Deutsches Zentrum
DLR für Luft- und Raumfahrt

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Artificial Neural Networks for Individual Tracking and Characterization of Wake Vortices in LiDAR Measurements

Masterthesis

Lars Stietz

INSTITUT FÜR
MATHEMATIK

November 7, 2022

Studiengang:	Technomathematik
Matrikelnummer:	6965877
Erstprüfer:	Prof. Dr. Daniel Ruprecht
Zweitprüfer:	MEng Niklas Wartha (DLR)

Eidestattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Technomathematik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, 07.11.2022

Ort, Datum

[illegible]

Unterschrift

Abstract

The main restricting factor of airport capacity are aircraft separations. These separations exist to avoid potentially hazardous wake vortex encounters (WVEs). Especially during the final approach, this hazard is of great concern since aircraft follow the same glide path. The severity of wake vortex encounters depends on the generating and the encountering aircraft, thus dynamic pairwise aircraft separations, which adapt depending on prevailing weather conditions, are desired. Light Detection and Ranging (LiDAR) scans are suggested for monitoring Wake Vortex Advisory Systems due to their fast-time strength and location characterization of wake vortices. The first approaches to automating such characterizations were made with multi-layer perceptrons (MLP) and convolutional neural networks (CNN). Those were shown to be sufficient for wake vortex characterization but could not yet compete with traditional methods in terms of accuracy. For that reason, this work proposes a machine-learning pipeline that uses bounding box predictions by a YOLOv4 network to restrict the input to single vortices for the following CNN to achieve higher accuracy. The LiDAR scans used for training contain radial velocity measurements made at Vienna International Airport. After preprocessing and testing feature engineering, those LiDAR scans are transformed into images as required input for YOLOv4. Afterward, the bounding box predictions are used to cut out individual vortices from the original scans. The individual vortices are then used to train a CNN to enhance localization and the vortex strength estimation further. The evaluation shows that a prediction pipeline is superior to a single CNN approach. The localization error was decreased by more than 90% and the vortex strength estimation by up to 31% to a localization error as low as 2.87 m and a vortex strength error as low as 20.88. Furthermore, the precision of detecting hazardous wake vortices was increased by 7.51% to gain a precision of 96.11%. This pipeline can be executed while maintaining a sufficiently low computation time.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

One of the most critical flight phases of an aircraft is its final approach. Close vicinity to the ground can cause severe accidents when the aircraft is subject to unexpected, abrupt disturbances. Wake vortices generated by previous aircraft hovering above the runway are one type of disturbance which can cause this undesirable condition [1]. The possibility of a wake vortex encounter (WVE) is given during any flight stage for an aircraft [2]. Nevertheless, the most problematic flight stage is during the approach and landing since the proximity to the ground leaves little room for flight path corrections. A WVE can induce a rolling moment onto the aircraft. The severity of impact and the resulting rolling moment is influenced by factors such as through which part of the vortex the aircraft flies. WVEs are not unusual nor unlikely, given that aircraft commonly follow similar flight paths during approach - unlike for take-offs. In the worst case, an encounter can lead to a severe accident of the aircraft. In January 2017 a CL604 lost approximately 9 000 ft of altitude after encountering wake vortices of an A380 that passed overhead in the opposite direction [3].

The severity of a WVE depends not only on the encountering aircraft but also on the aircraft generating the vortices. When the Boeing 747 came into service in the 1970s, serious problems with wake vortices were first considered [4]. To mitigate the possibilities of a WVE, the International Civil Aviation Organization (ICAO) introduced landing separations that take wake vortices into account [4]. Three aircraft categories were introduced, light, medium, and heavy, where the following and leading aircraft pair categories determine the separation. Studies have shown that these landing separations are too conservative for a variety of meteorological situations [4] and leads to a capacity bottleneck and congestion at airports. Although the traffic volume has declined due to the COVID-19 pandemic, it is expected to recover until 2025, or latest, 2027 [5, 6]. With a five-year delay, we can expect roughly 1.5 million unaccommodated flights by 2045, according to Eurocontrol predictions from 2018 [7].

To address this problem, several ideas exist to increase airport capacity. The most forward solution would be to increase the number of runways. With new runways, substantial environmental issues arise. Thus an approach to increase the capacity is to reduce the aircraft separation standards and adjust them to different scenarios, while maintaining or even increasing safety standards. A recategorization program (RECAT) for that need was introduced by the Federal Aviation Administration (FAA) that aims to refine aircraft categorization by including more involved wake-related parameters [8]. The RECAT program follows three phases, where the ultimate goal is to achieve dynamic pairwise aircraft separations [8]. In the first phase, six new aircraft categories are introduced, taking into account the strength of the generated wake vortices and the vulnerability of the following aircraft besides only using the aircraft's weight. The second phase introduces static pairwise aircraft separations, and phase three aims at turning these separations dynamic, adjusting the separations to prevailing meteorological conditions [8]. An airport relies on real-time knowledge of the position and strength of wake vortices to achieve dynamic pairwise separations. For that reason, a Wake Vortex Monitoring System (WVMS) is desirable.

Furthermore, the development of devices to reduce the strength and durability of wake vortices could lead to a decrease in aircraft separation. To evaluate the effectiveness of such devices, measurement campaigns are conducted. For example, a measurement campaign at Vienna International Airport was conducted to measure the success of so-called plate lines [9]. With current methods, the evaluation of large data sets takes a long time. Thus, the need for automatic fast-time evaluation methods is given.

Implementing a fast-time automatic wake vortex parameter characterization algorithm could evaluate these wake vortex measurements and be included in WVMSs. In recent years the focus shifted from traditional algorithms to machine learning approaches to achieve fast-time computations. A first approach on the characterization of wake vortex parameters was done in [10].

1.2 Goal

The goal of this thesis is to make use of the first approach of wake vortex parameter characterization with a convolutional neural network (CNN) from [10] and combine this idea with new approaches to wake vortex detection using artificial neural networks (ANNs) for object detection. We aim to develop a prediction pipeline that first detects single vortices and then characterizes these individually. A sketch of the prediction pipeline can be found in Figure 1. The object detection and characterization accuracy are studied individually and, in the end, compared to the CNN only approach of [10].

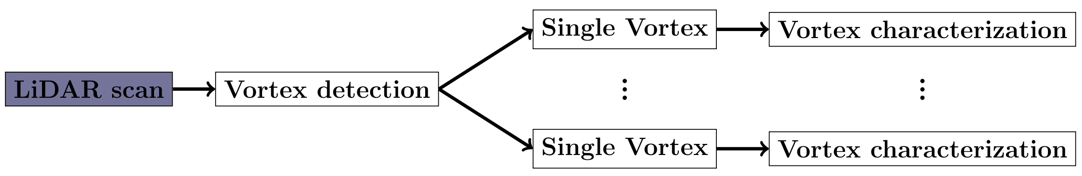


Figure 1: Sketch of the prediction pipeline.

1.3 Overview

In chapter 2, a brief introduction to the fluid dynamics, basic wake vortex principles and measurement techniques of wake vortices is given.

A literature review in chapter 3 gives rise to the open research questions of this thesis based on traditional wake vortex detection algorithms, and recent ANN approaches to the problem.

After that, we continue with the essential machine learning tools necessary for this thesis in chapter 4. An introduction to artificial neural networks (ANNs), particularly convolutional neural networks (CNNs), is given. We describe the basic idea behind object detection networks, and continue with the essential machine learning tools needed for the work on this thesis in chapter 4.

In the following chapter 5, the data sets and the ANNs we work with are introduced. The network architecture and essential components of YOLOv4 are explained. The same goes for the CNN regression we use.

A detailed evaluation of the networks is given in chapter 6 for the YOLOv4 network and in chapter 7 for the CNN regression. This evaluation contains tests on data preprocessing mechanisms and hyperparameter tuning. The obtained networks are validated, and the results are discussed. A comparison of the resulting prediction pipeline with state-of-the-art approaches on wake vortex characterization is given in chapter 8.

In the final chapter 9, a conclusion of the results and answers to the research questions are given. Furthermore, an outlook on future research is presented.

2 Wake Vortex Principles

One uses conservation laws to describe fluid flow with the help of a velocity field, such as conservation of mass, momentum, and energy. In the most general case, considering a viscous flow, the momentum equations are called *Navier-Stokes* (2.2) equation and for an inviscid flow *Euler equation* [11, p.135]. The continuity and Navier-Stokes equations read [12, p. 86 & 104]

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 \tag{2.1}$$

$$\rho \left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} \right) = -\nabla p + \rho \mathbf{f} + \mu \Delta \mathbf{V} + \frac{\mu}{3} \nabla \left(\nabla \cdot \mathbf{V} \right). \tag{2.2}$$

In equations (2.1) and (2.2), p describes the pressure, ρ the density, $\mathbf{V}$ the velocity, $\mathbf{f}$ body forces and μ the viscosity. To simplify those equations, we can use the assumption of incompressible flow. With the assumption of an incompressible flow, meaning a constant density, the conservation of mass or continuity equation (2.1) and Navier-Stokes eqution (2.2) reduce to [12, p. 86 & 104]

$$\nabla \cdot \mathbf{V} = 0 \tag{2.3}$$

$$\rho \left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} \right) = -\nabla p + \rho \mathbf{f} + \mu \Delta \mathbf{V}. \tag{2.4}$$

Assuming incompressibility leading to equations (2.3) and (2.4) is sufficient for a flow with speeds less than approximately 100 m/s at sea level [12, p. 86]. During aircraft approach this requirement is met [13].

2.1 Basic Vortex

In fluid dynamics, four elementary flows are considered to be superimposed to form more complex flows. Uniform, source, and doublet flow can be superimposed to obtain a non-lifting flow. To get a flow with finite lift, the *vortex flow* has to be introduced [11, p. 262]. A *potential vortex* or vortex flow is a two-dimensional flow, thus can be described by V_x and V_y as velocity components. In a vortex flow, all streamlines are concentric about a given point and constant along a given streamline. Hence, to describe the flow velocity, a radial V_r and a tangential V_θ component are used. The radial velocity is $V_r = 0$, and the tangential velocity is given by $V_\theta = C/r$ with a constant C [11, p. 262]. A potential vortex with origin O is sketched in in Figure 2.

To understand the flow of a vortex and calculate the constant C, we take the circulation Γ around a given circular streamline $\mathbf{C}$ of radius r into account. The integral can be evaluated as

$$\Gamma \equiv \oint_{\mathbf{C}} \mathbf{V} \cdot \mathrm{d}s = V_\theta (2\pi r). \tag{2.5}$$

Hence the constant is $C = \Gamma/2\pi$ and the tangential velocity $V_\theta = \Gamma/(2\pi r)$ [11, p. 263].

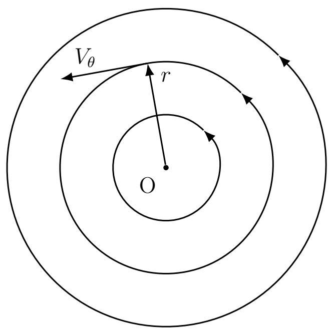


Figure 2: A sketch of a basic potential vortex flow around origin O.

