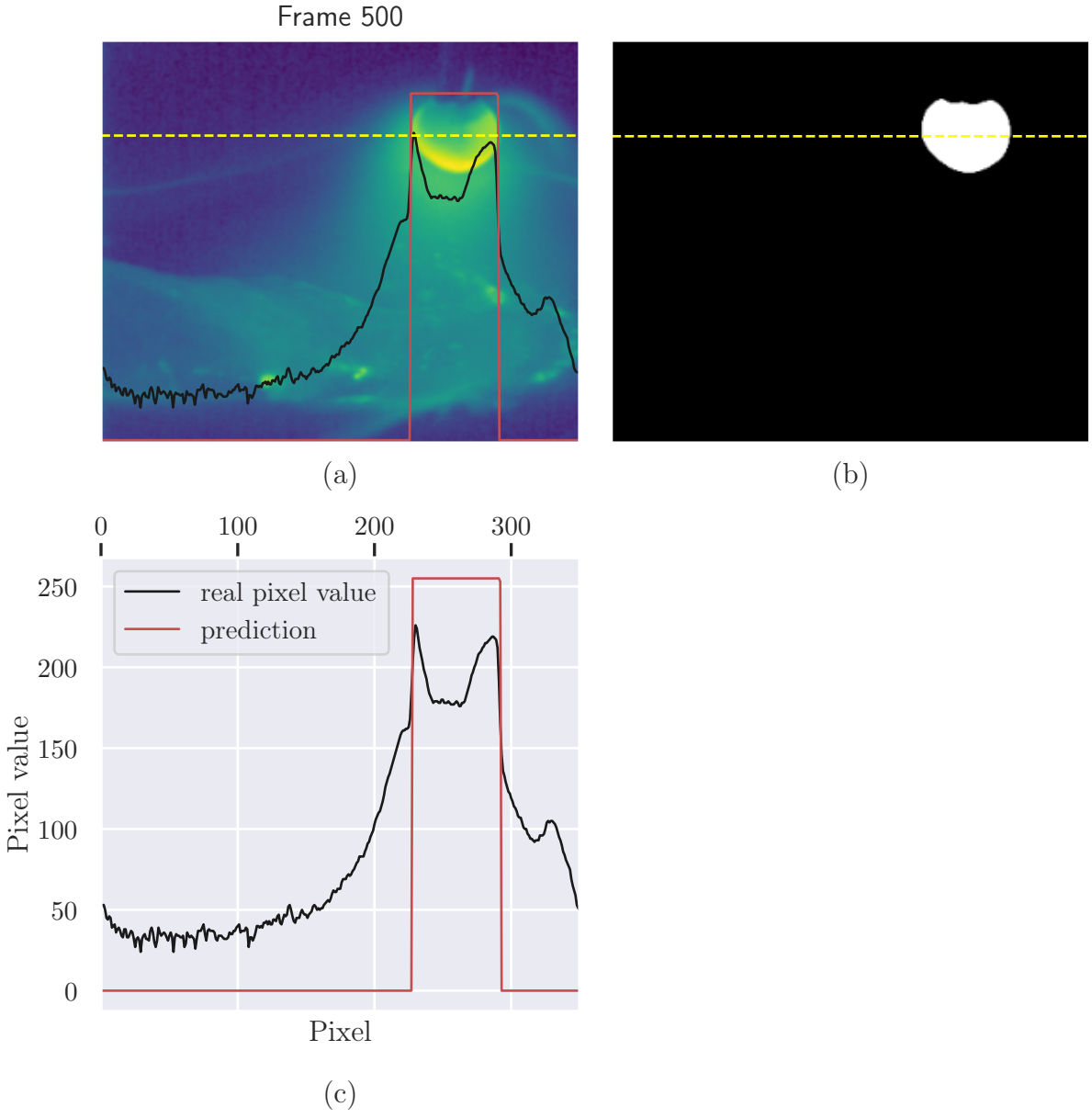


Figure 6.7: Predicted droplet boundary compared to original globular image. A horizontal strip is taken along the droplet's centroid (dashed yellow line in (a) and (b)) and the pixel values along that line are shown in (c). In (b) the predicted mask which generates the red curve in (c) is shown.

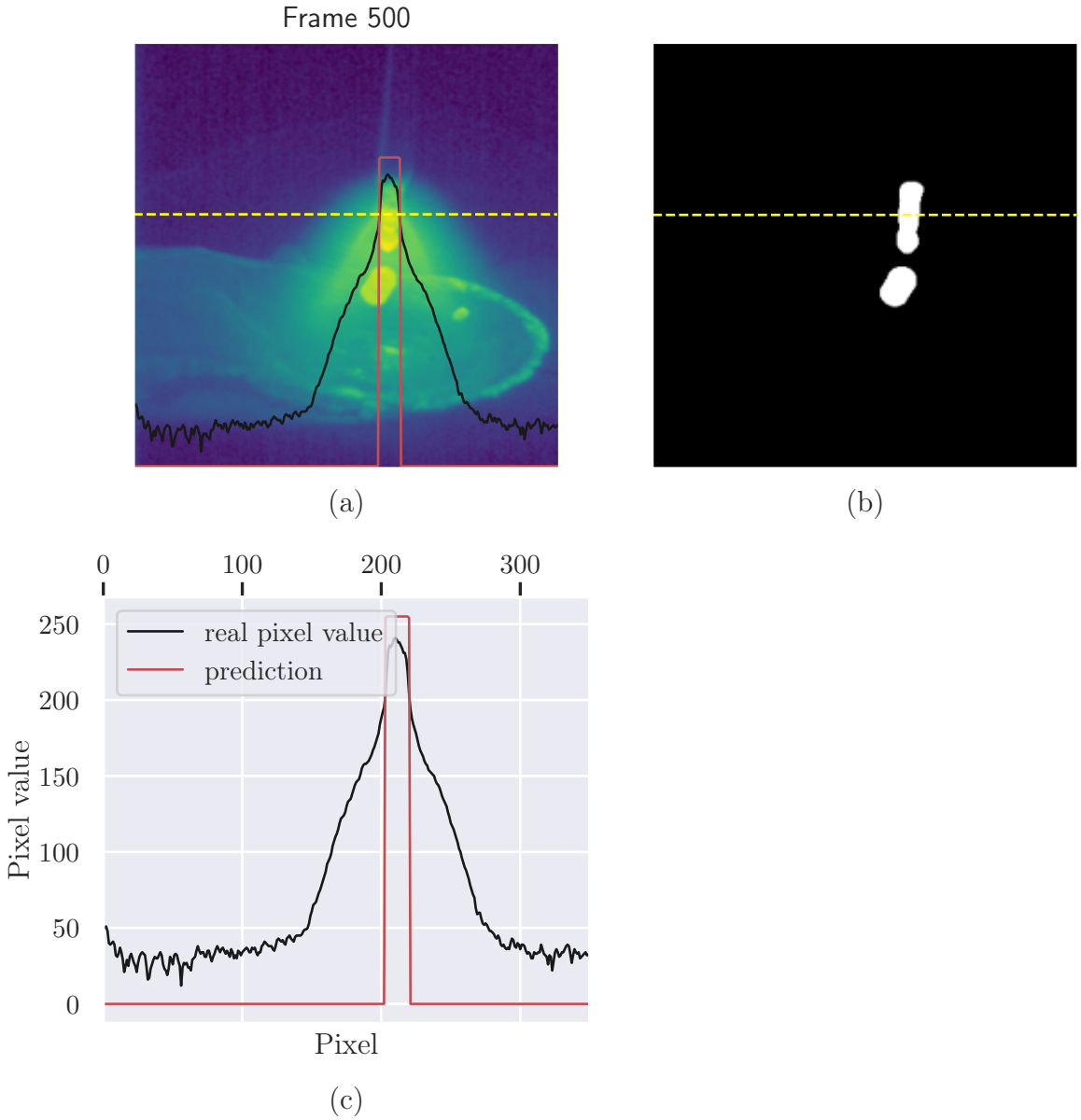


Figure 6.8: Predicted droplet boundary compared to original spray image.

6.4 Post processing

In this section the results after generating the segmentation maps are shown. These results consist in geometric and physical properties of the processes that help to understand them better. In Appendix B.1 there is a summary of the techniques used to obtain properties from the segmentation maps and general signal processing.

6.4.1 Centroids, velocity and acceleration

The centroids of each frame are obtained using `opencv` which can identify contours in the segmented images and then compute the moments of each contour. In images 6.9 and 6.10 some of the examples are shown. The results show that the centroids are accurately found in the segmentation maps and correspond to those in the original image. Moreover, it is possible to detect multiple contours within an image, so frames with more than one droplet have a separate centroid for each droplet.

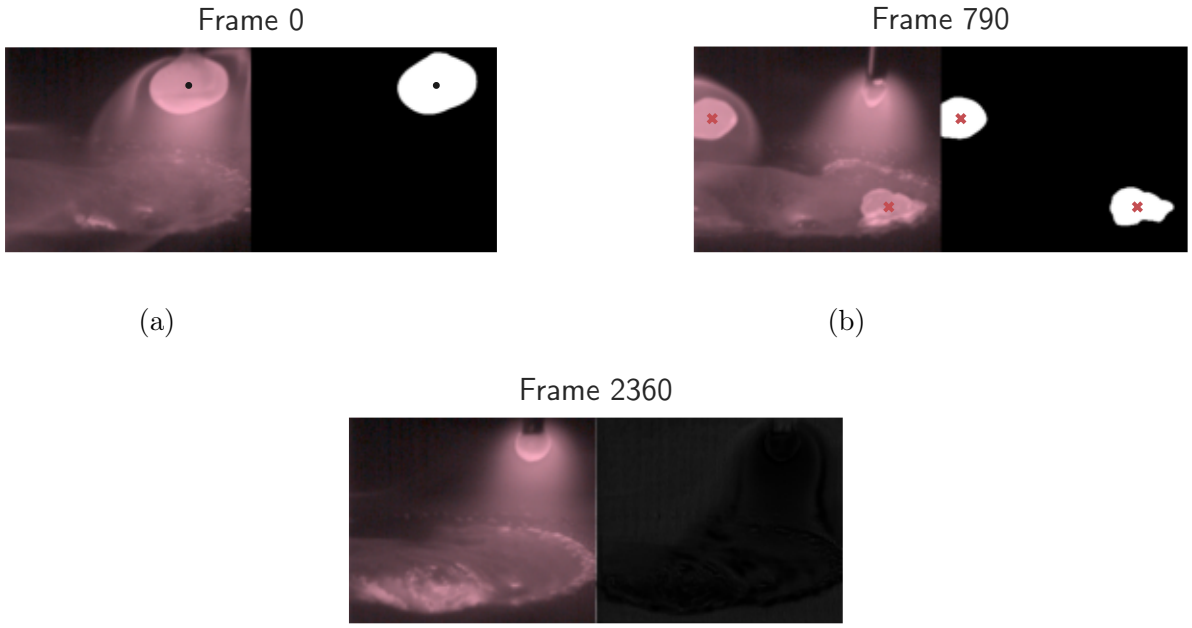
In globular transfer, the growth process usually has only one droplet and when the droplet detaches, more droplets appear in the frame. This can be due to the main droplet splitting into two or more, or the network identifying the tip of the wire as another droplet or some portion of the weld pool which is too bright. The tip of the wire being identified as a droplet occurs because the melting of the wire is a continuous process, so when the droplet detaches, another droplet begins to form. Nevertheless, most of the time the initial state of the growth process does not have any segmented droplets, as shown in figure 6.9c.

On the other hand, spray transfer has a faster growth and detachment of droplets since these are smaller than globular. This means that in most frames multiple droplets can be identified. The droplet forming in the wire is always segmented as a droplet and usually has an elongated shape because it is from where the droplets fall to the welding pool. The droplets that are falling have a more round shape compared to the one in the wire.

Furthermore, given the x and y coordinates at each point in time, it is possible to compute velocity and acceleration with the use of numerical differentiation (see Appendix B.1.2). Specifically, free flight centroid data is used to compute velocity and acceleration, that is the periods of time in which the droplet is detached from the wire and has not yet reached the weld pool. Calculations are not made throughout the whole video since the focus should be in each single droplet, as opposed to the stream of droplets which would cause discontinuities in velocity or area, obfuscating the analysis.