With the help of Stoke's theorem, we can rewrite the integral in (2.5) even further [14] to get

$$\Gamma = \oint_C \mathbf{V} \cdot \mathrm{d}s = \int_S (\nabla \times \mathbf{V}) \mathrm{d}\mathbf{S} = \int_S \omega \cdot \mathbf{n} \mathrm{d}\mathbf{S}. \tag{2.6}$$

The curl of the velocity is called vorticity $\omega = \nabla \times \mathbf{V}$ and is used in (2.6) to abbreviate the equation even further.

2.2 Wake Vortex

To describe the flow around an airfoil, we introduce a line vortex, also called vortex filament, based on the potential vortex. A line vortex can be seen as an extension of a potential vortex perpendicular to its circulation plane. Helmholtz's vortex theorems state that the strength of a vortex filament is constant along its length, and it cannot end in a fluid [15]. Those vortex filaments can be combined to form a vortex sheet describing the airflow over an airfoil by combining with a uniform flow and replacing the airfoil's surface with a vortex sheet of variable strength $\gamma(s)$ [11, p. 328f]. The total circulation strength Γ of the vortex sheet replacing the airfoil can be calculated by adding up the circulations of the single vortex filaments $\gamma(s)$ according to (2.8). The Kutta-Joukowski lift theorem (2.7) [11, p. 239] describes how the circulation around an aircraft's wing enables it to develop lift. This theorem states that the lift force per unit span L' is directly proportional to the circulation with flight speed V_∞ and air density ρ_∞.

$$\mathrm{L}' = \Gamma V_\infty \rho_\infty \text{ ,with} \tag{2.7}$$

$$\Gamma = \int \gamma \mathrm{d}s \tag{2.8}$$

So far, we only have considered infinite airfoils. In reality, the wings of an aircraft are finite. The pressure difference between the lower pressure side and the upper suction side leads to a fluid acceleration in a spanwise direction, which leads to vortex sheets shedding off and rolling up [13]. The spanwise velocity difference at the top and bottom of the airfoil leads to the roll-up process along the entire length of the wing. Those small vortices roll up to eventually form a counter-rotating vortex pair [15]. This detachment of the vortex sheet and bending along the free stream direction leads to a so-called horseshoe vortex. To model the flow around a finite airfoil, multiple horseshoe vortices are used, as depicted in Figure 3. As Helmoltz's vortex theorem states that the strength of a vortex

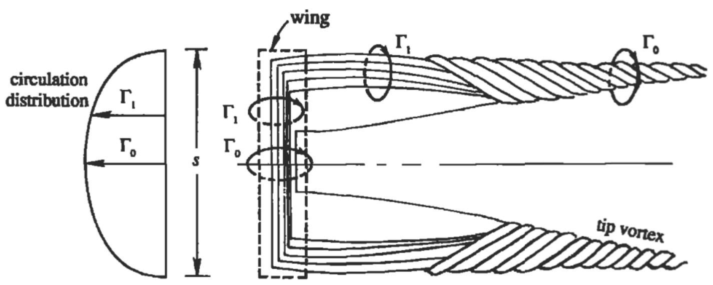


Figure 3: Sketch of a finite wing with circulation distribution and vortex sheet forming a horseshoe vortex (taken from [16, p. 646]).

filament is constant, the vortex circulation strength of the trailing vortices is the same as the initial vortex circulation Γ_0 [15]. If we consider an individual aircraft with maximum landing weight M, wing span B and final approach speed V_∞, the circulation of the generated wake vortices can be calculated by

$$\Gamma_0 = \frac{Mg}{\rho_\infty b_0 V_\infty}, \text{ with } b_0 = \frac{\pi}{4}B, \tag{2.9}$$

where ρ_∞ is the air density for the standard atmosphere at sea level and g is the gravitational constant [17]. The *initial vortex spacing* b_0, is a standard referencing length in the research on wake vortices. The equation for b_0 in (2.9) assumes an elliptically loaded wing [17]. Due to significant regions of concentrated streamwise and cross-stream perturbations on a wing, e.g., control surfaces, flaps, spoilers, landing gear, more than one vortex pair can develop [1]. We consider the combination of wing and flap tip vortices as primary wake vortices, resulting from fully extracted flaps during approach and landing [18].

The wake vortex evolution can be split into four phases [13]. The shedding of the vortex sheet and roll up of the primary vortex structures occurs in the near field. In the extended near field, the flap-tip vortex pair and wing-tip vortex pair merge to form two counter-rotating wake vortices - the primary vortices. The counter-rotating vortex pair mutually induces descent velocity in the mid to far field. The fourth phase is the decay phase. A significant reason for the decay is the so-called Crow instability [19], which comes from mutual interactions between the sinusoidally deformed vortices leading to the linking of the vortex pair, and thus the decay of the vortex strength resulting from the opposite circulation [1]. While Crow instability is a long-wavelength instability, a shortwave instability inside the vortex cores, termed elliptic instability, accelerates vortex decay further [20].

During the landing of an aircraft, another decay mechanism factors in. When an aircraft flies below an altitude of its wingspan, ground effects appear [13]. The trajectories of the descending vortices follow an outward-going hyperbola close to the ground tending to an altitude of $b_0/2$ [1]. The ground proximity of the primary vortices leads to the creation of countersign secondary vortex structures detaching from the ground and wrapping around the primary vortices [1]. The emergence of secondary vortices is caused by the formation of a boundary layer beneath each primary vortex of opposite-signed vorticity [20]. Once the adverse pressure gradient in this boundary layer is strong enough, the secondary vortices detach and wrap around the primary vortices [20]. Furthermore, the secondary vortices induce an upward motion onto the primary vortex causing a rebound effect [21].

The vortex trajectories alter in the presence of crosswinds. A crosswind, approximately equal to the initial descent speed, can cause the upwind vortex to stall over the runway [1]. A frontal view of the wake vortex trajectories of a landing aircraft can be seen in Figure 4.

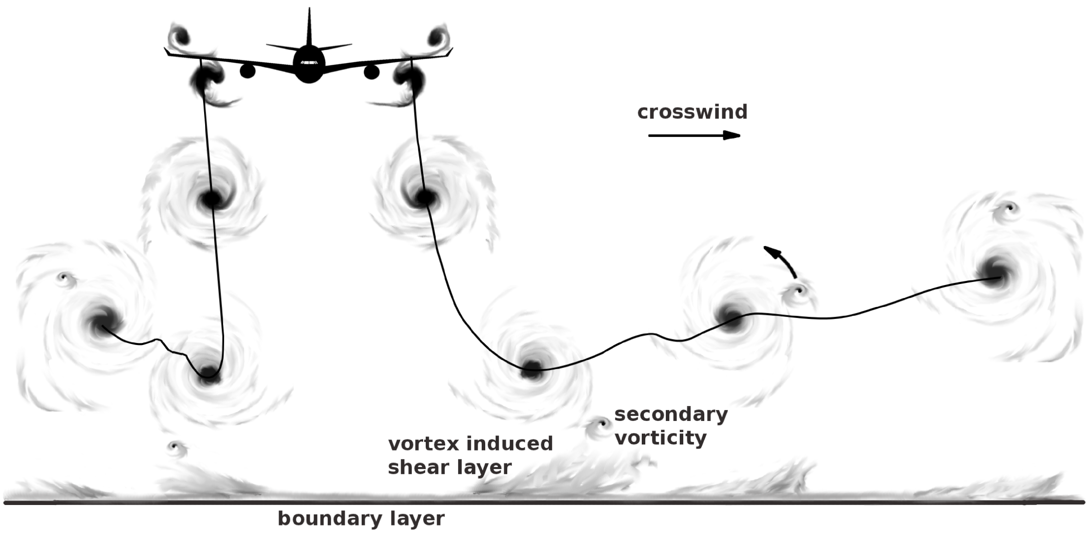


Figure 4: The trajectories of primary wake vortices and the formation of secondary vortices. The evolution is according to LES simulations from [22] and the trajectories according to [23](taken from [24]).

The creation of strong secondary vortices can be enhanced by obstacles on the ground intensifying the decay of the primary vortices [25]. This led to the development of so-called plate lines placed below the glide path of a landing aircraft in front of the runway. With simulations and field experiments, an enhanced decay of wake vortices could be verified [25, 26]. More on plate lines can be found in Section 5.1.

2.3 Measurement

Several techniques for measuring wake vortices exist. They can be split into active and passive detection methods. Radio Detection and Ranging (RADAR), Light Detection and Ranging (LiDAR), Sonic Detection and Ranging (SoDAR) represent active detection methods. Passive detection methods include microphone systems, optoacoustic and ultrasonic detection of circulation [27].

The most common techniques are RADAR and LiDAR. Pulsed coherent Doppler LiDARs (PCDLs) and Continuous Wave LiDARS (CWLs) are used for wake vortex detection. While CWLs send out a continuous wave of electromagnetic energy and measure simultaneously, PCDLs send out short bursts of electromagnetic energy and then listen for their return [27]. We use a data set measured with PCDLs. In wake vortex application PCDLs are preferred over CWLs, given the higher possible spatiotemporal resolution [27, 28]. Only with high spatiotemporal resolution can the wake vortex evolution over time be measured.

2.4 LiDAR

The essential components of a LiDAR are a transmitter, and a receiver [29, p. 3]. The transmitter emits laser beams to capture the movement of air particles, so-called aerosols. The backscattered signal is measured by the receiver and compared with the emitted signal. The information retrieval works as follows [30]: The emitted laser pulse replicates a normal distribution, thence not measuring a distinct point but a measurement volume. The measurement's range gate must be retrieved from the time of flight. Given that a time series of intensities are measured, the Fourier transformation is used to obtain a frequency spectrum. The Doppler shift f_{D} between the emitted and received signal is used to calculate the radial velocity according to equation (2.10). Given the Doppler shift f_{D}, the radial velocity is

$$V_r = \frac{\lambda f_{\mathrm{D}}}{2}, \tag{2.10}$$

where λ is the optical wavelength of the emitted laser beam [31].

A LiDAR emits several laser pulses along a so-called Line of Sight (LOS). Spherical coordinates can describe a LOS. The azimuth angle θ describes the horizontal rotation, the elevation angle φ the vertical rotation, and the range R the distance from the LiDAR. In Figure 5, a sketch of a LiDAR measurement can be found. The azimuth angle in this Sketch is constant, and only a change in the elevation angle is illustrated. This measurement gives a Range Height Indicator (RHI) scan type. This reflects the measurement scenario from the Vienna campaign [9], from which the data for this thesis originates.

Along each LOS, multiple measurements are done at different ranges and elevation angles. The result is a radial velocity profile of the aerosols at a cross-section perpendicular to the landing aircraft.

The quality of LiDAR measurements is rated by a Carrier-to-Noise Ratio (CNR) [32]. The CNR measures the energy the backscattered signal carries about filtered atmospheric noise. A high CNR can be interpreted as a high amount of information in the measurement, while unrealistically high CNR values can result from hitting hard targets. In contrast, low CNR values can be interpreted as noisy measurements [32]. Given that the

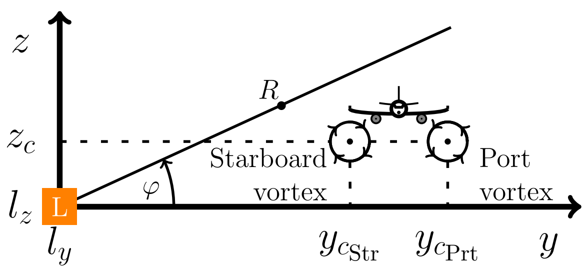


Figure 5: Sketch of an aircraft flying through a LiDAR measurement plane with counter-rotating wake vortices (inspired by [10]).

LiDARs used are calibrated to have a focal point of 500 m, the measurements outside that focal point are prone to noise leading to inaccuracies [10].

Inaccuracies in LiDAR measurements not only originate from the focal point, atmospheric turbulence, or not measuring a distinct point. The evaluation in cartesian coordinates also leads to measurement inaccuracies due to a conversion from polar coordinates and the loss of constant step sizes in coordinate directions. Hence the resolution further away from the LiDAR is lower, giving the need for interpolation.

Although this inaccuracies can lead to errors in evaluating the LiDAR scans, machine learning approaches are said to be able to adopt and generalize well. Furthermore, by first applying an object detection to treat wake vortices individually afterward is assumed to help focus on the important part and ignore most of the inaccuracies.

3 Literature Review

3.1 Traditional Wake Vortex Detection

On the one hand, there are predictive models such as the Probabilistic Two-Phase Wake Vortex Decay Model (P2P) developed at the German Aerospace Center (DLR) [23]. The P2P performs real-time wake vortex location and strength prediction based on theoretical models, aircraft configuration knowledge, meteorological data, and ground proximity information. The drawback of predictive models is the limited amount of real-world data, leading to inaccuracies [23, 33]. To account for such inaccuracies, fusion models were proposed that combine model predictions and sensor measurements [33]. Although sensor measurements like LiDAR measurements can account for physical turbulence detection, this comes at high computational costs and leads to errors like over- or underestimation, as well as the failure of recognizing mature vortices [33]. The combination of LiDAR measurements and a probabilistic model mitigates both systems' stand-alone usage drawbacks. An example is the Wake Vortex Prediction and Monitoring System (WSVBS) [17]. The WSVBS suggests utilizing LiDAR measurements to monitor and validate the wake vortex prediction of P2P [17]. A fast-time vortex detection and tracking algorithm is needed to integrate a conflict detection module that may issue warnings or adapt the WSVBS predictions [17]. Furthermore, the predictive models can not be used to evaluate large measurement campaigns such as the Vienna campaign [9].

On the other hand, characterization models characterize wake vortices in measurements. One characterization method is the Velocity Envelope (VE) method [34]. The reported characterization errors are a standard deviation of 9 m for the vertical, 13 m for the horizontal coordinates with a median absolute distance error of 7.91 m, and an absolute error of 13 m^2/s for the circulation [35, 10]. Another characterization method is the Radial Velocity (RV) method [36], explained in further detail in chapter 5.1. The advantage of the RV method over the VE method is the ability to be used for different kinds of LiDARs, i.e., the RV method can also operate on LiDARs with a lower CNR than needed for the VE method [28]. For higher accuracy measurements, LiDARs with a shorter wavelength are preferred. The accuracy of the RV method is given as root mean squared error for the elevation angle of $0.21°$, range gate of 1.8 m and 10.3 m^2/s for the circulation [36]. The accuracy of the RV method in terms of vortex center estimation is comparable to the VE method, but the circulation estimation of the VE method is superior [36]. Due to the semi-automatic nature of the RV method, the computation time for a single LiDAR scan is approximately 6 s [32]. Thus, the RV method is insufficient for the fast-time detection and tracking of wake vortices needed for WSVBS. Although the characterization models could be used to evaluate single LiDAR scans, they are not fast enough to evaluate large amounts of data from measurement campaigns. This gave rise to the research on new approaches to wake vortex detection such as artificial intelligence (AI).

3.2 Artificial Intelligence for Wake Vortex Detection

The topic of AI nowadays finds its way into any application. AI can also be found in applications of fluid dynamics [37]. Furthermore, the usage of AI in aviation safety is of interest, such that the European Union Aviation Safety Agency (EASA) just recently has published a concept paper "EASA Concept Paper: First usable guidance for Level 1 machine learning applications" [38]. This paper lays out a guideline of objectives one should address while developing and deploying AI into safety-related or environment-related applications in all domains covered by the EASA Basic Regulation (Regulation (EU) 2018/1139) [38]. Objectives taken into account can be grouped into trustworthiness analysis, learning assurance, explainability, and safety risk mitigation.

In the field of wake vortex characterization, there are three different tasks where AI finds application. The identification of wake vortices being present, the detection of single wake vortices, and the characterization of wake vortex parameters. To identify a LiDAR scan containing wake vortices, support vector machines (SVM) [39], random forest (RF) [40], and VVGNet [41] were used. A machine learning model, including other measurement data, was proposed using a CNN-LSTM approach [42]. An attention-based model is the most recent approach to wake vortex identification [43].

The second task is to detect wake vortices in a LiDAR scan. For that task, faster R-CNN [44], and YOLOv3 [45] were used. The faster R-CNN model was trained to detect vortices but not to classify the vortex type [44]. With a YOLOv3 approach, only the tail vortices (port) were trained [45] and achieved an accuracy of 94%.

All those approaches did not classify specific vortex parameters like vortex center or circulation strength. The first approach to classify those vortex parameters used a feedforward neural network (FNN) and a convolutional neural network (CNN) [10]. Experiments showed that a CNN model is superior to a FNN model [10]. Although this model could classify wake vortex parameters for different vortices in a LiDAR scan, it was trained on detecting a vortex pair leading to issues with scans containing more or fewer vortices. This can happen when a vortex of a previous overflight stalls over the runway due to crosswind or when aircraft land close to each other such that the vortex pair of the preceding aircraft has not yet vanished [1, 10]. Thus, it is desirable to detect and characterize the wake vortices individually. The goal is to use YOLO as an object detection ANN to detect individual vortices and employ a CNN for wake vortex parameter characterization on those vortices individually.

3.3 Reasearch Questions

Based on the literature review, the following research questions arise:

1. Can a bounding box detection network, in particular YOLO, accurately detect wake vortices in a LiDAR scan independent of the number of vortices being present in the respective LiDAR scan?
2. Can we use YOLO to detect vortices and classify port and starboard vortices accordingly?
3. Is the bounding box prediction of YOLO already sufficiently accurate to identify the vortex center?
4. Can an additional CNN improve the vortex center prediction based on the bounding box prediction?
5. Can we improve the vortex circulation prediction with a CNN used on the bounding box output compared to a CNN used on the complete scan?
6. Is a pipeline of using first YOLO and afterward a CNN still fast enough for fast-time prediction?

4 Artificial Neural Networks Theory

Artificial neural networks (ANN) were invented by McCulloch and Pitts [46] in 1943. Since then, the development of ANNs has become increasingly involved and transitioned from basic *feedforward neural networks* feedforewar(FNNs) to *convolutional neural network* CNNs proposed by Fukushima [47] and further developed in [48]. The basic idea of convolutional neural networks nowadays is still prominent in most neural networks.

In this chapter, we first introduce the basic idea of FNNs to then extend that idea toward CNNs. Afterward, the training mechanism is explained, and fundamental ideas of object detection networks are introduced.

4.1 Feedforward Neural Network

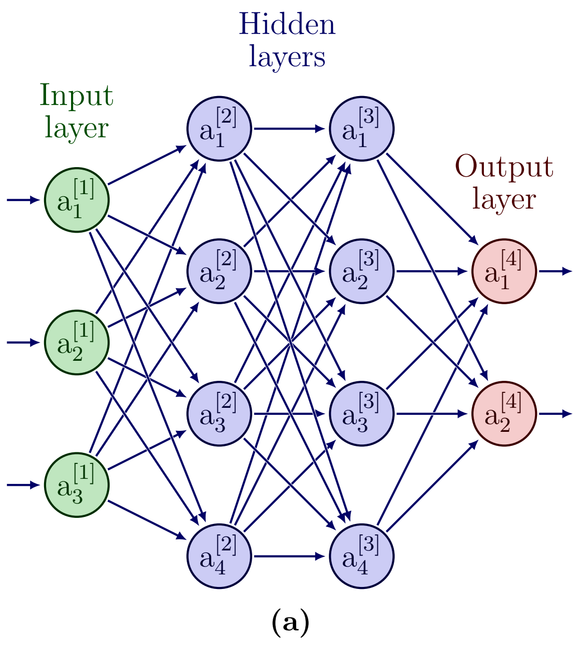
The underlying idea of a FNN is to approximate a function f^* that fulfills a particular task, for example, *classification* or *regression* [49, p. 164]. To find such a function, the FNN has to learn parameters p that define f^*. A FNN is built by collecting multiple *neurons* to a *layer* and stacking multiple layers on top of each other to form a network. The general definition of an FNN is:

Definition 4.1 (Feedforward Neural Network [50]). Suppose $L \in \mathbb{N}$ is the number of *layers* and n_l is the number of *neurons* of the l-th layer, with $l \in \{1, 2, \dots, L\}$. Moreover, let the network be a mapping from $\mathbb{R}^{n_1}$ to $\mathbb{R}^{n_L}$. Furthermore, let the *weight matrix* be $W^{[l]} \in \mathbb{R}^{n_{l+1} \times n_l}$ with $w^{[l]}_{j,k}$ being the weight applied to the output of the j-th neuron of layer l by the k-th neuron of layer $l+1$. The *bias* of the l-th layer is defined similarly by $b^{[l]} \in \mathbb{R}^{n_l}$. The *activation function* of the l-th layer is called $\alpha^{[l]}$ and is applied componentwise to the output of each neuron of the l-th layer. The output of the neural network with input $\mathbf{x}$ is

$$\begin{aligned} \mathrm{a}^{[1]} &:= \mathbf{x} \in \mathbb{R}^{n_1} \\ z^{[l]} &:= W^{[l]}\mathrm{a}^{[l]} + b^{[l]} \in \mathbb{R}^{n_{l+1}}, \quad l \in \{1, 2, \dots, L-1\} \\ \mathrm{a}^{[l+1]} &:= \alpha^{[l]}(z^{[l]}). \end{aligned} \tag{4.1}$$

We call the first layer *input layer*, the last layer *output layer*, and the layers in between *hidden layers*. More details on the activation function in the next chapter.

The term feedforward originates from the behavior of the data fed forwardly through the network. In Figure 6a, one can see a simple example of such a network. In this example, the FNN consists of four layers, with the input layer consisting of three neurons, the hidden layers of four neurons, and the output layer of two neurons. This network represents a function mapping an input vector $\mathbf{x} \in \mathbb{R}^3$ to an output vector $\mathrm{a}^{[4]} \in \mathbb{R}^2$. An illustration of the connection between the input layer and the second layer can be found in Figure 6b with the corresponding weights written on the connections to the first neuron of the second layer. Since the outputs of one layer are connected to every neuron of the next layer, those layers are called *fully connected layer* or *dense layer*.