The identification of the droplet in the frame is a problem that has to be addressed since multiple centroids can appear in one image. In order to track a specific droplet through time, a continuity criterion is used. An initial droplet is selected, then at each time the next centroids are compared to the current centroid and the one with the minimum euclidean distance is chosen since it is expected that a droplet can only move within its vicinity from frame to frame.

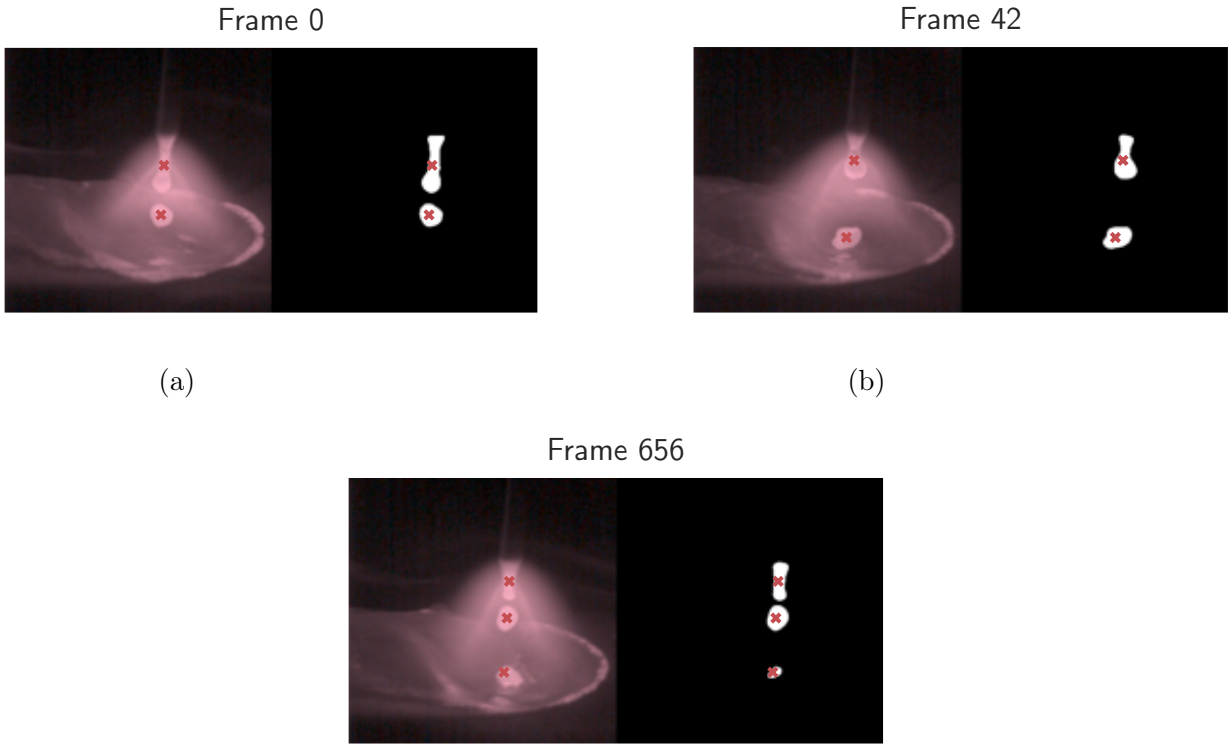
The trajectory, velocity and acceleration of different free flight droplet motion can be seen in figure 6.11. Also, statistical features for velocity and acceleration are shown in tables 6.3 through 6.6. Notice that the signs of velocity and acceleration are because of the origin of



(a) (b)

(c)

Figure 6.9: Globular transfer mode centroid predictions. Black dots are used for single droplet and red crosses when multiple droplets are in the frame.



(a) (b)

(c)

Figure 6.10: Spray transfer mode centroid predictions.

Table 6.3: Statistical features of velocity in each component for multiple free flight time windows.

Frames	Min $[cm/s]$		Max $[cm/s]$		Mean ± Std. dev. $[cm/s]$	
	v_x	v_y	v_x	v_y	v_x	v_y
$34-72$	-37.8	8.2	-10.9	35.1	-22.3 ± 7.1	23.7 ± 7.1
$289-342$	-29.9	5.5	-17.3	35.6	-23.1 ± 2.6	21.2 ± 8.2
$1429-1502$	-20.2	6.9	8.8	47.4	-7.9 ± 5.8	25.3 ± 8.5
$2279-2341$	-24.8	12.2	-3.9	45.9	-13.8 ± 5.7	31.4 ± 8.6

Table 6.4: Statistical features of the norm of the velocity for multiple free flight time windows.

Frames	$\lvert\vec{v}\rvert$		
	Min $[cm/s]$	Max $[cm/s]$	Mean ± Std. dev. $[cm/s]$
$34-72$	19.1	48.6	33.2 ± 7.1
$289-342$	21.6	44.7	31.8 ± 6.2
$1429-1502$	9.8	49.0	27.3 ± 8.6
$2279-2341$	17.5	51.4	34.8 ± 8.0

coordinates which is on the top left of the graphs shown in figure 6.11. Displacement in y is more or less fixed since the droplet will always land at about the same height while x displacement ranges from $1mm$ up to 4 mm. Lower x displacement is also seen in the velocity having a larger y component, hence landing faster as seen in figure 6.11. Moreover, velocity is around 30 cm/s having relatively small variations while acceleration has a much higher deviation both in magnitude as well as direction. This is because many forces contribute to the motion of the droplet, namely the gravitational force, Lorentz forces produced by the electromagnetic phenomena, forces produced by the shielding gas plasma and surface tension forces.

In comparison, in the work of *Weglowski* [49], accelerations of around 130 m/s^2 are obtained, similar to those shown here, although consistently smaller. The velocity is quite different, ranging from 50 m/s to 90 m/s for varying current intensities. The results are highly dependent on the experimental setup, so differences are to be expected. On the other hand, in the work of *Ray* [9] the values for acceleration are notably smaller, between 20 m/s^2 and 90 m/s^2.

Although these computations should be accurate based on the segmentation accuracy and subsequent centroid position, it is worth noting that higher resolution in space (more pixels) would get better results because usually the centroid displacements are around 1 pixel so many displacements that would yield different velocities are cast into the same velocity. The same happens with acceleration. This phenomenon can be seen in the graphs of figure 6.11 where the trajectory less smooth. On the other hand, higher resolution in time (sampling frequency) would allow to analyze shorter period motion, specifically spray transfer, which given the sampling frequency, the data points during free flight are scarce, which is the reason spray transfer mode is left out in this part of the analysis

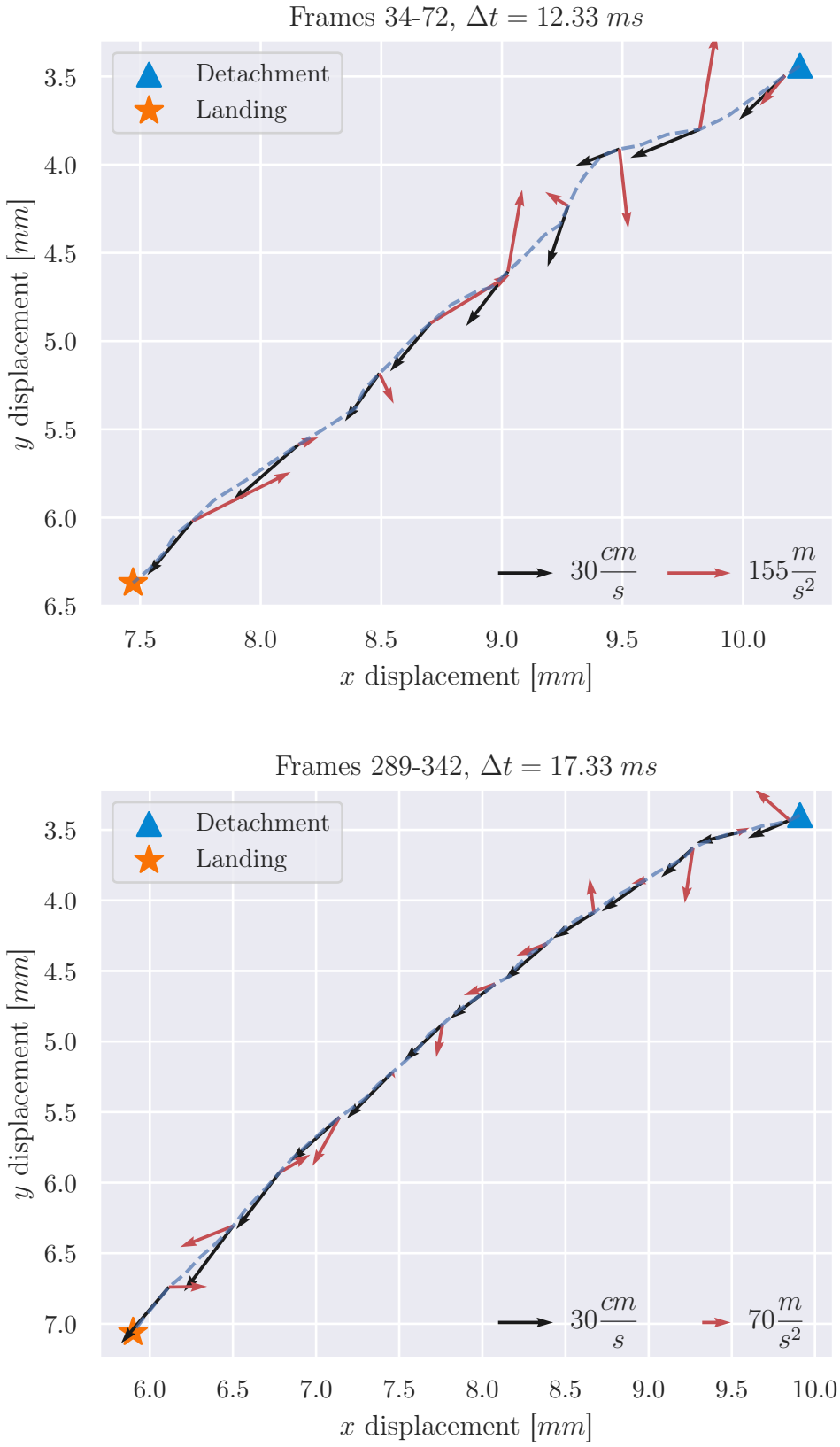


Figure 6.11: Trajectory (dashed blue line), velocity (black arrows) and acceleration (red arrows) of a free flight globular droplet for multiple time windows.