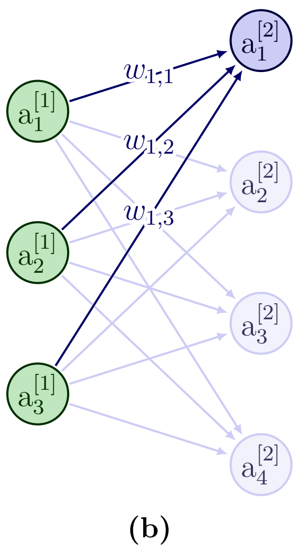


Figure 6: (a) Example of a FNN with two hidden layers mapping an input vector of length three to an output vector of length two. (b) Illustration of how one neuron has information fed forward from the previous layer. [51]

4.2 Activation Functions

The task of the activation function is to mimic some decision boundary at which a neuron is activated [52]. The aforementioned bias shifts that decision boundary, thus the name. A principal idea is whether a neuron fires or not. The use of the Heaviside function (4.2) can accomplish this idea.

$$\mathrm{H}(x) := \begin{cases} 0 & , x < 0 \\ 1 & , x \geq 0 \end{cases} \tag{4.2}$$

Over time multiple different activation functions were intoduced [52]. The activation functions employed in this thesis are the *logistic function* σ, the *rectified linear unit (ReLU)*[53], *leaky ReLU*[54] (LReLU) and *Mish*[55]. The logistic function, LReLU and Mish are the activation functions used in YOLOv4. The activation function used in the base model of our CNN regression network is ReLU.

The logistic function can be seen as a continuous version of the Heaviside function (4.2). It is *sigmoidal* as the logistic function approaches zero for the negative limit $\lim_{x\to-\infty} \sigma(x) = 0$ and approaches one for the positive limit $\lim_{x\to\infty} \sigma(x) = 1$ [56]. This leads to the *universal approximation theorem* by Cybenko [56], stating that a network with a single hidden layer can uniformly approximate any continuous function of n variables with support in the unit hypercube ($[0, 1]^n$) when using a sigmoidal activation function. A similar result was proven by Leshno et al. [57] for a wider variety of activation functions, particularly for ReLU. In Table 1, the activation functions used are displayed with their respective first derivative.

Table 1: Activation functions and their derivative. In the derivative of Mish, $\omega = 4(x+1) + 4e^{2x} + e^{3x} + e^{x}(4x+6)$ and $\delta = 2e^{x} + e^{2x} + 2$.

	Activation function	**First derivative**
Logistic	$\sigma(x) = (1+e^{-x})^{-1}$	$\sigma'(x) = \sigma(x)(1-\sigma(x))$
Mish [55]	$\mathrm{Mish}(x) = x \cdot \tanh\left(\ln\left(1+e^{x}\right)\right)$	$\mathrm{Mish}'(x) = \frac{e^{x}\omega}{\delta^{2}}$
ReLU [53]	$\mathrm{ReLU}(x) = x \cdot \mathrm{H}(x) = \max(0,x)$	$\mathrm{ReLU}'(x) = \mathrm{H}(x),\ x \neq 0$
LReLU [54]	$\mathrm{LReLU}_{\alpha}(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$	$\mathrm{LReLU}'(x) = \begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases}$

We can introduce a so-called *shape parameter* to the logistic function ending up with $\sigma_{\beta}(x) = \sigma(\beta x)$. The shape parameter changes the slope of the logistic function at zero to $\beta/4$. In Figure 7, one can see the different activation functions plotted on an interval of $[-1,1]$.

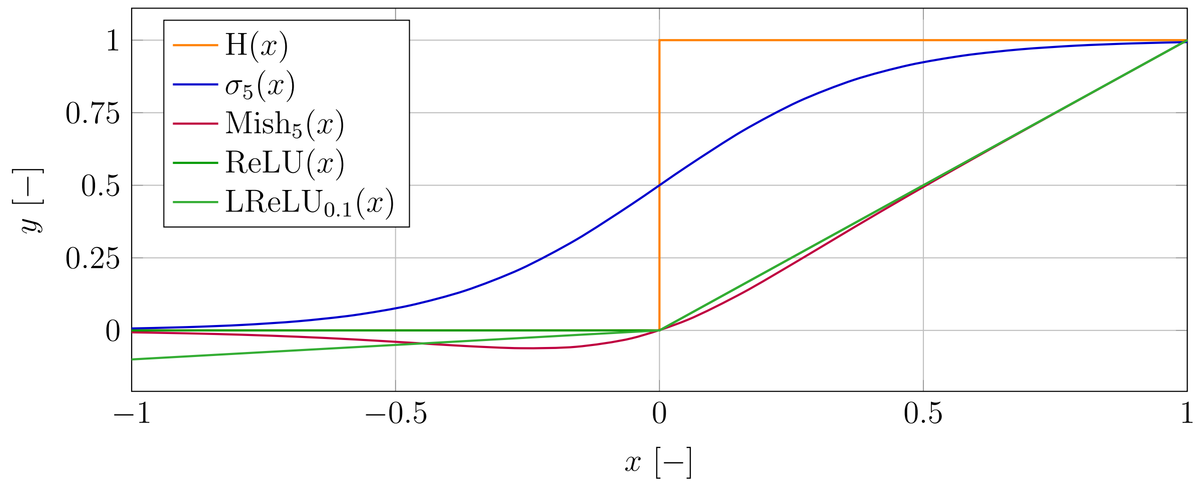


Figure 7: Heaviside-function (orange), Logistic function with shape parameter $\beta = 5$ (blue), Mish (red), ReLU (green), and LReLU (light green). The shape parameter is chosen to be 5 only for visualization purposes.

The choice of activation function depends on different features. The logistic function is an analytical function. Thus we do not have problems with differentiation. Furthermore, it is bounded from above and below. The image of the logistic function lies in the interval of $(0,1)$. Thus, it is often used to display some probability or binary classification task. The ReLU activation function is most prominently for computer vision task [58]. It is differentiable almost everywhere and bounded from below and unbounded above. Due to the clipping of negative values, an optimization only feeds back the values of an active neuron. However, this is only occasionally desired, thus LReLU overcomes that problem but is unbounded above and below. The Mish activation function is similar to ReLU bounded from below and unbounded above but preserves a small amount of negative information [55]. Furthermore, it is continuously differentiable everywhere, often a desired feature in gradient-based optimization [55].

4.3 Optimization

Typical tasks of an ANN are regression or classification. This thesis focuses only on *supervised learning* since the data sets introduced in Section 5.1 are labeled, i.e., the data has corresponding targets, also called annotations. Furthermore, we aim at mapping input LiDAR scans to specific targets like bounding boxes, vortex center and vortex strength which can be achieved with supervised learning [59, p. 94]. In contrast, *unsupervised learning* exists, where a data set without annotations is used. Unsupervised learning is most common in tasks like data denoising, compression, or visualization, where the basic idea is to find properties of the underlying data set [59, p. 94].

We assume that a *labeled data set* with D data points is given. For each data point $\mathbf{x}^{\{i\}}$ there exists a label $\mathbf{y}^{\{i\}}$ with $i \in \{1, 2, \dots, D\}$. Such a labeled data set makes it possible to formulate a *loss function*. The widely used loss function for regression tasks is the *mean squared error* (MSE) loss function [59, p. 91]

$$\mathrm{L}(\mathbf{p}) = \frac{1}{D} \sum_{i=1}^{D} \| f^*(\mathbf{x}^{\{i\}}, \mathbf{p}) - \mathbf{y}^{\{i\}} \|_2^2. \tag{4.3}$$

Let vector $\mathbf{p} \in \mathbb{R}^s$ contain all the weights and biases of the network. Then the loss function is a mapping $L : \mathbb{R}^s \to \mathbb{R}$. We aim to find the optimal weights and biases to minimize the MSE. This process is also called learning. To do so the basic idea is to use the *stochastic gradient descent* (SGD) [60], which is based upon the *gradient descent* method described by

$$\mathbf{p} \leftarrow \mathbf{p} - \eta \nabla \mathrm{L}(\mathbf{p}). \tag{4.4}$$

In the machine learning context, the *step size* η is also known as *learning rate*. Since the MSE consists of a sum running over our data set, the gradient of the MSE can also be split into the sum of gradients. One summand of the MSE is

$$\mathrm{L}_i \coloneqq \| f^*(\mathbf{x}^{\{i\}}, \mathbf{p}) - \mathbf{y}^{\{i\}} \|_2^2. \tag{4.5}$$

Thus the gradient of (4.3) can be calculated with the help of (4.5)

$$\nabla \mathrm{L}(\mathbf{p}) = \frac{1}{D} \sum_{i=1}^{D} \nabla \mathrm{L}_i(\mathbf{p}). \tag{4.6}$$

The idea of the SGD (4.7) is now not to consider all samples of a data set like in (4.6) - as an update - but instead only consider a single data point chosen uniformly at random, such that the update becomes

$$\mathbf{p} \leftarrow \mathbf{p} - \eta \nabla \mathrm{L}_i(\mathbf{p}). \tag{4.7}$$

Alternate variants based on this idea exist. For instance, we can employ so-called *mini-batches* [61], where a random set of data points $\mathcal{I} \subset \{1, 2, \dots, D\}$ is chosen to calculate the gradient ending up with the update

$$\mathbf{p} \leftarrow \mathbf{p} - \eta \sum_{i \in \mathcal{I}} \nabla \mathrm{L}_i(\mathbf{p}). \tag{4.8}$$

A method based on (4.8) is sometimes referred to as mini-batch gradient descent.

Another choice of loss function used by YOLOv4 is the *binary-crossentropy* loss (4.9)[62, p. 206], which is often implemented in binary classification tasks in combination with a logistic activation function in the last layer [59, p. 60]. As binary-crossentropy measures the distance between two probability distributions it is favorable in case of predicting class probabilities [59, p. 73]. The binary-crossentropy is defined as

$$\mathrm{H}(\mathbf{p}) = -\frac{1}{D} \sum_{i=1}^{D} y^{\{i\}} \log(f^*(\mathbf{x}^{\{i\}}, \mathbf{p})) + (1 - y^{\{i\}}) \log(1 - f^*(\mathbf{x}^{\{i\}}, \mathbf{p})). \tag{4.9}$$

The optimization algorithm used in this thesis is ADAM, it has shown promising performance in preceding works [10]. The idea behind ADAM is to employ decay parameters $\beta_1 < \beta_2 < 1$, which adjust the learning rate. For more details, see "ADAM: A Method for Stochastic Optimization" by Kingma and Ba [63].

To update the weights and biases individually we need the derivative of the loss function. To calculate the derivative, FNN suggests an efficient way via *backpropagation* developed by Rumelhart et al. in "Learning Internal Representations by Error Propagation"[64].

4.4 Convolutional Neural Network

To get an idea of spatial dependence, *Convolutional Neural Networks* (CNNs) were introduced. Typical CNNs are comprised of five types of layers, explained in this section. The layers are *convolutional layers*, *batch normalization layers*, *pooling layers*, *flatten layers*, and *fully-connected layers*.

Convolutional Layer

The main building block of a CNN is the convolutional layer. It is based on the linear operation convolution defined in (4.10).

Definition 4.2 (Discrete 2D-Convolution)**.** Let $\mathbf{x}, \mathbf{w} \in l^1(\mathbb{Z}^2)$ and $* \colon l^1(\mathbb{Z}^2) \times l^1(\mathbb{Z}^2) \to l^1(\mathbb{Z}^2)$, where $l^1(\mathbb{Z}^2) := \{f \colon \mathbb{Z}^2 \to \mathbb{R} : \sum_{k \in \mathbb{Z}^2} |f(k)| < \infty\}$. The discrete convolution of $\mathbf{x}$ with the *convolutional kernel* $\mathbf{w}$ is then defined by

$$\mathbf{f}(k) = (\mathbf{x} * \mathbf{w})\ (k) := \sum_{j \in \mathbb{Z}^2} \mathbf{w}(k - j)\mathbf{x}(j), k \in \mathbb{Z}^2. \tag{4.10}$$

In the context of machine learning, $\mathbf{x}$ is called input, $\mathbf{w}$ is called *kernel* or sometimes also referred to as *filter*, and the output of the convolution is called *feature map* [49]. In our case, the convolutional kernel $\mathbf{w} \in \mathbb{R}^{k \times k}$ is a finite real square matrix, and $\mathbf{x} \in \mathbb{R}^{n \times m}$ corresponds to the LiDAR scans, which are also represented by a finite real-valued matrix thus (4.10) becomes

$$\mathbf{f}_{i,j} = \sum_{q=0}^{k-1} \sum_{r=0}^{k-1} \mathbf{w}_{q,r} \mathbf{x}_{i-q,j-r} \text{ , with } (i,j) \in \{1,2,\dots m\} \times \{1,2,\dots n\}. \tag{4.11}$$

From (4.11), we can get the notion of a weighted sum of neighboring pixel values. Thus a spatial representation is given. There is also *strided* convolution with stride $s \in \mathbb{N}$ such that (4.11) becomes

$$\mathbf{f}_{i,j} = \sum_{q=0}^{k-1} \sum_{r=0}^{k-1} \mathbf{w}_{q,r} \mathbf{x}_{si-q,sj-r} \text{ , with } (i,j) \in \{1,2,\dots m\} \times \{1,2,\dots n\}. \tag{4.12}$$

Strided convolution is sometimes used instead of pooling for dimensionality reduction.

In both convolution cases, the finite dimension of $\mathbf{x}$ leads to the problem of missing values outside the domain. Close to the edge, it occurs such that values outside of the matrix domain are needed. To provide those values, in machine learning, one uses *zero-padding*, extending the matrix dimension by $p \in \mathbb{N}$ and filling in the new values with zero before applying the convolution [65]. The output shape of a convolutional layer depends on the input size, the kernel size, the zero-padding, and stride. The most used padding cases are called *valid* padding, which corresponds to no padding, and *same* padding. The latter corresponds to zero-padding such that the input shape does not change. While the former causes a shrinkage of the original matrix depending on the kernel size ending up with $\mathbf{f} \in \mathbb{R}^{n-s+1 \times m-s+1}$ [65].

The number of trainable parameters of a convolutional layer P_{CL} depends on the number of input channels c_{input}, the number of output channels c_{output}, i.e., the number of filters in the input layer and output layer as well as the kernel size $k_h \times k_w$. The formula for calculating the number of trainable parameters of a convolutional layer is then defined by

$$P_{\text{CL}} = \underbrace{c_{\text{output}} \cdot (c_{\text{input}} k_h k_w)}_{\text{number of weights}} + \underbrace{c_{\text{output}}}_{\text{number of biases}} . \tag{4.13}$$

Batch Normalization

A regularisation method sometimes used in CNNs is *batch normalization* introduced in [66]. Since the input of each layer follows a different distribution, training can be challenging as layers have to always adapt to that distribution [66]. The idea behind batch normalization is to scale each batch with a mean of 0 and a standard deviation of 1. To accomplish that, each batch's mean and standard deviation is calculated. Let

$\mathcal{B} = \{x_1, \ldots, x_m\}$ be a batch. The mean and standard deviation are given by

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{and} \quad \sigma_{\mathcal{B}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2}. \tag{4.14}$$

Batch normalization is now performed by normalizing each data point from the batch accordingly to (4.15). With the help of (4.14) the normalization of x_i is given by

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \varepsilon}, \quad 0 < \varepsilon. \tag{4.15}$$

We avoid dividing by zero by adding a small value ε.

Pooling Layer

Another option for dimensionality reduction is using a *pooling layer*. A pooling layer combines values in small regions to summarize. Mostly pooling layers are utilized after the activation of a convolutional layer and help to make the representation approximately invariant to small translations of the input [49]. There are several pooling methods, but the one used in this thesis is *max pooling* [67], which filters out the maximal value in a fixed-size neighbourhood. Another pooling option is *average pooling* [67], which averages over a local region. Max pooling can detect subtle local features which average pooling would miss [67]. In most CNNs, max pooling with a filter of size 2×2 and a stride of 2 is used, hence also applied in this thesis [68].

Flatten Layer

The *flattten layer* is used right before the fully connected layer and after the convolutional layers. It flattens the multidimensional output array to a one-dimensional vector, which the fully connected layers can use. This is orchestrated by concatenating each row of a specific feature map one after another, ending up with a large one-dimensional vector and thus having no trainable parameters.

Fully Connected Layer

The desired output in most applications of ANNs is a vector encoding some information. Hence as the output layer of a CNN, in most cases fully connected layers are chosen. An example of a *fully connected layer* can was previously depicted in Figure 6b. Each neuron of a fully connected layer is connected with each neuron of the previous layer. The input to a fully connected layer is a one-dimensional vector. Let $\mathbf{a} \in \mathbb{R}^{n_{\text{input}}}$ be the input to a fully connected layer and $\mathbf{y} \in \mathbb{R}^{n_{\text{output}}}$ the output of the fully connected layer. The number of trainable parameters P_{FCL} of such a layer is

$$P_{\text{FCL}} = \underbrace{n_{\text{output}} \cdot n_{\text{input}}}_{\text{number of weights}} + \underbrace{n_{\text{output}}}_{\text{number of biases}}. \tag{4.16}$$

4.5 Object Detection

The task of *object detection* is not only to classify objects in an image but also to localize those objects. ANNs performing object detection aim to mark existing objects in any image with a rectangular *bounding box* [69]. We define a bounding box as follows.

Definition 4.3 (Bounding Box)**.** Let $(x_c, y_c, w, h) \in \mathbb{R}^4$, with (x_c, y_c) being the center point and (w, h) being the dimension defining the rectangular bounding box $\mathbf{B}$ as

$$\mathbf{B} := \left\{ (x, y) \in \mathbb{R}^2 : x_c - \frac{w}{2} \leq x \leq x_c + \frac{w}{2}, y_c - \frac{h}{2} \leq y \leq y_c + \frac{y}{2} \right\}. \tag{4.17}$$

There are two main machine learning approaches to object detection. The first approach starts with a network that proposes regions for interest, the *region-proposal network* (RPN). A second network - of detection type - then classifies objects in those regions of interest [70]. Networks following this approach are called *two-stage* detectors. Prominent state-of-the-art models following this approach are R-CNN [71], fast R-CNN [72], faster R-CNN [73], mask R-CNN [74] and FPN [75]. The second approach is the *one-stage* detectors. One-stage detectors accomplish regression and classification in a single shot. The most prominent one-stage detectors are YOLO [76], YOLO9000 [77], YOLOv3 [78], YOLOv4 [79], RetinaNet [80] and SSD [81].

Because the one-stage detectors do not need an RPN, they are faster than the two-stage detectors enabling fast-time detection. In case of WVMSs and wake vortex tracking fast-time detection is crucial. For that reason, the chosen object detection network will be YOLOv4[1]. It outperformed other state-of-the-art object detection networks in terms of accuracy and speed[79].

Despite the difference of having two stages or only one to do object detection, modern object detection networks consist of three main parts, the *backbone*, the *neck*, and the *head* illustrated in Figure 8 [79]. The backbone network, in most cases, is pretrained for the ImageNet [82] classification task [83]. The task of the backbone is to extract higher-level features of an image. The role of the neck is to collect feature maps from different stages of the backbone and is generally composed of several *top-down-paths* and *bottom-up-paths*. The idea is to let lower-level features and higher-level features interact [84]. The head predicts the bounding boxes and classification.

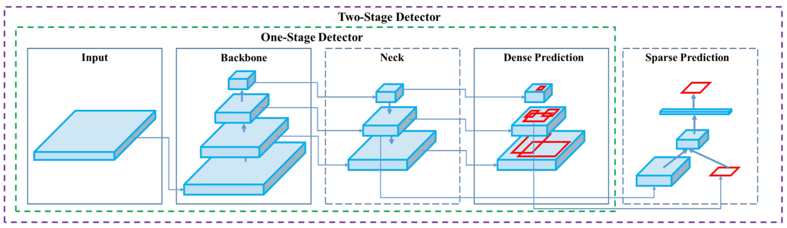


Figure 8: Sketch of general object detectors (taken from [79]).

[1]During this thesis, YOLOv7 was published, which is superior to YOLOv4 but could not be adapted to in time.

5 Data Sets, YOLO and Regression Network

The focus of this thesis is on the data acquired during the Vienna measurement campaign [9]. This data set was generated with LiDAR measurements at Vienna International Airport from May 2019 until November 2019 by DLR. The data contains the radial velocity RHI LiDAR scans. Another data set based on vortex models is used for testing purposes. This data set contains artificial *proxy* scans that aim to model LiDAR measurements. The scans in the LiDAR data set from Vienna include atmospheric effects, secondary vortices, noise, and measurement errors. The proxy data set, in comparison, contains immaculate data but does not reflect real-life scenarios as it lacks crosswind and other atmospheric effects. Therefore this data set was only used to evaluate whether YOLOv4 is suitable for detecting different numbers of vortices in scans.

Both data sets also contain corresponding labels with the vortex center locations in polar coordinates, $R_{\mathrm{t}}, \varphi_{\mathrm{t}} \in \mathbb{R}_{\geq 0}$ and vortex circulation $\Gamma_{\mathrm{t}} \in \mathbb{R}_{\geq 0}$. Hence labels for supervised training and evaluation are given. The data set used to train the regression net contains cut-out vortices only from the LiDAR data set. We set up a predicition pipeline to obtain the individual vortices from the YOLOv4 prediction and input those into the CNN. Throughout this thesis, we use the terms *prediction* and *estimation* equivalently. In essence, prediction does not represent a prediction in time.