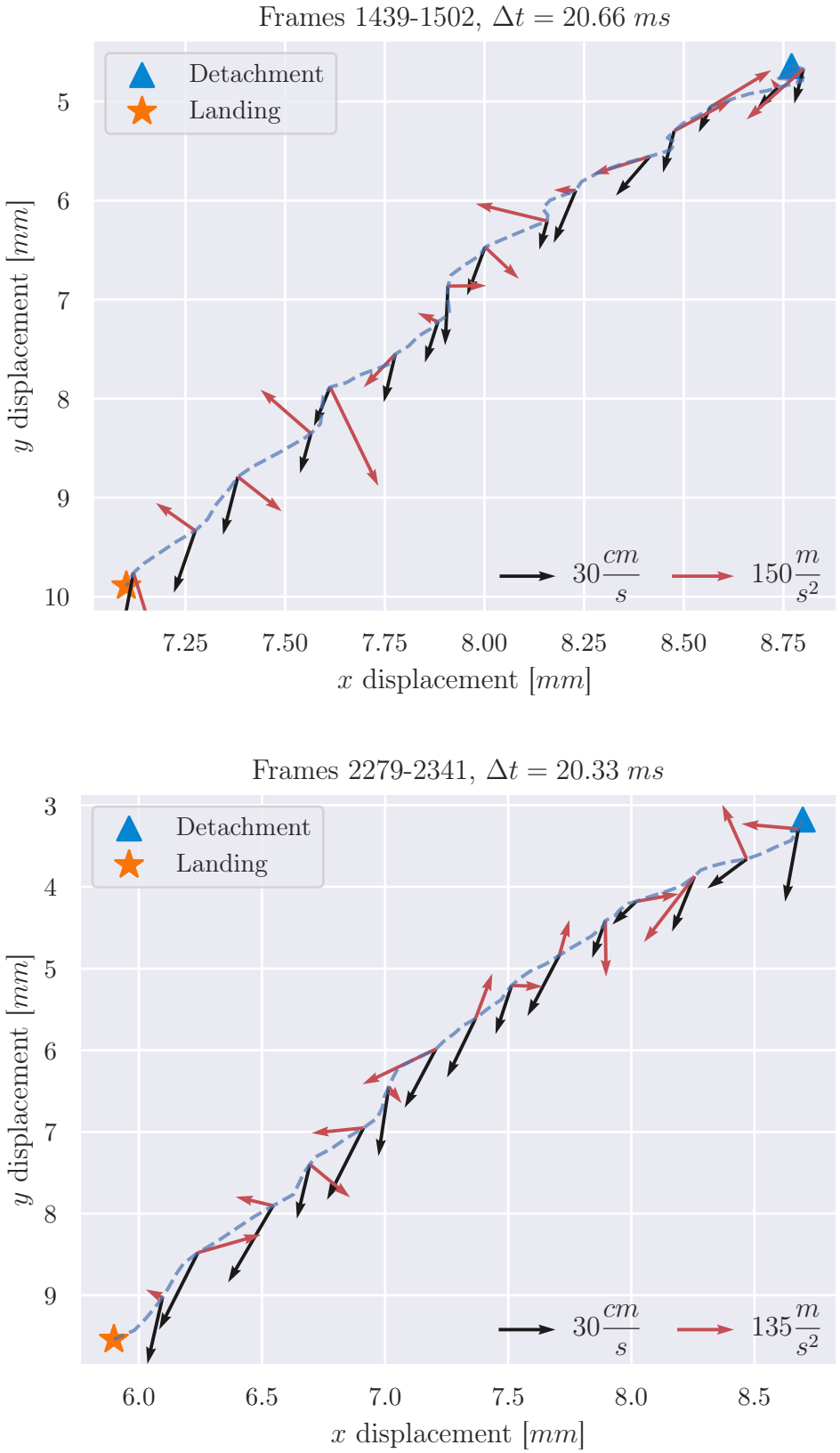


Figure 6.11 (cont.): Trajectory (dashed blue line), velocity (black arrows) and acceleration (red arrows) of a free flight globular droplet for multiple time windows.

Table 6.5: Statistical features of acceleration in each component for multiple free flight time windows.

Frames	Min $[m/s]$		Max $[m/s]$		Mean ± Std. dev. $[m/s]$	
	a_x	a_y	a_x	a_y	a_x	a_y
$34-72$	-314.9	-236.3	314.64	245.5	-3.8 ± 132.2	7.1 ± 115.0
$289-342$	-129.7	-100.9	93.8	160.1	-2.4 ± 52.3	12.3 ± 60.9
$1429-1502$	-410.2	-302.1	214.2	351.8	-9.7 ± 123.1	18.2 ± 127.0
$2279-2341$	-258.8	-211.8	253.4	262.8	-11.2 ± 109.3	-0.5 ± 102.6

Table 6.6: Statistical features of the norm of the acceleration for multiple free flight time windows.

Frames	$\lvert\vec{a}\rvert$		
	Min $[m/s]$	Max $[m/s]$	Mean ± Std. dev. $[m/s]$
$34-72$	17.9	328.7	155.4 ± 85.4
$289-342$	11.8	191.1	71.7 ± 38.2
$1429-1502$	19.7	497.1	152.6 ± 91.8
$2279-2341$	29.9	313.6	136.2 ± 63.7

6.4.2 Area and detachment rate

The contours found in each image can also be used to compute the area of a droplet[3]. Since the segmentation maps are binary, this can be done simply by summing all of the pixel intensities inside a given contour. Then, using the relationship in (5.1), it is possible to estimate the area in mm^2 as shown in figures 6.12a and 6.12b[4].

The globular case shows a pattern where there is a growing phase in which only one droplet is detected (black dots), then the area drops suddenly and the number of droplets detected is more than one (red crosses). Later, the network detects no droplets in the frame for a brief time (blue dots) and then the cycle is repeated.

Since the value of the area is an indicator of when the droplet detaches from the wire, it is useful to detect when this is happening. This is done by smoothing the signal using a *hanning* window and convolving it with the original signal which effectively works as a weighted average. Notice that since the area can have multiple values for a given frame, the sum of these values is used instead. In this way, the smoothed signal is easier to process simply by finding its peaks, which is carried out using `scipy`'s `find_peaks` function which allows to set a variety of parameters such as height, prominence and width of peaks[5]. As seen in figure 6.12a, the smoothed curve's (green curve) minimum peaks (vertical dashed lines) coincide with the detachment of droplets accurately. In this way it is possible to count the

[3]Perimeter and volume are also computed and show similar trends.

[4]Not all the frames are displayed so the plot does not get cluttered. The full graph can be seen in video format in the GitHub project.

[5]Notice that `scipy` finds maximum peaks and in this case the opposite is needed, so in practice the negative value of the signal is used.

number of droplets over time as well as the point in time when that occurs. Specifically, the number of globular droplet detachments observed is 25 while the detected is 24.

The droplet detachment is found accurately across almost all of the globular video, detecting only one detachment fewer than the original if one counts by eye the number of droplets detached from the original video. Nonetheless, due to the method used for finding peaks, this means that the detachment was not found because the area did not diminish significantly (hence, no peak was found). This part of the signal is shown in figure 6.13 in which it can be seen that it breaks the pattern because there is no portion with no droplets present (blue dots) and multiple droplets are detected instead.

This behavior is due to a shortcoming of the neural network. In some occasions the weld pool is too bright, this can happen simply because of the movement of material in the weld pool or because the droplet, when is reaching the weld pool, is reflected by it and therefore the image effectively shows two droplets. Examples of this are depicted in figure 6.14. These examples are found throughout the whole video, but only in the time window shown in figure 6.13 they affect the results. Although this could be solved with a more robust deep learning architecture, it is probably more effective to have more labeled images of those specific cases, i.e. droplet reflection and overall brightness of the weld pool.

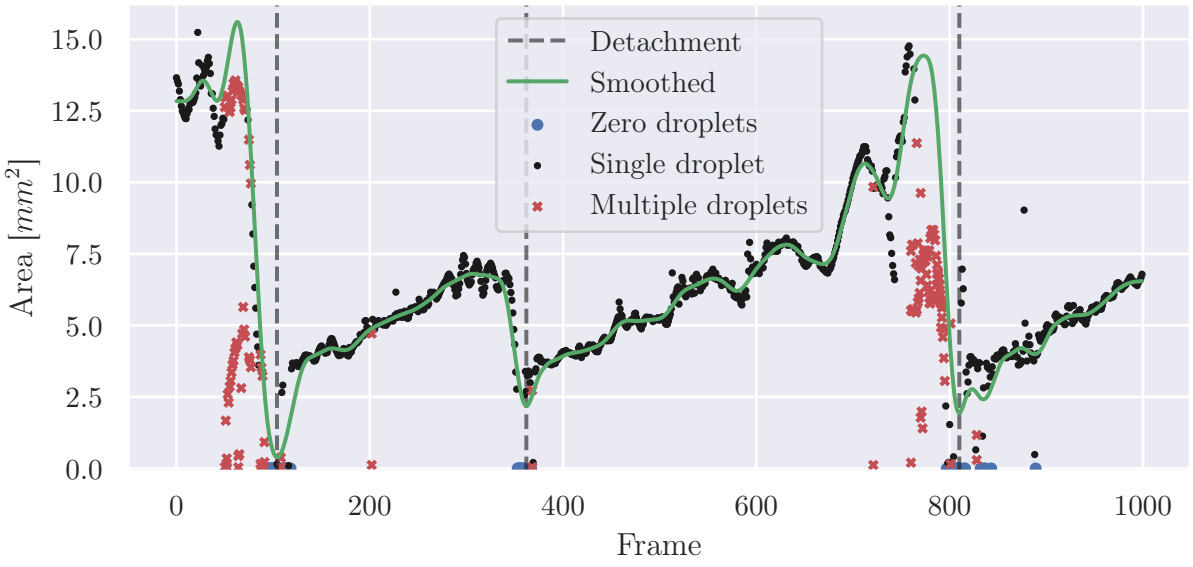
In the spray transfer case, the pattern of the area is different from globular, a much shorter cycle is seen, around 10 times faster than in globular. Almost all of the time more than one droplet is detected in the image, and the process consists in that one of the droplets grows in size until reaching a limit where it suddenly decays while the other droplet remains with a similar area usually oscillating around similar values. The reason for this is that the tip of the wire is always forming droplets and therefore is detected in every frame and it corresponds to the droplet with increasing area in figure 6.12b. Then, when the area is large enough, a droplet is released which corresponds to the smaller area in the graph and the oscillation is due to the droplets movement during its free flight to the weld pool.

Additionally, some occurrences of single droplet detection can be seen, notably when the area is higher. This occurs when the forming droplet accumulates more material and therefore takes longer to detach, which allows for the current flying droplet (smaller area) to submerge in the weld pool before the new droplet releases from the wire.

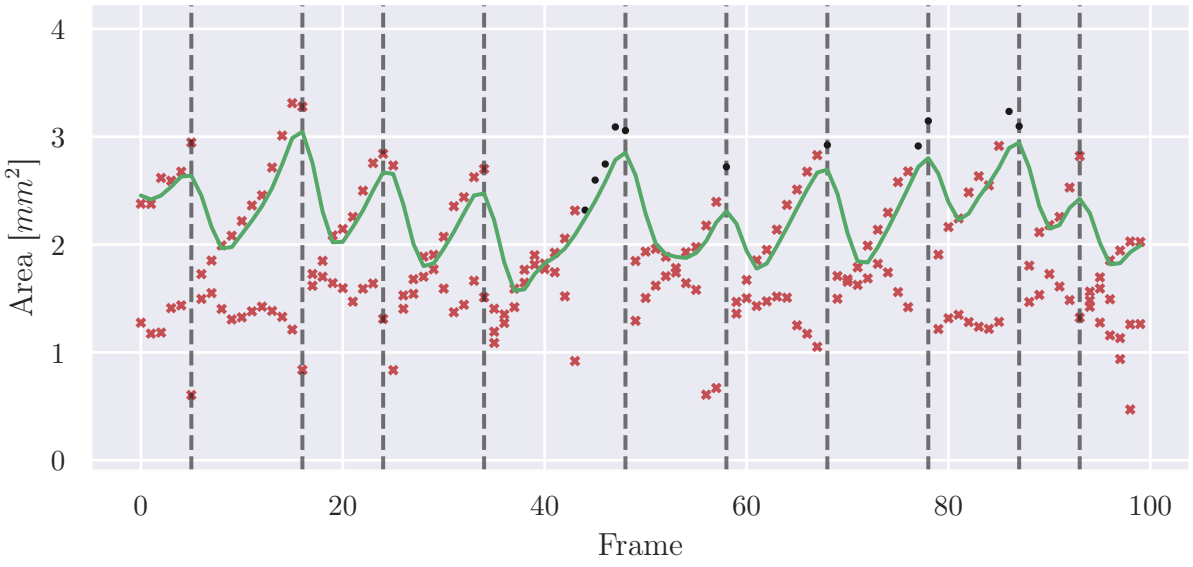
Although the graphs are different in spray and globular, the same approach can be used to detect spray droplet detachment[6], that is finding peaks in the area. The difference in the spray case is that the indicator is always the largest droplet, so the maximum area value is taken (as opposed to the sum in globular) and then the peaks are found in the same fashion as in globular. The number of detachments found is of 673 while the actual detachments are 676. Furthermore, the shortcomings of the segmentation maps are still present as in globular but to a lesser extent, specially droplet reflection since the droplets are smaller.

Also, it is worth mentioning that although the method of finding peaks is similar, in the globular case the prominence is found in the minima while in spray are the maximum values. Therefore, the time of the detachment is delayed in globular transfer but not in spray, in

[6]Parameters must be tweaked accordingly, e.g., a much smaller hanning window is needed.



(a) Globular



(b) Spray

Figure 6.12: Area over time computed from segmentation maps. Black dots represent one droplet, red crosses are two or more droplets in a frame and blue dots appear when no droplet is present in the image. The smoothed signal (green curve) is used to find peaks (vertical dashed lines) which correspond with droplet detachment.

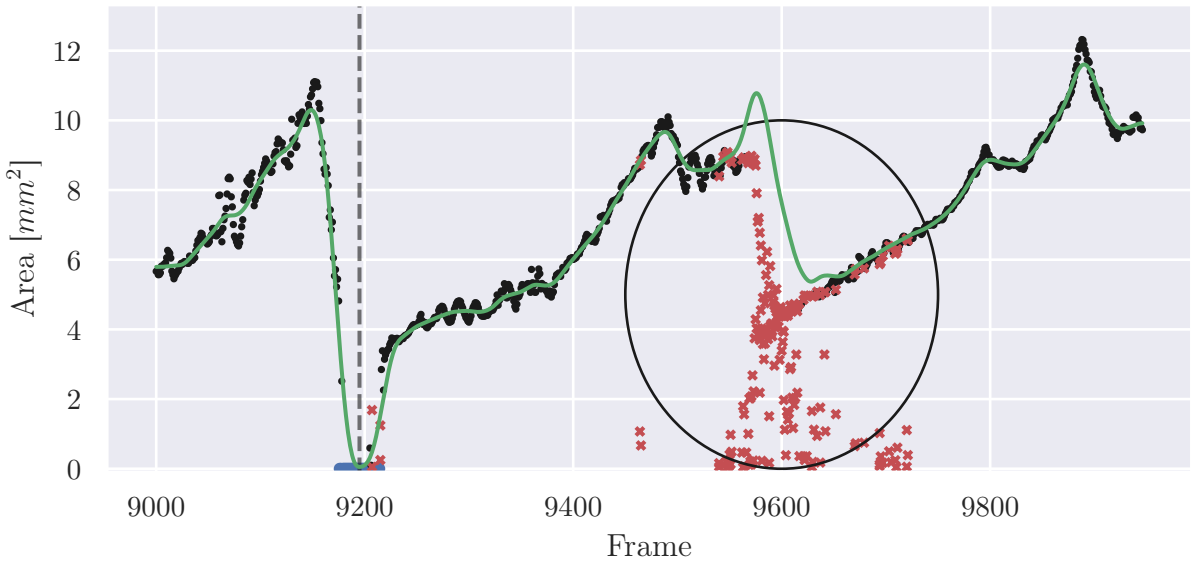


Figure 6.13: Globular area curve where no detachment is found.

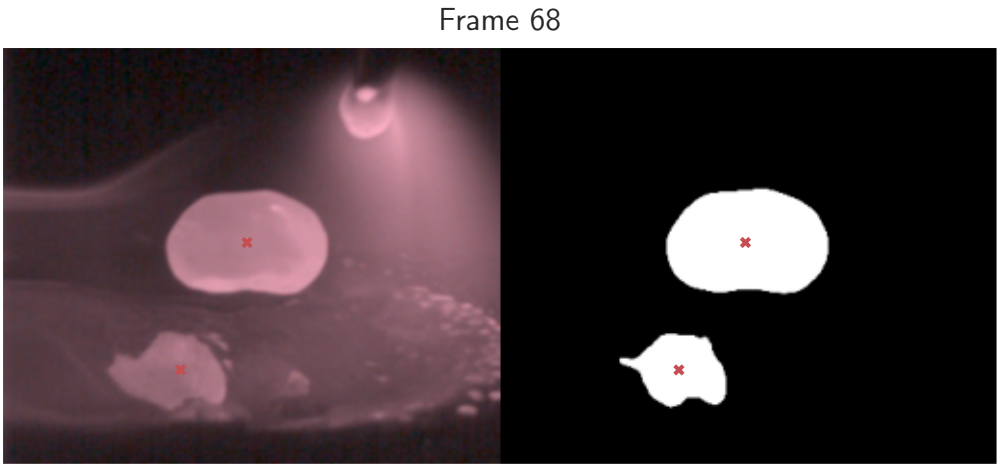
Table 6.7: Statistical data of the detachment process.

	Time between droplet detachment $[ms]$				
Transfer mode	Min	Max	Mean	Std. dev.	Detachment area $[mm^2]$
Globular	51.32	239.61	131.72	47.14	$[10-15]$
Spray	1.00	8.65	3.31	1.12	$[2-4]$

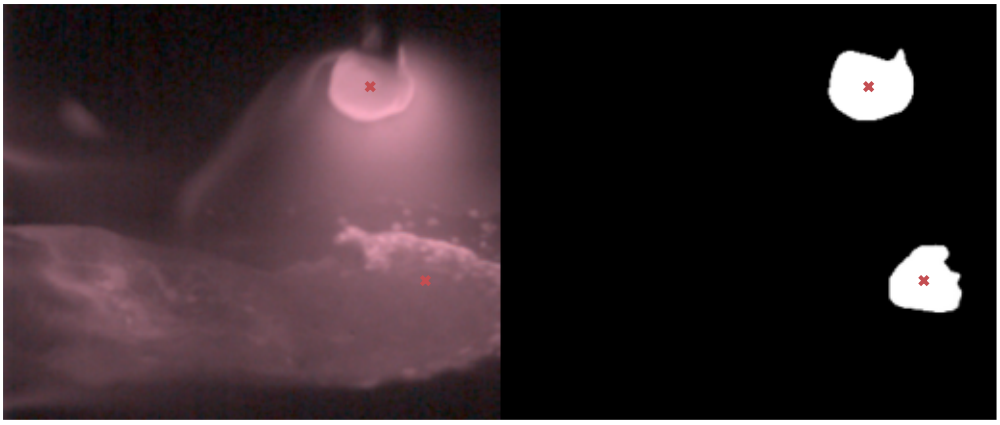
Transfer mode	Total predicted detachments	Total detachments
Globular	24	25
Spray	673	676

which the time of detachment obtained should match the actual detachment. Nonetheless, this does not affect the number of detachments overall nor the time spacing between them.

Finally, a summary of the detachment data is shown in table 6.7 where it can be seen that, in general, the detachment times can vary considerably given the maximum and minimum values as well as the mean to deviation ratio. As expected, the detachment rate in spray transfer is much faster than in globular. Moreover, the time at which droplets detach is somewhat determined by area in the sense that it could happen anywhere between the ranges shown in the table. Also, almost all of the detachments were found both in terms of number as well as moment in time.



(a) Droplet reflection



(b) Overall brightness of weld pool

Figure 6.14: Examples of over segmentation.

6.5 Surface tension

Rayleigh's formula [50] is used to calculate the surface tension (see Appendix B.1.2, eq. (B.1)). Approximations of density, volume and oscillation period are used. Density is obtained from literature and a value of 6500 $kg \cdot m^{-3}$ is used [51], volume is calculated by fitting an ellipse to the segmentation using `opencv` and assuming the 3D droplet is a prolate ellipsoid, that is it has symmetry along its minor axis, so formula (B.2) can be applied. Additionally, the volume during free flight should remain constant since it is a liquid droplet and no mass is added or removed. However, because the volume is calculated from the area, it will have inherent variations since the area is not constant, so the mean of the volume is taken to calculate the surface tension. Furthermore, the oscillation period can be obtained from Fourier analysis.

In figure 6.15 the perimeter signal as well as Fourier spectrum is shown for different free flight time windows. From the graphs, the main frequencies are between $50 - 100\ Hz$ which indicates that those frequencies are indeed the ones associated to the main oscillation in the perimeter plot since the scale of the grid is $0.003\ s$ (i.e. $333\ Hz$) which is around the same order of magnitude. Hence, it is unlikely that the frequency of interest is higher than the sampling theorem would allow and aliasing is not a concern. Thus, these frequencies can be converted to a period by taking the inverse and then used to compute the surface tension.

Table 6.8 shows the calculated surface tension for different time windows, including those in figure 6.15. If we compare to other literature results, the values are rather high, ranging from[7] $1\ Nm^{-1}$ up to $4.4\ Nm^{-1}$ while in the work of *Subramanian* [51] values between 0.9 and 1.8 are found. Moreover, in *Ray* [9] values range between $1.25\ Nm^{-1}$ and $3.2\ Nm^{-1}$ which are closer to those shown here. Nonetheless, the specific values depend on the experimental setup, that is gas composition, wire material, current, among others. Another consideration to have is that the droplet size in this work is larger than other works based on volume calculations. For example, in *Ray*, the maximum volume considered is of $13\ mm^3$ while in this case volumes up to around $40\ mm^3$ are obtained. Despite the fact that the volume is an approximation, it does seem reasonable if we compare the sizes of droplets in figures 3.2 and 5.1 since in the latter case, the droplet is several times larger than the wire. Also, the values get as low as $0.83\ Nm^{-1}$ which is lower than in *Subramanian*, so here the values have a higher variance in general. This is because the globular process in the analyzed video is erratic, and the droplet release and landing can occur in a variety of ways.

An important remark is that these results only apply for globular transfer. This is because the fast pace of the droplets in spray transfer do not allow to have enough data points during free flight motion. The number of points that can be gathered are between $5 - 10$ consecutive frames, so the Fourier analysis becomes unfeasible and is not possible to accurately estimate the oscillation period. Nevertheless, volume is calculated, and results are similar to those in *Subramanian* [51] of around $1\ mm^3$ for free flight droplets. Notice that the significantly smaller volume does not necessarily mean less surface tension, since the period should also be much smaller and surface tension is inversely proportional to the square of the period.

[7] Although the whole video has more instances of droplet detachment those shown here, all lie within the same range.