5.1 LiDAR Data Set

Since the Vienna measurement campaign was conducted to evaluate the effectiveness of plate lines in wake vortex mitigation, the data set contains measurements with and without plate line usage. Hence in some scans, plate line effects are present. In the following, a summary of the measurement campaign is given based on "Mitigating Wake Turbulence Risk During Final Approach via Plate Lines"[9] by Holzäpfel et al. [9].

Figure 9 depicts the setup of the measurement instruments at the runway of the Vienna International Airport. During the campaign, two plate lines were installed consisting of eight and nine plates of dimension 4.5 m $\times$ 9 m displayed by red dashes. For the radial velocity measurement, at most three Leosphere Windcube 200S (1.543 μm) micro-PCDLs were used at once, positioned at three out of five possible positions (L1-L5). The three measurement plane combinations used were: (L1, L2, L3), (L2, L3, L4) and (L3, L4, L5) [25]. At positions A-C, additional meteorological instruments were placed. The runway is at the bottom of Figure 9, and the aircraft approaches from the top. The average flight altitudes above ground at LiDAR planes are 40.8 m at L1, 45.8 m at L2, 54.3 m at L3, 64.8 m at L4, and 74.5 m at L5 with a standard deviation of 4.9 m [25].

The LiDAR measured the radial velocities at discrete points along each LOS perpendicular to the runway. As the campaign analysis focused only on landings at weak crosswinds of 1.5 m/s [9], it can be assumed that the vortices shared the same plane as the LiDAR's LOSs. Each LOS has 151 measuring points, called *range gates*, starting at a range of $R_{\min} = 80$ m to a range of $R_{\max} = 530$ m with a step size of $\Delta R = 3$ m. The minimum and maximum elevation angles used depend on the LiDAR position adjusted to the average flight altitudes at the different LiDAR positions. The elevation angle step size

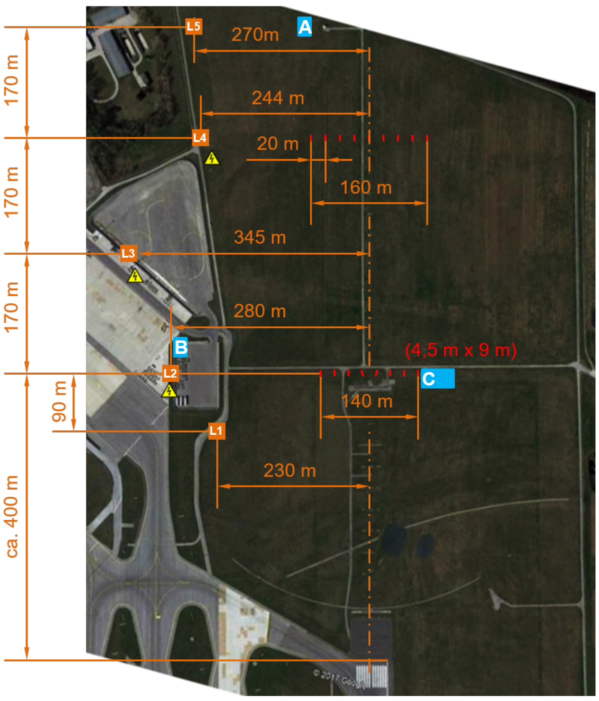


Figure 9: Vienna measurement campaign setup of instrumentation with L1-L5 being the LiDAR positions, A-C being additional meteorological instruments, and the red dashes being the plates (taken from [9]).

was $\Delta\varphi = 0.2°$ for all positions. The corresponding minimum and maximum elevation angles used can be found in Table 2. A list of three-tuples can therefore describe a raw LiDAR scan

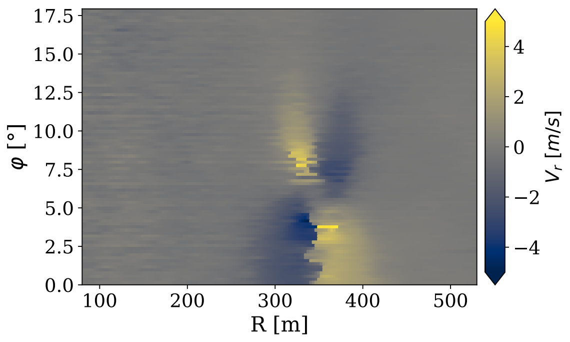
$$(R_i, \varphi_j, V_r(R_i, \varphi_j)) \in \mathbb{R}^3.$$

The polar coordinates (R_i, φ_j) represent at which the corresponding radial velocity $V_r(\varphi_i, R_j)$ was measured, with $i \in \{0, 1, \dots, 150\}$ and $j \in \{0, 1, \dots H_{\text{lid}}\}$.

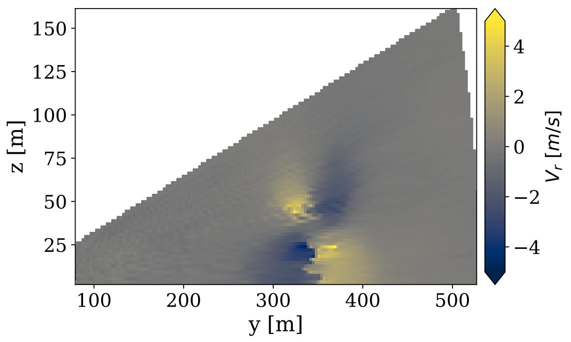
Table 2: The elevation angles covered correspond to each LiDAR position and the number of LOSs needed.

LiDAR position	φ range	H_{lid} number of LOS beams
L1	$0° - 25°$	125
L2	$0° - 20°$	100
L3	$0° - 18°$	90
L4	$1° - 28°$	135
L5	$0° - 29°$	145

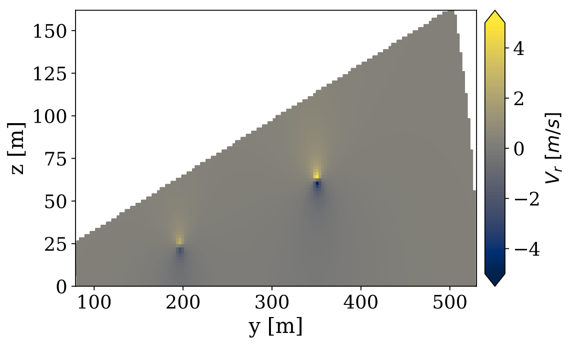
An example of a raw LiDAR scan can be seen in Figure 10a with the corresponding scan transformed to cartesian coordinates in Figure 10b. The grid in polar coordinates is equidistant. Hence transforming to cartesian coordinates, one loses this property. In Figure 10c and Figure 10d proxy scans with two and four vortices, respectively, are illustrated.



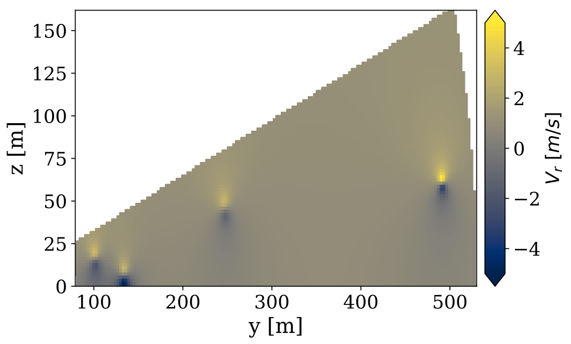
(a)



(b)



(c)



(d)

Figure 10: (a) Example of a raw LiDAR scan. (b) Example of a raw LiDAR scan transformed to cartesian coordinates. (c),(d) Example of a proxy scan containing two and four vortices in cartesian coordinate system.

YOLOv4 needs bounding box labels, thus we focus on scans in cartesian coordinates, explained in more detail in section 6.1.1. The input format required for YOLOv4 is that of images. Thus we have to transform the polar coordinates to an equidistant cartesian grid, which requires data interpolation. This may lead to inaccuracies, especially for high range gate values, since the distance between two grid points at the same range gate but with different elevation angles increases with the range gate.

Another source of inaccuracies is the time a single measurement takes. Changing the LOSs takes 50 ms. Hence a LiDAR scan cannot be instantaneous. This time difference results in radial velocities, measured with a maximal time difference of 7.25 s [32]. The LiDAR scans were initiated at either end of the elevation angle interval. Consequently, the resulting data set contains roughly an equal amount of data with a time distortion from top to bottom and vice versa. Thus this effect can be considered negligible [32].

5.2 Labels - Radial Velocity Method

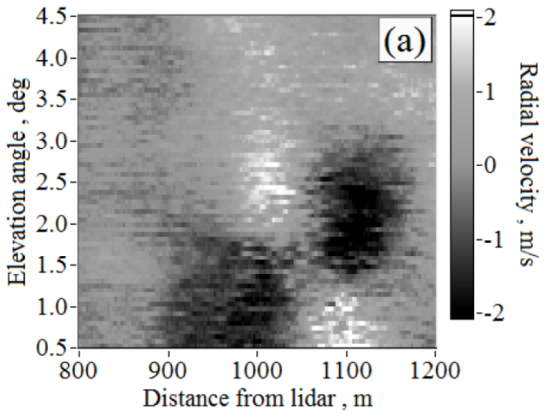
The labels for the data set originate from the assessment of the measurement campaign, which used micro-PCDLs hence the RV method [36] was used. The idea behind the RV method is first to estimate the center of the vortices and afterward estimate the circulations via minimizing a functional. The RV method is described as an example for two vortices. To estimate the range gate of the vortex center R_{C_i}, the maxima of

$$D(R_k) := \sum_{j=0}^{B_{\text{lid}}} V_r(R_k, \varphi_j)^2 \tag{5.1}$$

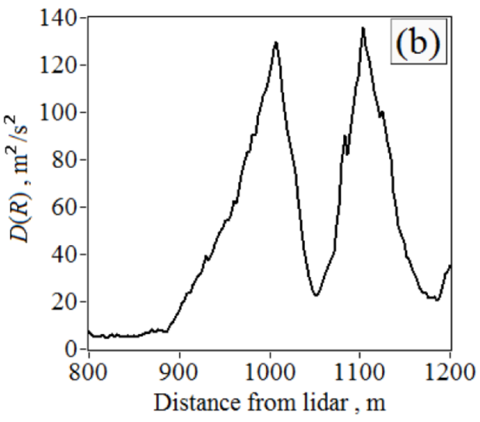
are sought [36]. One example plot of (5.1) can be seen in Figure 11b with the underlying LiDAR scan in Figure 11a. The corresponding angles can be found by calculating the mean of the angle $\varphi_{\min}(R_{C_i})$ of minimal radial velocity and $\varphi_{\max}(R_{Ci})$ at the estimated range gate center [36]. A functional

$$\rho(\Gamma_i) = \sum_{j=0}^{B_{lid}} (V_r(R_{C_i}, \varphi_j) - \widetilde{V}_r(R_{C_i}, \varphi_j; \Gamma_1, \Gamma_2))^2 \tag{5.2}$$

is minimized to fit the best circulation. This functional (5.2) describes the difference between the radial velocities at the axis of the center range gate of the true LiDAR scan $V_r(R_{C_i}, \varphi_j)$ and a modeled scan $\widetilde{V}_r(R_{C_i}, \varphi_j; \Gamma_1, \Gamma_2)$ calculated theoretically with arbitrary circulations Γ_1 and Γ_2 [85].