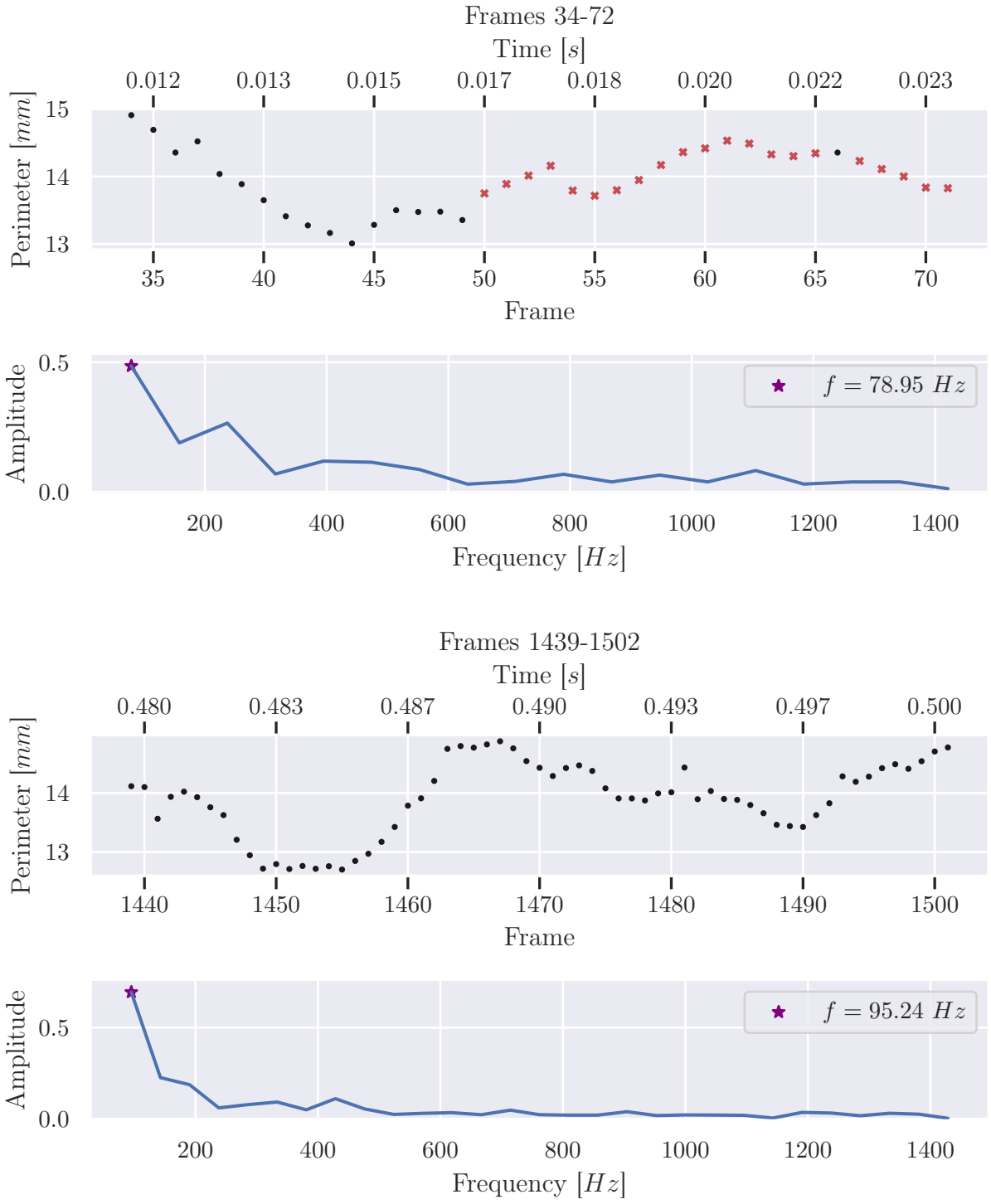


Figure 6.15: Perimeter signal and corresponding Fourier spectrum for different free flight time windows.

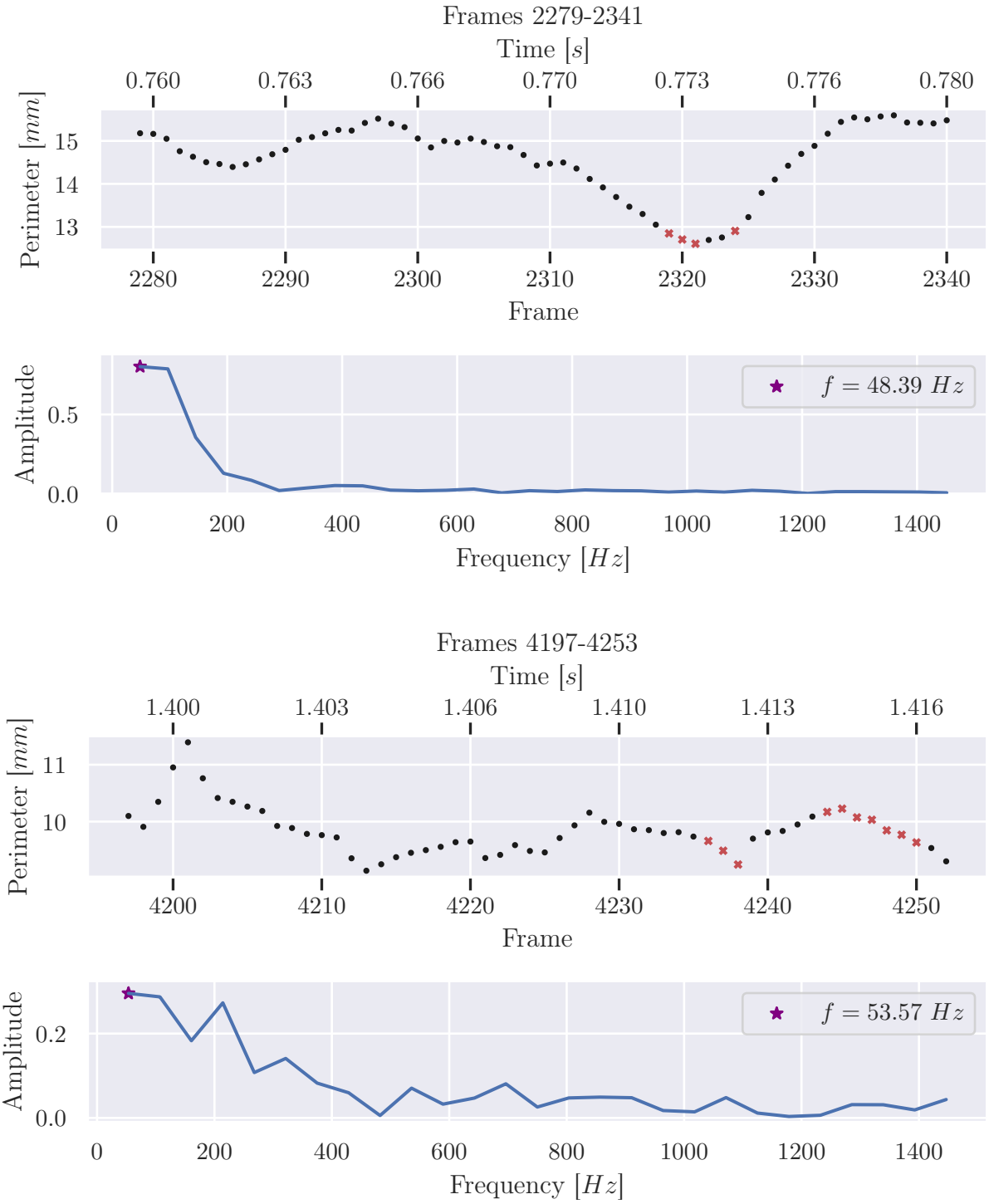


Figure 6.15 (cont.): Perimeter signal and corresponding Fourier spectrum for different free flight time windows.

Table 6.8: Volume and surface tension calculated values for multiple time windows.

Frames	Volume $[mm^3]$	Surface tension $[Nm^{-1}]$
$34 - 72$	29.09 ± 2.27	1.38 ± 0.11
$1439 - 1502$	28.24 ± 3.23	1.67 ± 0.19
$2279 - 2341$	25.22 ± 1.76	0.81 ± 0.05
$4196 - 4253$	11.28 ± 2.10	3.83 ± 0.71
$4553 - 4577$	19.90 ± 5.82	1.77 ± 0.52

Frames	Volume $[mm^3]$	Surface tension $[Nm^{-1}]$
$4761 - 4853$	11.00 ± 1.41	4.39 ± 0.56
$4983 - 5031$	8.86 ± 1.16	2.35 ± 0.31
$5732 - 5816$	23.81 ± 2.10	2.09 ± 0.18
$6589 - 6653$	23.04 ± 1.28	1.50 ± 0.08
$7754 - 7817$	27.60 ± 1.48	1.91 ± 0.10

Chapter 7

Concluding remarks and future work

7.1 Conclusions

In this work the aim is to be able to implement a novel approach to GMAW video analysis using deep learning to gain further understanding of the process.

First, a small dataset was manually labeled in order to train in a supervised fashion. Additionally, manual labels were augmented using image augmentation. Then, despite having a large, unlabeled dataset, it is possible to generate a working dataset with relative ease and more importantly, results can be improved simply by generating more labels.

The proposed deep learning architectures based on Fully Convolutional Networks prove to be a good candidate to approach the segmentation problem after validating results using grid search and comparing the DeconvNet, U-Net and MultiResUnet architectures. Specifically the U-Net architecture is able to reach a small loss both in training and testing generating accurate mappings for most of the images while not being an excessively complex model. This approach is able to generate segmentation mappings of thousands of images in seconds/minutes as opposed to previous work which was more centered in analyzing a handful of images.

Nevertheless, the network has some drawbacks when predicting segmentation maps, particularly over segmentation due to the brightness and reflection of the welding pool. This can be solved by adding more data to the model, since the labels were made with a diversity in geometry as a priority and not the mentioned problems, which is why the shape of the actual droplets is mostly correctly segmented.

Later, the calculation of relevant parameters that characterize the process is carried out, namely for droplet location (centroid), velocity, acceleration, perimeter, area, detachment frequency, volume and surface tension. The perimeter, area and volume are calculated throughout the whole video and the rest of the parameters are obtained in specific time windows during the free flight movement of droplets since these parameters only make sense when analyzing a specific droplet, as opposed to the constant stream of droplets. Specific droplets can be tracked simply by using a continuity criterion for their location.

The obtained parameters are compared with literature values and most of them lie within expected ranges, although there is discrepancy, mainly due to different experimental setups and focus of the analysis of different papers. Nonetheless, these parameters can help gain a better understanding of the process by measuring droplet detachment rate which can be

further used for deposition rate using density and volume calculations. Also, kinematic properties are also relevant since the interplay of forces acting on a droplet affect acceleration. Furthermore, velocity can be further related to kinematic energy of the droplet. Moreover, the parameters calculated here are rather exploratory, and more information could be gained depending on the context and necessity of the specific experiment.

In conclusion, the U-Net is able to successfully generate segmentation mappings of globular and spray GMAW metal transfer. Then, several geometric and physical parameters are obtained, mostly in agreement with the literature. Then, these can be used to better understand the process which in turn can help to analyze and optimize the process. The main drawbacks can be solved by gathering more data and increasing its resolution. Consequently, this opens many possibilities to apply deep learning in the welding field for many applications given that deep learning techniques have not been widely adopted in the field yet.

7.2 Further work

A number of possibilities arise given that the proposed model is successful. First, more experiments can be carried out to have a better comparison ground against the available literature. A common approach is to get results for different current intensities. Moreover, a use of gear with more spatial and time resolution can would be helpful when computing some parameters that come from small displacements of frequencies.

As with all data driven models, results can improve greatly if more data is available. Specifically, investing efforts in generating proper labels and augmenting them can help overcome the drawbacks mentioned before.

The architectures used are mainly for image data, but in this case, since videos are used, it could be beneficial to couple the used models with recurrent networks which make use of the temporal dependence of the data. Also, there are segmentation models that are used in videos (as opposed to separate images) such as Fast-RCNN and Mask-RCNN which could also be used. Another improvement would be using transfer learning, to take advantage of previously trained strong models.

Finally, since this work is mainly an offline tool, that is, it is intended to analyze the process after it has been carried out, it would be interesting to apply the same procedure in an online fashion, retrieving data in real time which would be helpful in the welding in-situ operation.

Bibliography

[1] S. Ozcelik and K. Moore. *Modeling, sensing and control of gas metal arc welding.* Elsevier, 2003.

[2] M. Yapıcı, A. Tekerek, and N. Topaloglu. Literature review of deep learning research areas. *Gazi Journal of Engineering Sciences*, 5:188–215, 12 2019.

[3] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[4] B. Rezaeianjouybari and Y. Shang. Deep learning for prognostics and health management: State of the art, challenges, and opportunities. *Measurement*, 163:107929, 2020.

[5] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72-73:303 – 315, 2016.

[6] J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold. Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics*, 34:1 – 11, 2016. System-Integrated Intelligence: New Challenges for Product and Production Engineering.

[7] Y. Zhang, D. You, X. Gao, N. Zhang, and P. P. Gao. Welding defects detection based on deep learning with multiple optical sensors during disk laser welding of thick plates. *Journal of Manufacturing Systems*, 51:87 – 94, 2019.

[8] Martins A Thomson R., Alvarez G. and Absi S. Analysis of gmaw process with deep learning and machine learning techniques. *Journal of Manufacturing Processes*, 62:695–703, 2021.

[9] N. Ray, A. Ahmetovic, S. Das, K.M. Scott, A.P. Gerlich, and P.F. Mendez. Active contour (snake) methodology for minimally user-interactive visual tracking of high speed videos of free-flight metal transfer. In *Fabtech/AWS Annual Meeting*, page 165, Atlanta, GA, 2010.

[10] P. Zhai, S. Xue, T. Chen, J. Wang, and Y. Tao. An image-processing method for extracting kinematic characteristics of droplets during pulsed gmaw. *Applied Sciences*, 9:5481, 2019.