(a)



(b)

Figure 11: (a) An arbitrary LiDAR scan. (b) $D(R)$ for the LiDAR scan from (a) (taken from [36]).

During the measurement campaign, 9 473 approaches were measured with approximately 20 scans per overflight [9]. Since the targets for the data set were constructed by the RV method, which is time-consuming, only a fraction of the data can be used for training, ending up with a data set of 16 349 samples in the complete data set used in three different variants. The total numbers of the samples per data set can be found in Table 3 with the corresponding splitting of the data set into training and validation data sets.

Table 3: The number of data samples contained in each data set used for training and validation of YOLOv4

Plate Line status	Train	Validation	Total number
up	5 994	958	6 963
down	8 428	969	9 386
both	14 422	1 927	16 349

An internal DLR python package by Grigory Rotshteyn was used to create the proxy data. It uses the Lamb-Oseen vortex model [86] and places it inside a LiDAR scan field corresponding to LiDAR position L3 (see Table 2). The initial vortex separation corresponds to an A320 with $b_0 = 26.8$ m. We place the vortices randomly such that the vortex center is still contained in the proxy scan. In the case of four vortices, we use two counter-rotating vortex pairs. In the case of three vortices, we use a counter-rotating vortex pair and a remaining starboard vortex in the first quarter of the scan. In Figure 10c, a resulting proxy scan with two vortices is depicted, and in Figure 10d, one with four vortices.

5.3 YOLO

First, object detection is employed to detect wake vortices in the overall LiDAR scans, in order to tackle the problem of detecting different numbers of wake vortices in different LiDAR scans. The name YOLO is an abbreviation for "You Only Look Ones", which reflects the nature of a one-stage detector. We use the original YOLOv4[2] written in C/C++ [87].

5.3.1 Network architecture

Like all object detectors, YOLOv4 consists of three parts: the backbone, the neck, and the head (see Section 4.5). As backbone YOLOv4 uses *CSPDarknet-53* [88]. In the neck, spatial pyramid pooling (SPP) [89] and path-aggregation network (PAN) [90] are used. For the detection head, YOLOv3 [78] is used. The basic building blocks of YOLOv4 are illustrated in Figure 12.

CSPDarknet-53 is an update to Darknet-53 [78] incorporating so-called *cross-stage partial* (CSP) connections. A CSP block contains convolutional layers with batch normalization and Mish as an activation function, thus abbreviated with CBM. The first CBM layer in a CSP block uses a stride of two to downsample the input. After that, the output is copied and fed through another CBM layer, N residual blocks, illustrated in Figure 12a, and another CBM layer before being concatenated with the second copy

[2]https://github.com/AlexeyAB/darknet

which was only fed through a CBM layer. The concatenation is then again fed through a CBM layer. An illustration of that process can be found in Figure 12b.

The PAN block combines features extracted at lower layers with features at higher layers. It uses bottom-up paths to make low-layer information easier to propagate [90]. In Figure 13, PAN is marked by the red connections. The difference between the PAN YOLOv4 uses to the originally proposed PAN is the usage of the concatenation of feature maps instead of addition [79].

The SPP block added after the CSPDarknet-53 backbone is used to significantly increase the network's receptive field and separate out the most significant context features [79]. This is accomplished using three different max pooling layers, each with a different pooling size, namely 5×5, 9×9, and 13×13. The SPP block is illustrated in Figure 12c.

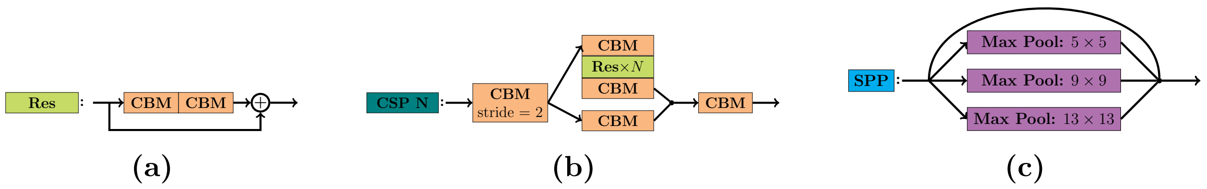


Figure 12: The basic building blocks of the YOLOv4 Network. (a) Residual Block with CBM blocks. (b) CSP block with N residual blocks. (c) SPP block.

Instead of the Mish activation function, YOLOv4 uses LReLU in the neck and the head of the network. After the convolution, batch normalization is also performed in the neck and head. Those blocks are abbreviated by CBL.

The detection head used is based on the anchor box idea from YOLOv3 [78, 79]. The functionality of the detection mechanism will be explained in more detail in the next section. The entire network of YOLOv4 can be seen in Figure 13.

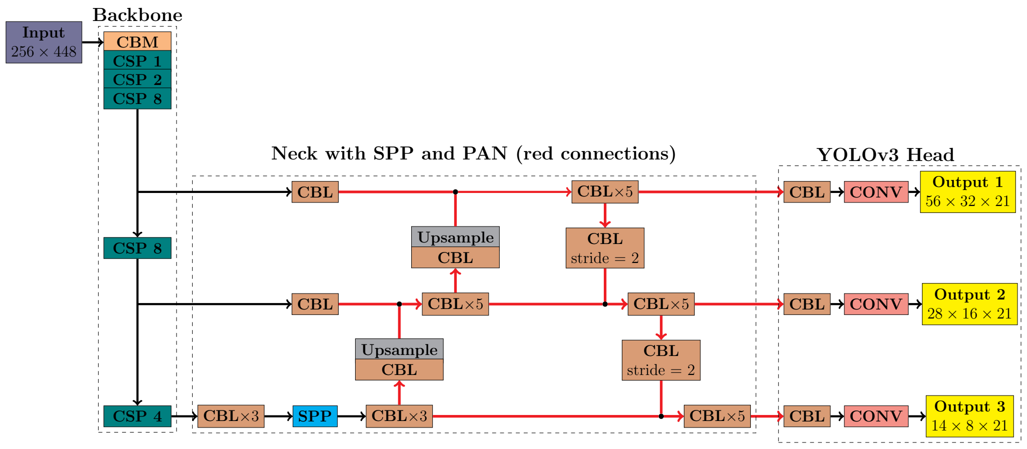


Figure 13: An illustration of the architecture of the YOLOv4 network. When the paths split up, the output is copied. When two paths join, the dot represents the concatenation of the inputs.

YOLOv3 Head

As detection head YOLOv4 uses the YOLOv3 detection head [79]. The prediction in the YOLOv3 detection head is made at three different scales, such that we have three outputs [78]. Each detection head's prediction is based on a grid of different sizes such that objects of different sizes can be detected. The grid dimensions can be found in Table 4 and a LiDAR scan with the corresponding grids in Figure 14.

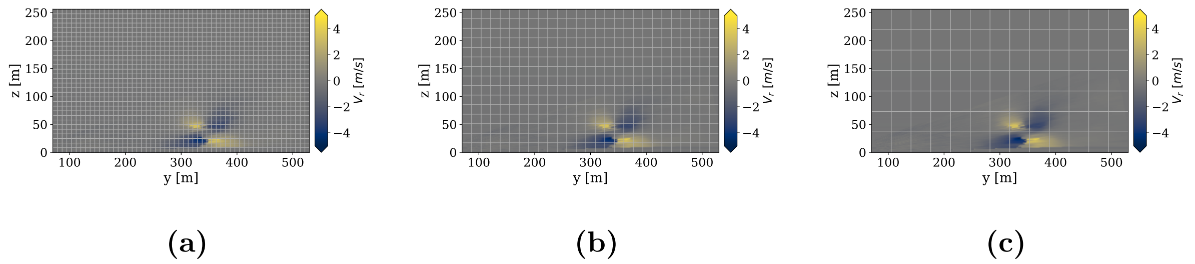


Figure 14: Three different grids used in the detection head of YOLOv3. (a) Fine grid: 56×32. (b) Medium grid: 28×16. (c) Coarse grid: 14×8.

To predict the position and dimension of a bounding box, YOLOv3 uses *anchor box priors* [78] with width $p_w \in \mathbb{N}$ and height $p_h \in \mathbb{N}$. At each scale, YOLOv3 uses three anchor box priors, such that a total of nine anchor box priors are used. Different object shapes can be represented by using different anchor box priors. For each anchor box and grid cell, an offset, as well as a class and *objectness*, is predicted. In our case, there are two classes: port and starboard. The objectness predicts the intersection over union (IoU) (5.3) of the ground-truth and the proposed box [77]. A metric to measure how much a predicted box and a ground-truth box match IoU is defined as follows.

Definition 5.1 (Intersection over Union [91])**.** Let $\mathbf{B}$ be a bounding box defined by (x, y, w, h) and $\mathbf{B}_{\text{gt}}$ be another bounding box defined by $(x_{\text{gt}}, y_{\text{gt}}, w_{\text{gt}}, h_{\text{gt}})$, corresponding to Definition 4.3. The intersection over union (IoU) of $\mathbf{B}$ and $\mathbf{B}_{\text{gt}}$ is then defined by

$$\text{IoU}(\mathbf{B}, \mathbf{B}_{\text{gt}}) := \frac{|\mathbf{B} \cap \mathbf{B}_{\text{gt}}|}{|\mathbf{B} \cup \mathbf{B}_{\text{gt}}|}. \tag{5.3}$$

The operation $|\mathbf{B}|$ corresponds to the cardinality of the finite set $\mathbf{B}$, which is understood as the number of elements a set contains. With IoU and the probability of an object existing in the predicted bounding box $\mathbf{B}$ objectness is defined by [76]

$$C(\mathbf{B}) := P(\text{object})\text{IoU}(\mathbf{B}, \mathbf{B}_{\text{gt}}). \tag{5.4}$$

The offset prediction of YOLOv3 is made by predicting coordinates t_x, t_y, t_w, t_h representing the center, width, and height offset, respectively. Furthermore, the objectness t_o and a probability for each class t_{c_i} is activated by the logistic function σ to represent a probability. The bounding box is then calculated with respect to the center of a grid cell, defined by the offset of c_x, c_y from the origin at the top left corner, by

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ C(B) &= \sigma(t_o) \\ P(\text{class}_i) &= \sigma(t_{\text{class}_i})\ i \in \{1, 2\}. \end{aligned} \tag{5.5}$$

At the testing time, the objectness is multiplied by a conditional class probability $P(\text{class}_i|\text{object})$ to give a class-specific confidence score. In that way, the probability of class i appearing in the bounding box and the quality of bounding box fitting is encoded [76]. Figure 15 is a sketch of the positioning and scaling of an anchor box to a predicted bounding box.