[11] G. Litjens, T. Kooi, B. E. Bejnordi, A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image

analysis. *Medical Image Analysis*, 42:60 – 88, 2017.

[12] M. de Bruijne, B. van Ginneken, M. Viergever, and W. J. Niessen. Interactive segmentation of abdominal aortic aneurysms in cta images. *Medical Image Analysis*, 8(2):127 – 138, 2004.

[13] B. H. Menze, A. Jakab, S. Bauer, and J. Kalpathy-Cramer. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015.

[14] M. Silveira, J. C. Nascimento, and J. S. Marques. Comparison of segmentation methods for melanoma diagnosis in dermoscopy images. *IEEE Journal of Selected Topics in Signal Processing*, 3(1):35–45, 2009.

[15] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 878–885 vol. 1, 2005.

[16] N. Friedman and S. J. Russell. Image segmentation in video sequences: A probabilistic approach. *CoRR*, abs/1302.1539, 2013.

[17] A. Albiol, L. Torres, and E. J. Delp. An unsupervised color image segmentation algorithm for face detection applications. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, volume 2, pages 681–684 vol.2, 2001.

[18] Y. Du, E. Arslanturk, Z. Zhou, and C. Belcher. Video-based noncooperative iris image segmentation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):64–74, 2011.

[19] B. M. Mehtre, N. N. Murthy, S. Kapoor, and B. Chatterjee. Segmentation of fingerprint images using the directional image. *Pattern Recognition*, 20(4):429 – 435, 1987.

[20] J. Barlow, S. Franklin, and Y. E. Martin. High spatial resolution satellite imagery, dem derivatives, and image segmentation for the detection of mass wasting processes. *Photogrammetric Engineering & Remote Sensing*, 72, 06 2006.

[21] S. Ghosh, N. Das, I. Das, and U. Maulik. Understanding deep learning techniques for image segmentation. *ACM Comput. Surv.*, 52(4), August 2019.

[22] D. Q. Duong, K. C. T. Nguyen, N. R. Kaipatur, E. H. M. Lou, M. Noga, P. W. Major, K. Punithakumar, and L. H. Le. Fully automated segmentation of alveolar bone using deep convolutional neural networks from intraoral ultrasound images. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6632–6635, 2019.

[23] V. Casser, K. Kang, H. Pfister, and D. Haehn. Fast mitochondria detection for connectomics. In *Medical Imaging with Deep Learning*, 2020.

[24] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmen-

tation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, 2015.

[25] N. Ibtehaz and M. S. Rahman. Multiresunet : Rethinking the u-net architecture for multimodal biomedical image segmentation. *Neural Networks*, 121:74 – 87, 2020.

[26] M. P. Groover. *Fundamentals of modern manufacturing : materials, processes, and systems*. Third edition. Hoboken, NJ : J. Wiley, 2007.

[27] J.F Lancaster. The physics of welding. *Physics in Technology*, 15(2):73–79, mar 1984.

[28] P. F. Mendez, G. Goett, and S. D. Guest. High-speed video of metal transfer in submerged arc welding. *Welding Journal*, 94(10):325s–332s, 2015.

[29] Y. M. Zhang and P. J. Li. Modified active control of metal transfer and pulsed gmaw of titanium. *Welding Journal(USA)*, 80(2):54, 2001.

[30] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, Jan 1988.

[31] Z. Wang and Y. Zhang. Image processing algorithm for automated monitoring of metal transfer in double-electrode gmaw. *Meas. Sci. Technol*, 18:2048–2058, 07 2007.

[32] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[34] Q. Wang, Y. Ma, K. Zhao, and Y. Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, Apr 2020.

[35] K. Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[36] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[37] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[38] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[39] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[40] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[41] D. Stutz. Learning shape completion from bounding boxes with cad shape priors. http://davidstutz.de/, September 2017.

[42] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[43] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, Dec 2020.

[44] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41 – 65, 2018.

[45] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[47] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[49] Huang Y. Weglowski M. S. and Zhang Y.M. An investigation of metal transfer process in gmaw. *Institute of Fundamental Technologies Research*, 56:345–362, 2008.

[50] Lord Rayleigh. VI. on the capillary phenomena of jets. *Proceedings of the Royal Society of London*, 29(196-199):71–97, December 1879.

[51] S. Subramaniam and D. R. White. Effect of shield gas composition on surface tension of steel droplets in a gas-metal-arc welding arc. *Metallurgical and Materials Transactions B*, 32(2):313–318, Apr 2001.

[52] Ramakrishnan Mukundan and KR Ramakrishnan. *Moment functions in image analysis: theory and applications*. World Scientific, 1998.

[53] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

Appendix A

Grid search results

In this appendix the results obtained via grid search are shown. The hyperparameters changed are batch size (8, 16, 32), number of filters (8, 16, 32) and learning rate (0.01, 0.005, 0.001). The rest of the parameters are fixed, namely, the loss function is the Jaccard distance, the optimizer is Adam and a maximum of 200 epochs is used with early stopping of patience 20. Furthermore, each model is validated with k-fold cross validation,therefore each model is run over 4 different and disjoint sets of training and validation.

The results are shown in the following tables, where each is for one specific pair of dataset (globular, spray) and architecture (U-Net, DeconvNet, MultiResUnet). A total of 134 different models were run. Notice that the number of models is less than all of the possible combinations of parameters, this is due to memory constraints with larger models.

Table A.1: Grid search results for globular dataset with U-Net architecture.

N	Batch size	Filters	Learning rate	Last epoch				Train loss						Val. loss					
				1	2	3	4	1	2	3	4	Mean	Std. dev.	1	2	3	4	Mean	Std. dev.
1	8	8	0.01	43	100	60	67	0.18	0.13	0.17	0.19	0.16	0.028	0.26	0.23	0.30	0.27	0.27	0.031
2	8	8	0.005	56	82	54	63	0.17	0.13	0.15	0.16	0.15	0.017	0.23	0.25	0.29	0.24	0.25	0.028
3	8	8	0.001	127	56	73	55	0.09	0.16	0.12	0.14	0.13	0.030	0.19	0.25	0.18	0.22	0.21	0.029
4	8	16	0.01	73	64	60	92	0.13	0.14	0.14	0.11	0.13	0.013	0.21	0.22	0.31	0.20	0.23	0.048
5	8	16	0.005	69	73	55	70	0.14	0.14	0.14	0.14	0.14	0.002	0.19	0.22	0.22	0.22	0.21	0.015
6	8	16	0.001	70	70	56	71	0.12	0.12	0.14	0.12	0.13	0.009	0.21	0.20	0.24	0.19	0.21	0.021
7	8	32	0.01	65	107	65	57	0.12	0.12	0.13	0.14	0.12	0.009	0.25	0.20	0.22	0.22	0.22	0.019
8	8	32	0.005	63	56	71	96	0.12	0.14	0.13	0.13	0.13	0.008	0.29	0.20	0.29	0.26	0.26	0.044
9	8	32	0.001	59	81	65	49	0.15	0.11	0.12	0.16	0.13	0.021	0.24	0.18	0.20	0.21	0.21	0.026
10	16	8	0.01	66	94	44	143	0.16	0.15	0.17	0.10	0.15	0.031	0.22	0.23	0.23	0.18	0.21	0.023
11	16	8	0.005	75	79	54	57	0.15	0.14	0.16	0.15	0.15	0.009	0.20	0.22	0.23	0.22	0.22	0.011
12	16	8	0.001	59	73	84	107	0.14	0.14	0.11	0.13	0.13	0.013	0.24	0.23	0.18	0.19	0.21	0.028
13	16	16	0.01	132	75	83	60	0.09	0.11	0.13	0.14	0.12	0.022	0.20	0.21	0.25	0.20	0.21	0.025
14	16	16	0.005	75	134	51	61	0.14	0.09	0.16	0.14	0.13	0.031	0.23	0.18	0.22	0.20	0.21	0.022
15	16	16	0.001	61	68	104	47	0.14	0.13	0.10	0.13	0.12	0.020	0.20	0.20	0.19	0.23	0.21	0.018
16	16	32	0.01	74	55	51	81	0.12	0.15	0.16	0.11	0.14	0.025	0.19	0.19	0.25	0.22	0.21	0.029
17	16	32	0.005	123	77	110	94	0.08	0.13	0.12	0.11	0.11	0.022	0.20	0.20	0.23	0.20	0.21	0.015
18	16	32	0.001	95	82	118	67	0.10	0.11	0.08	0.12	0.10	0.015	0.16	0.19	0.18	0.23	0.19	0.028
19	32	8	0.01	155	95	83	68	0.10	0.13	0.14	0.16	0.13	0.025	0.20	0.20	0.21	0.21	0.20	0.006
20	32	8	0.005	113	85	58	85	0.13	0.13	0.14	0.13	0.13	0.008	0.22	0.20	0.23	0.19	0.21	0.019
21	32	8	0.001	93	106	82	85	0.11	0.11	0.12	0.12	0.12	0.007	0.19	0.19	0.21	0.23	0.20	0.020
22	32	16	0.01	80	49	107	85	0.13	0.19	0.12	0.13	0.14	0.031	0.19	0.26	0.19	0.22	0.21	0.032
23	32	16	0.005	101	97	68	124	0.12	0.13	0.16	0.10	0.13	0.024	0.22	0.17	0.21	0.19	0.20	0.020
24	32	16	0.001	46	69	72	61	0.14	0.13	0.11	0.13	0.13	0.013	0.22	0.36	0.33	0.24	0.29	0.069
25	32	32	0.01	95	94	138	125	0.11	0.11	0.09	0.11	0.11	0.013	0.23	0.20	0.18	0.20	0.20	0.019
26	32	32	0.005	99	145	115	107	0.10	0.08	0.09	0.09	0.09	0.006	0.18	0.21	0.22	0.20	0.20	0.018

Table A.2: Grid search results for globular dataset with DeconvNet architecture.

N	Batch size	Filters	Learning rate	Last epoch				Train loss						Val. loss					
				1	2	3	4	1	2	3	4	Mean	Std. dev.	1	2	3	4	Mean	Std. dev.
27	8	8	0.010	75	83	189	91	0.28	0.26	0.25	0.28	0.26	0.016	0.44	0.41	0.43	0.42	0.43	0.014
28	8	8	0.005	134	73	70	137	0.21	0.29	0.29	0.21	0.25	0.047	0.45	0.41	0.38	0.39	0.40	0.031
29	8	8	0.001	154	74	94	74	0.16	0.21	0.20	0.23	0.20	0.030	0.35	0.35	0.43	0.37	0.37	0.038
30	8	16	0.010	21	98	93	61	1.00	0.28	0.26	0.32	0.46	0.358	1.00	0.53	0.43	0.44	0.60	0.269
31	8	16	0.005	93	98	138	80	0.26	0.22	0.17	0.30	0.24	0.053	0.46	0.40	0.39	0.42	0.42	0.030
32	8	16	0.001	68	90	89	105	0.24	0.19	0.20	0.18	0.20	0.025	0.35	0.36	0.38	0.38	0.37	0.013
33	8	32	0.010	109	93	21	188	0.25	0.23	0.96	0.22	0.42	0.364	0.38	0.43	0.96	0.37	0.53	0.285
34	8	32	0.005	95	98	21	72	0.21	0.22	0.96	0.27	0.41	0.365	0.45	0.38	0.96	0.41	0.55	0.273
35	8	32	0.001	106	126	95	89	0.20	0.17	0.20	0.22	0.20	0.019	0.39	0.35	0.37	0.40	0.38	0.023
36	16	8	0.010	101	99	109	63	0.24	0.20	0.22	0.34	0.25	0.062	0.47	0.38	0.43	0.56	0.46	0.077
37	16	8	0.005	62	57	21	77	0.26	0.33	1.00	0.22	0.45	0.367	0.46	0.46	1.00	0.39	0.58	0.284
38	16	8	0.001	93	62	60	69	0.19	0.26	0.24	0.26	0.24	0.034	0.40	0.35	0.40	0.48	0.41	0.055
39	16	16	0.010	51	30	105	21	0.35	1.00	0.21	0.96	0.63	0.410	0.53	1.00	0.43	0.96	0.73	0.291
40	16	16	0.005	133	56	77	82	0.17	0.30	0.28	0.23	0.24	0.061	0.37	0.56	0.48	0.45	0.46	0.082
41	16	16	0.001	72	97	58	64	0.24	0.20	0.26	0.26	0.24	0.027	0.46	0.41	0.40	0.49	0.44	0.042
42	16	32	0.010	79	21	21	21	0.23	0.96	0.96	1.00	0.79	0.375	0.46	0.96	0.96	1.00	0.84	0.259
43	16	32	0.005	59	92	28	63	0.28	0.21	1.00	0.29	0.44	0.372	0.49	0.49	1.00	0.48	0.61	0.257
44	16	32	0.001	48	53	64	85	0.26	0.28	0.25	0.19	0.24	0.038	0.47	0.46	0.45	0.41	0.45	0.025
45	32	8	0.010	77	51	145	51	0.24	0.29	0.17	0.27	0.24	0.054	0.46	0.53	0.38	0.60	0.49	0.092
46	32	8	0.005	132	60	54	91	0.17	0.23	0.29	0.22	0.23	0.049	0.36	0.41	0.60	0.42	0.45	0.106
47	32	8	0.001	151	52	75	89	0.15	0.30	0.20	0.18	0.21	0.065	0.34	0.38	0.38	0.32	0.35	0.029
48	32	16	0.010	97	125	60	111	0.20	0.19	0.23	0.19	0.20	0.019	0.42	0.37	0.41	0.34	0.39	0.039
49	32	16	0.005	113	24	140	100	0.18	1.00	0.17	0.19	0.39	0.409	0.44	1.00	0.38	0.40	0.55	0.298
50	32	16	0.001	120	50	48	97	0.17	0.22	0.32	0.17	0.22	0.069	0.35	0.41	0.56	0.34	0.41	0.100
51	32	32	0.010	21	82	21	21	0.96	0.26	0.96	0.96	0.79	0.350	0.96	0.58	0.96	0.96	0.87	0.188
52	32	32	0.005	21	21	153	21	0.96	0.96	0.15	0.96	0.76	0.405	0.96	0.96	0.36	0.96	0.81	0.302
53	32	32	0.001	50	77	49	103	0.22	0.23	0.27	0.19	0.23	0.032	0.44	0.41	0.53	0.35	0.43	0.077

Table A.3: Grid search results for globular dataset with MultiResUnet architecture.

N	Batch size	Filters	Learning rate	Last epoch				Train loss						Val. loss					
				1	2	3	4	1	2	3	4	Mean	Std. dev.	1	2	3	4	Mean	Std. dev.
54	8	8	0.010	24	41	46	63	0.52	0.50	0.50	0.50	0.50	0.008	0.68	0.39	0.54	0.48	0.52	0.117
55	8	8	0.005	32	48	58	65	0.50	0.51	0.50	0.49	0.50	0.008	0.77	0.69	0.55	0.68	0.67	0.090
56	8	8	0.001	114	115	71	92	0.48	0.48	0.49	0.48	0.48	0.005	0.55	0.53	0.64	0.52	0.56	0.055
57	8	16	0.010	31	46	50	39	0.51	0.50	0.50	0.50	0.50	0.006	0.59	0.57	0.56	0.61	0.58	0.024
58	8	16	0.005	31	30	60	30	0.50	0.50	0.49	0.51	0.50	0.009	0.67	0.56	0.51	0.60	0.59	0.068
59	8	16	0.001	73	88	84	104	0.48	0.48	0.49	0.47	0.48	0.005	0.55	0.53	0.57	0.60	0.56	0.030
60	8	32	0.010	39	28	29	41	0.50	0.52	0.52	0.51	0.51	0.011	0.56	0.57	0.60	0.96	0.67	0.195
61	8	32	0.005	30	46	31	76	0.51	0.49	0.50	0.48	0.50	0.009	0.55	0.57	0.56	0.61	0.57	0.025
62	8	32	0.001	111	106	94	116	0.47	0.48	0.49	0.47	0.48	0.006	0.53	0.56	0.52	0.58	0.55	0.025
63	16	8	0.010	58	54	21	37	0.49	0.49	0.53	0.50	0.50	0.016	0.52	0.60	0.84	0.53	0.62	0.149
64	16	8	0.005	87	38	78	38	0.48	0.50	0.48	0.50	0.49	0.010	0.86	0.71	0.52	0.65	0.69	0.143
65	16	8	0.001	149	117	119	105	0.48	0.48	0.47	0.49	0.48	0.006	0.56	0.53	0.58	0.58	0.56	0.025
66	16	16	0.010	49	34	41	39	0.49	0.51	0.50	0.50	0.50	0.009	0.52	0.54	0.50	0.51	0.52	0.018
67	16	16	0.005	37	51	79	37	0.50	0.49	0.48	0.50	0.49	0.008	0.74	0.53	0.57	0.64	0.62	0.091
68	16	16	0.001	111	172	134	39	0.48	0.47	0.47	0.77	0.55	0.151	0.55	0.57	0.54	0.78	0.61	0.114

Table A.4: Grid search results for spray dataset with U-Net architecture.

N	Batch size	Filters	Learning rate	Last epoch				Train loss						Val. loss					
				1	2	3	4	1	2	3	4	Mean	Std. dev.	1	2	3	4	Mean	Std. dev.
69	8	8	0.01	88	64	37	90	0.24	0.25	0.26	0.24	0.25	0.012	0.28	0.30	0.39	0.29	0.31	0.053
70	8	8	0.005	74	91	105	122	0.24	0.24	0.21	0.21	0.23	0.017	0.28	0.29	0.28	0.29	0.28	0.008
71	8	8	0.001	113	97	74	114	0.19	0.19	0.23	0.21	0.20	0.019	0.28	0.29	0.28	0.28	0.28	0.006
72	8	16	0.01	52	119	90	57	0.27	0.21	0.22	0.24	0.23	0.027	0.31	0.28	0.30	0.29	0.29	0.013
73	8	16	0.005	144	119	93	111	0.19	0.19	0.21	0.19	0.19	0.010	0.27	0.28	0.28	0.29	0.28	0.009
74	8	16	0.001	86	70	118	49	0.21	0.21	0.19	0.25	0.21	0.025	0.31	0.29	0.26	0.33	0.30	0.030
75	8	32	0.01	93	68	89	99	0.18	0.21	0.22	0.18	0.20	0.018	0.29	0.32	0.29	0.30	0.30	0.013
76	8	32	0.005	114	92	80	100	0.19	0.20	0.20	0.22	0.20	0.012	0.26	0.31	0.29	0.29	0.29	0.021
77	8	32	0.001	86	114	99	105	0.21	0.19	0.19	0.17	0.19	0.014	0.28	0.26	0.28	0.29	0.28	0.010
78	16	8	0.01	69	83	56	136	0.23	0.23	0.27	0.21	0.24	0.024	0.29	0.33	0.34	0.34	0.32	0.026
79	16	8	0.005	55	61	62	88	0.26	0.24	0.25	0.23	0.25	0.011	0.28	0.37	0.33	0.27	0.31	0.046
80	16	8	0.001	61	118	91	74	0.24	0.19	0.21	0.24	0.22	0.023	0.30	0.27	0.29	0.30	0.29	0.016
81	16	16	0.01	109	70	87	86	0.19	0.23	0.22	0.24	0.22	0.024	0.31	0.29	0.30	0.29	0.30	0.010
82	16	16	0.005	120	58	119	92	0.19	0.24	0.19	0.24	0.21	0.028	0.30	0.33	0.31	0.27	0.30	0.026
83	16	16	0.001	69	105	111	82	0.22	0.18	0.18	0.20	0.20	0.021	0.33	0.27	0.28	0.27	0.29	0.027
84	16	32	0.01	82	80	64	44	0.21	0.21	0.20	0.25	0.22	0.023	0.32	0.27	0.32	0.33	0.31	0.026
85	16	32	0.005	150	84	151	115	0.20	0.21	0.17	0.20	0.19	0.018	0.31	0.28	0.29	0.26	0.28	0.020
86	16	32	0.001	83	48	56	49	0.21	0.24	0.22	0.24	0.23	0.017	0.29	0.37	0.33	0.28	0.32	0.041
87	32	8	0.01	101	72	142	114	0.23	0.25	0.22	0.22	0.23	0.017	0.29	0.30	0.30	0.31	0.30	0.008
88	32	8	0.005	99	152	77	64	0.23	0.18	0.25	0.25	0.23	0.030	0.37	0.28	0.31	0.29	0.31	0.040
89	32	8	0.001	129	43	64	46	0.19	0.32	0.28	0.31	0.28	0.061	0.30	0.36	0.38	0.35	0.35	0.034
90	32	16	0.01	89	138	119	93	0.24	0.21	0.20	0.21	0.22	0.017	0.33	0.30	0.31	0.30	0.31	0.012
91	32	16	0.005	78	65	140	155	0.24	0.25	0.20	0.19	0.22	0.033	0.29	0.38	0.28	0.27	0.31	0.049
92	32	16	0.001	80	102	94	37	0.21	0.21	0.21	0.35	0.24	0.071	0.31	0.34	0.30	0.57	0.38	0.127

Table A.5: Grid search results for spray dataset with Deconvnet architecture.

N	Batch size	Filters	Learning rate	Last epoch				Train loss						Val. loss					
				1	2	3	4	1	2	3	4	Mean	Std. dev.	1	2	3	4	Mean	Std. dev.
93	8	8	0.01	53	87	91	75	0.56	0.52	0.43	0.51	0.51	0.055	0.65	0.62	0.53	0.63	0.60	0.053
94	8	8	0.005	98	100	140	98	0.44	0.37	0.34	0.38	0.38	0.045	0.63	0.48	0.49	0.49	0.52	0.072
95	8	8	0.001	63	90	76	80	0.34	0.29	0.31	0.33	0.32	0.021	0.50	0.47	0.46	0.46	0.47	0.016
96	8	16	0.01	56	66	74	71	0.59	0.57	0.56	0.61	0.58	0.023	0.62	0.63	0.62	0.63	0.62	0.006
97	8	16	0.005	45	58	101	43	0.57	0.58	0.60	0.57	0.58	0.014	0.62	0.64	0.63	0.63	0.63	0.008
98	8	16	0.001	94	75	62	111	0.30	0.43	0.33	0.31	0.34	0.062	0.45	0.64	0.46	0.47	0.50	0.088
99	8	32	0.01	172	96	79	53	0.41	0.64	0.62	0.59	0.56	0.106	0.53	0.65	0.63	0.62	0.61	0.053
100	8	32	0.005	127	199	45	120	0.36	0.57	0.59	0.56	0.52	0.107	0.51	0.59	0.63	0.64	0.59	0.057
101	8	32	0.001	53	61	78	103	0.54	0.46	0.33	0.31	0.41	0.108	0.70	0.64	0.48	0.45	0.57	0.123
102	16	8	0.01	124	142	56	127	0.31	0.30	0.53	0.36	0.37	0.108	0.50	0.52	0.63	0.49	0.54	0.068
103	16	8	0.005	123	115	125	118	0.32	0.29	0.33	0.30	0.31	0.015	0.48	0.47	0.52	0.47	0.48	0.023
104	16	8	0.001	76	94	86	105	0.30	0.24	0.27	0.28	0.27	0.024	0.48	0.49	0.45	0.48	0.47	0.018
105	16	16	0.01	128	97	52	107	0.33	0.32	0.53	0.32	0.38	0.103	0.47	0.46	0.76	0.44	0.53	0.154
106	16	16	0.005	110	107	46	75	0.30	0.29	0.53	0.31	0.36	0.114	0.47	0.47	0.63	0.45	0.50	0.084
107	16	16	0.001	85	101	80	69	0.28	0.26	0.28	0.28	0.27	0.013	0.45	0.46	0.44	0.44	0.45	0.008
108	16	32	0.01	21	102	33	23	1.00	0.40	0.62	1.00	0.75	0.299	1.00	0.47	0.81	1.00	0.82	0.250
109	16	32	0.005	118	68	116	65	0.33	0.61	0.44	0.53	0.48	0.119	0.44	0.65	0.62	0.62	0.58	0.096
110	16	32	0.001	103	94	96	89	0.27	0.29	0.36	0.26	0.29	0.045	0.48	0.50	0.54	0.45	0.49	0.040
111	32	8	0.01	59	69	88	83	0.48	0.36	0.31	0.34	0.37	0.072	0.64	0.67	0.49	0.58	0.60	0.080
112	32	8	0.005	69	101	80	62	0.46	0.32	0.33	0.32	0.36	0.068	0.63	0.49	0.53	0.50	0.54	0.063
113	32	8	0.001	63	64	48	124	0.28	0.40	0.42	0.29	0.35	0.069	0.46	0.60	0.56	0.53	0.54	0.057
114	32	16	0.01	80	97	95	97	0.37	0.30	0.34	0.33	0.34	0.032	0.63	0.65	0.53	0.60	0.60	0.056
115	32	16	0.005	76	46	48	84	0.38	0.49	0.49	0.36	0.43	0.067	0.56	0.67	0.77	0.58	0.65	0.097
116	32	16	0.001	63	52	63	73	0.34	0.37	0.32	0.29	0.33	0.032	0.60	0.53	0.47	0.46	0.51	0.068
117	32	32	0.01	77	109	60	23	0.54	0.40	0.53	1.00	0.62	0.264	0.66	0.51	0.65	1.00	0.71	0.206
118	32	32	0.005	41	83	50	91	0.52	0.36	0.58	0.34	0.45	0.117	0.63	0.52	0.72	0.50	0.59	0.104
119	32	32	0.001	65	69	75	74	0.33	0.29	0.32	0.33	0.32	0.017	0.51	0.48	0.56	0.52	0.52	0.032

Table A.6: Grid search results for spray dataset with MultiResUnet architecture.

N	Batch size	Filters	Learning rate	Last epoch				Train loss						Val. loss					
				1	2	3	4	1	2	3	4	Mean	Std. dev.	1	2	3	4	Mean	Std. dev.
120	8	8	0.01	46	38	49	63	0.39	0.39	0.37	0.35	0.37	0.020	0.46	0.50	0.51	0.43	0.48	0.036
121	8	8	0.005	58	37	39	49	0.37	0.38	0.37	0.37	0.37	0.006	0.68	0.72	0.47	0.77	0.66	0.132
122	8	8	0.001	134	113	92	109	0.31	0.30	0.32	0.31	0.31	0.009	0.47	0.46	0.45	0.44	0.46	0.014
123	8	16	0.01	57	43	72	42	0.35	0.38	0.37	0.37	0.37	0.011	0.48	0.38	0.46	0.52	0.46	0.058
124	8	16	0.005	94	43	64	47	0.33	0.36	0.35	0.36	0.35	0.013	0.42	0.42	0.46	0.42	0.43	0.021
125	8	16	0.001	134	129	108	25	0.31	0.31	0.30	0.84	0.44	0.268	0.43	0.44	0.44	0.84	0.54	0.201
126	8	32	0.01	44	50	61	50	0.36	0.35	0.35	0.37	0.36	0.010	0.64	0.42	0.44	0.42	0.48	0.110
127	8	32	0.005	76	68	61	75	0.32	0.32	0.33	0.32	0.32	0.007	0.41	0.39	0.48	0.43	0.43	0.038
128	8	32	0.001	25	91	124	29	0.84	0.30	0.32	0.79	0.56	0.296	0.83	0.44	0.51	0.80	0.65	0.200
129	16	8	0.01	53	70	62	59	0.37	0.37	0.36	0.36	0.37	0.006	0.75	0.44	0.39	0.90	0.62	0.247
130	16	8	0.005	53	107	78	100	0.36	0.32	0.34	0.33	0.34	0.017	0.50	0.49	0.40	0.42	0.45	0.049
131	16	8	0.001	155	149	30	30	0.30	0.30	0.91	0.91	0.61	0.356	0.52	0.47	0.90	0.93	0.71	0.241
132	16	16	0.01	42	62	44	89	0.36	0.34	0.37	0.34	0.35	0.015	0.41	0.48	0.92	0.49	0.57	0.233
133	16	16	0.005	79	66	105	95	0.32	0.33	0.32	0.32	0.32	0.004	0.45	0.72	0.65	0.45	0.57	0.138
134	16	16	0.001	34	31	30	180	0.91	0.91	0.92	0.28	0.75	0.316	0.91	0.91	0.92	0.47	0.80	0.225

Appendix B

Post processing

In this appendix are described the postprocessing techniques used after the segmentation of images to compute kinematic and geometric properties. These techniques are mainly from computer vision and signal processing.

B.1 Calculation of properties

B.1.1 Geometry

A useful concept in computer vision is the *moment*, which is a specific weighted average computed based on pixel intensity values of an image. Although a moment is a purely mathematical device, some moments have useful interpretations that can describe the image.

The n^{th} moment of a function f around point c is defined as

$$\mu_n = \int_{-\infty}^{\infty} (x - c)^n f(x) \mathrm{d}x$$

which can be extended to the 2D case as follows

$$\mu_{mn} = \int \int_{-\infty}^{\infty} (x - c_x)^m (y - c_y)^n f(x, y) \mathrm{d}y \mathrm{d}x$$

Then, a discrete version is defined in order to work with pixels. Also, notice that c_x and c_y are arbitrary constants which can be set to 0 to simplify the expression.

$$\mu_{mn} = \sum_x \sum_y x^m y^n f(x, y)$$

In this case, the function $f(x, y)$ yields the pixel intensity values evaluated at coordinates (x, y) of an image.

There are two properties that will be obtained using moment calcualtions: area and centroid. Notice that for a binary image of size (w, h) the zeroth moment is given by

$$\mu_{00} = \sum_{x=0}^{h} \sum_{y=0}^{w} x^0 y^0 f(x, y) = \sum_{x=0}^{h} \sum_{y=0}^{w} f(x, y)$$

which is just the sum of every pixel intensity across the image, i.e., its area. In a similar fashion, it can be proven[1] that the centroid is given by

$$\{\vec{x}, \vec{y}\} = \left\{ \frac{\mu_{10}}{\mu_{00}}, \frac{\mu_{01}}{\mu_{00}} \right\}$$

Finally, it is also possible to measure the outline of a contour by counting its bordering pixels, that is perimeter which can be done using the `opencv` library.

B.1.2 Physical properties

In this thesis the images come from videos, so the centroids can then be used to compute velocity and acceleration from consecutive frames simply by taking frame to frame distance and time differences if the frame rate and pixel to distance conversion are available (see Chapter 5 for these conversions). In particular, numerical approximations using central differences are used, in which the derivative of a function at point x is

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{f(x+h) - f(x-h)}{2h}$$

where h is a sufficiently small number. In this case, h is the sampling period and $f(x)$ is the position or velocity of the centroid at every sampling time. In the borders it is not possible to apply central differences, so forward and backward differentiation is used at the start and the end of the data respectively.

Another relevant property that can be obtained is the surface tension of the droplet using Rayleigh's formula [50]

$$\sigma = \frac{3\pi\rho V}{8\tau^2} \tag{B.1}$$

where ρ is the droplet's density, V is the volume and τ is the oscillation period during free flight. Values for ρ can be found in literature [51], the volume can be approximated assuming a free flight droplet is a prolate ellipsoid, that is the 2D droplet is an ellipse and the depth is equal to the semi-minor axis of the ellipse in the 2D plane. Then, the volume can be obtained by

$$V = \frac{4}{3}\pi a^2 b \tag{B.2}$$

where a is the semi-minor axis and b is the semi-major axis. The ellipse's parameters are obtained with `opencv` which can fit an ellipse to a contour through a least squares approximation returning the center coordinates, axes lengths (major and minor) and angle with respect to the horizontal axis.

The remaining parameter needed for the surface tension is the period of oscillation and it can be approximated with Fourier analysis. Specifically, free flight time windows are taken from the perimeter curve, that is the time between the droplet detachment and landing on

[1] An in depth analysis of moments and their applications in images can be found in *Mukundan* [52].

the weld pool. Then, the signal can be processed using the Fast Fourier Transform (FFT) to extract its principal frequencies. Furthermore, the period can be calculated simply by taking the inverse of the frequency. In order to be able to apply this method of analysis the sampling must satisfy the Nyquist-Shanon sampling theorem which states that the frequencies that can be reconstructed are, at most, half the sampling frequency [53]

$$f^* < \frac{f_s}{2}$$

where f_s is the sampling frequency and f^* are the frequencies for which perfect reconstruction is guaranteed. In this case, the sampling frequency is known and has a value of 3 kHz, so any frequency below 1.5 $[kHz]$ can be accurately detected by Fourier analysis[2].

B.2 Signal smoothing

The properties obtained are usually noisy, so a smoother signal is useful to proceed with some calculations. This is done by convolving the 1D signal (e.g. area of a droplet in each frame) with a window function. Specifically, the Hanning window is used which is defined as

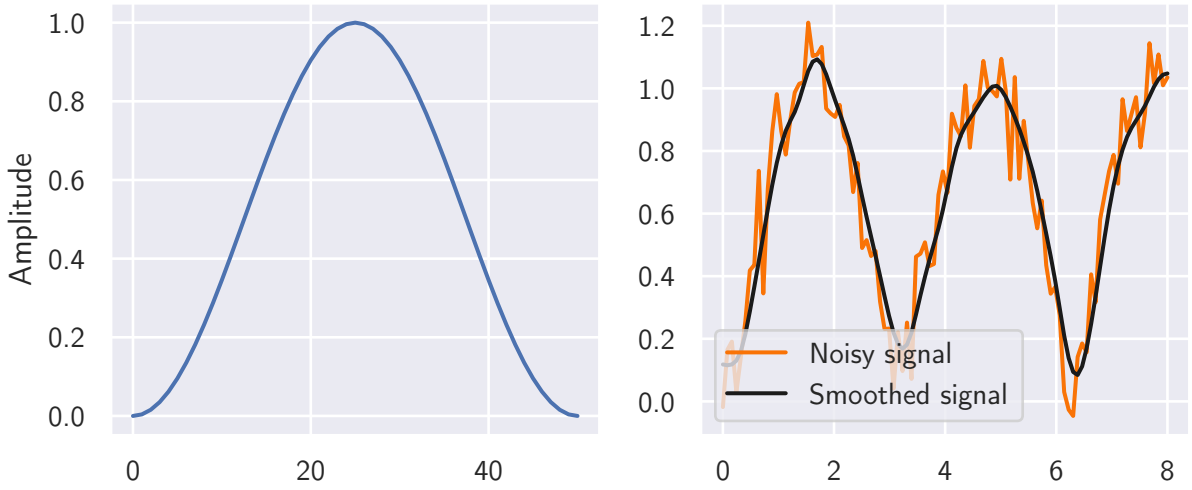
$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{M-1}\right), \quad 0 \leq n \leq M-1$$

where M is the size of the window and n are the points along the window.

In figure B.1 the hanning window is shown as well as the result when smoothing a noisy sine wave. It can be seen that signal smoothing is useful to highlight the major peaks in the original signal and attenuating noise. The window is effectively computing a weighted average at each step, a flat window would be a moving average smoothing. The implementation is done as documented by `scipy`[3]. It is worth mentioning that the window size should be chosen in relation to the pattern of interest. A window that is too wide might obfuscate relevant information of short cycle variations while a short window may not reduce the noise enough. Notice that in the context of this thesis, the variations are not noise and rather are just the inherent variation of calculated parameters. Regardless, the smoothing of the signal is useful for specific purposes.

[2]Higher frequencies could appear in the spectrum due to aliasing if they are multiples of the ones found through Fourier analysis.

[3]`https://scipy-cookbook.readthedocs.io/items/SignalSmooth.html`



(a) Hanning window

(b) Signal smoothing

Figure B.1: Examples of a hanning window of size 51 (a) and signal smoothing of a noisy sine signal (b). The hanning window is moved across the signal and at each step one-dimensional convolution is computed.