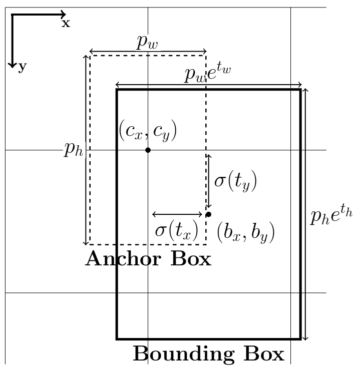


Figure 15: The process of transforming bounding box predictions from an anchor box prior to a predicted bounding box. (Inspired by [77])

With that knowledge, we can calculate the output dimension of each detection head. Let the detection head divide the image into a $S_1 \times S_2$ grid and let each grid cell predict B_{num} bounding boxes. Let furthermore be C_{num} be the number of classes to predict. The dimension of the detection head output can then be calculated by

$$S_1 \times S_2 \times [B_{\text{num}}(\underbrace{4}_{\text{bounding box coordinates}} + \underbrace{1}_{\text{confidence score}} + C_{\text{num}})]. \tag{5.6}$$

In our case, the detection heads are after the 139th layer, the 150th layer, and the 161st layer. Each detection head uses three anchor boxes, i.e., $B_{\text{num}} = 3$. Recalling that we aim to distinguish between the port and starboard vortices, two classes are used, i.e., $C_{\text{num}} = 2$. In Table 4, the output dimensions for each detection head are calculated according to equation (5.6).

Since each grid cell predicts three bounding boxes, some large objects or objects near the border of multiple cells can be well localized by multiple cells [76]. The tool

Table 4: The dimension of the output tensors from the three different detection heads and the corresponding anchor boxes.

Detection Head Layer	Output Tensor Dimensions	Anchor Box Dimensions
139	$56 \times 32 \times 21$	12×16, 19×36, 40×28
150	$28 \times 16 \times 21$	36×75, 76×55, 72×146
161	$14 \times 8 \times 21$	142×110, 192×243, 459×401

to mitigate that problem is non-maximum suppression (NMS). YOLOv4 uses greedy NMS [79]. The idea behind greedy NMS is to select a bounding box with the highest objectness and suppress all other bounding boxes that have an IoU above a given threshold $T_{\text{nms}} \in [0,1]$[92]. To assign a predicted bounding box to the correct ground-truth box, an IoU threshold of 0.231 is used [79].

5.3.2 YOLOv4 Loss Function

The loss function for all versions of YOLO can be split into three parts. The first part evaluates the bounding box prediction, the second part evaluates the objectness prediction, and the last part evaluates the class predictions. The loss function used by YOLOv4 is an updated version of the loss function from YOLOv3. For the bounding box regression, instead of the MSE loss, complete IoU (CIoU) [91] loss is used [79]. The CIoU loss not only considers the MSE between the bounding box predictions but considers the overlapping area, the distance between center points, and the aspect ratio [91].

Definition 5.2 (CIoU Loss [91])**.** Let $\mathbf{B}_{\text{p}} = B(\mathbf{b}, w_{\text{p}}, h_{\text{p}})$ be a predicted bounding box and $\mathbf{B}_{\text{gt}} = B(\mathbf{b}_{\text{gt}}, w_{\text{gt}}, h_{\text{gt}})$ the ground-truth bounding box. Furthermore, let the center of the bounding boxes be $\mathbf{b}$ and $\mathbf{b}_{\text{gt}}$, respectively. The consistency of the aspect ratio is measured by

$$v := \frac{4}{\pi^2}\left(\arctan\frac{w_{\text{gt}}}{h_{\text{hgt}}} - \arctan\frac{w_{\text{p}}}{h_{\text{hp}}}\right)^2. \tag{5.7}$$

The CIoU loss is then defined, with a trade-off parameter $\alpha \in \mathbb{R}_{\geq 0}$ and the diagonal $c \in \mathbb{R}_{\geq 0}$ of the smallest enclosing box covering $\mathbf{B}_{\text{p}}$ and $\mathbf{B}_{\text{gt}}$, by

$$\mathcal{L}_{\text{CIoU}}(\mathbf{B}, \mathbf{B}_{\text{gt}}) := 1 - \text{IoU}(\mathbf{B}, \mathbf{B}_{\text{gt}}) + \frac{\|\mathbf{b} - \mathbf{b}_{\text{gt}}\|^2}{c^2} + \alpha v. \tag{5.8}$$

The objectness is evaluated with binary-cross entropy loss (4.9) split up into the cases of an object being present in a grid cell for each bounding box and the case of no object being present. This is done to force the objectness to be zero if no object is present. Finally, the classification task is also measured with binary-cross entropy loss (4.9).

The training of YOLOv4 was performed on the TUHH cluster using an NVIDIA A100-80 GPU. The training of YOLOv4 took aproximately 5 h for the proxy data set and 7 h for the LiDAR data set.

5.3.3 Accuracy Measurement

To evaluate object detectors, we count the true positive (TP), false positive (FP), and false negative (FN) predictions. In classification problems *recall* and *precision* are used as a measure of success. Recall measures the ratio of TP predictions out of all positive samples, i.e., a low recall hints at many FN predictions [93, p. 87]. Precision measures the ratio of TP predictions out of all samples predicted as positive, i.e., a low precision hints at many FP predictions [93, p. 87]. The formulas to calculate precision and recall are [93, p. 86]

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}. \tag{5.9}$$

To assign a class to a bounding box, YOLO uses a class probability, i.e., for each class, a probability of that class being present in a bounding box is predicted. To finally label that bounding box with a class, a threshold $T \in [0, 1]$ is used at which probability a class prediction is used. To find the best threshold and the best compromise between high recall and high precision, a precision-recall curve is used . The precision-recall curve plots the precision against the recall for all thresholds $T \in [0, 1]$ [93]. The average precision (AP) for a given class now is defined as the area under the precision-recall curve [58]. For each class we want to detect, the AP is calculated. The mean average precision (mAP) is the mean of the AP per class and the metric used to evaluate YOLOv4. The mAP can be calculated for different IoU thresholds at which a prediction is considered positive also to reflect the quality of the bounding box.

5.4 Regression Network

The regression network is employed after the YOLOv4 bounding box prediction on the predicted LiDAR scan sections, where vortices can be found. The task is to enhance the localization and include a circulation strength estimation for each vortex.

5.4.1 Network Architecture

As base architecture, we use the CNN from "Characterizing aircraft wake vortex position and strength using LiDAR measurements processed with artificial neural networks" by Niklas Wartha [10], given that it achieved promising results on complete LiDAR scans and is assumed to perform even better on single vortices. Instead of training separate networks for each coordinate prediction and circulation strength prediction, in this work we only split the task into localization and circulation strength prediction such that we have two networks to train. The CNN consists of blocks of convolutional layers followed by max pooling layers, so-called *ConvPool* blocks. Four of such blocks are used, with the convolutional layers having a filter size of 3×3 and the pooling layers having a filter size of 2×2 and a stride of 2. The first layer uses 32 filters which number is subsequently doubled [10]. A graph of the CNN is depicted in Figure 16.

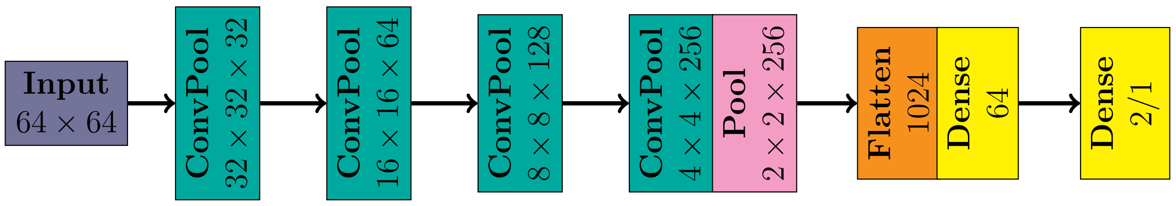


Figure 16: The CNN used for the regression tasks with the output dimensions of each block. The final output dimension depends on whether the vortex center or vortex circulation strength is predicted.

5.4.2 Training

We use ADAM [63] as an optimizer and MSE (4.3) as a loss function to train the CNN. To avoid overfitting we include an early stopping mechanism to stop training if no improvement of the validation loss is seen for 30 epochs. In the section 7 we perform some hyperparameter tuning and evaluate different preprocessing and data augmentation.

Python is the most prominent programming language in machine-learning due to its open-source nature. Therefore, we use Python as the programming language. The machine-learning package we employ for implementing CNN regression is Keras [94]. Keras is a high-level API based on Tensorflow [95]. We use Keras version 2.9.0 and Tensorflow version 2.9.1., which were the latest versions available at the time of starting the work on this thesis. The training of the regression networks was performed on the HOREKA cluster using an NVIDIA A100-40 GPU and took, on average, 30 min for each network.

5.4.3 Accuracy Measurement

Given that we want to evaluate the vortex center and the vortex circulation strength estimation, we use two metrics. The first metric we use is the mean absolute error (MAE) defined for a set $\mathbf{D}$ of N prediction-label pairs $(\hat{\mathbf{y}}_i, \mathbf{y}_i)$ with $i \in \{1, 2, \dots, N\}$ by

$$\mathrm{MAE}(\mathbf{D}) = \frac{1}{N}\sum_{i=1}^{N} |\hat{\mathbf{y}}_i - \mathbf{y}_i|. \tag{5.10}$$

The MAE can be calculated for vectors using the 1-norm or for scalars with the absolute value. In the case of vortex center prediction, the MAE can be considered for the two coordinates separately to get a feeling for which coordinate prediction is more precise. If we replace the absolute error in the sum of (5.10) with the 2-Norm, we obtain the mean absolute distance error (MADE) which can be used to evaluate the overall distance error. The MADE is defined by

$$\mathrm{MADE}(\mathbf{D}) = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|. \tag{5.11}$$

6 YOLOv4 for Wake Vortex Detection

6.1 Data Preprocessing

Before we can use the LiDAR data sets for the training of YOLOv4, the labels and scans have to be converted to match the architecture of YOLOv4 and the required input data type. The required input format of YOLOv4 is an image in the form of ".jpg" or ".png". Also, the labels have to be converted since the original labels of the data set only contain the center of each vortex and the circulation strength but no information about the width and height, which is needed for bounding box predictions. The following sections explain how to transform the labels and LiDAR scans.

6.1.1 Label Transformation

As a label, YOLOv4 needs the center position of a vortex, a width $w_{\mathrm{B}} \in \mathbb{R}_{\geq 0}$, and height $h_{\mathrm{B}} \in \mathbb{R}_{\geq 0}$ of the bounding box. As previously mentioned, the initial vortex spacing b_0 is a standard referencing length in the research on wake vortices. We employ b_0 as the width and height of the bounding box as label. The initial vortex spacing is unique to each aircraft type with wing span B and can be calculated according to equation (2.9) [1]. Furthermore, that choice ensures only one vortex center being present in each bounding box contributing to our goal of individual vortex characterization. The underlying raw LiDAR data is in the form of polar coordinates (see Section 5.1). LiDAR scans of polar coordinates displayed in a grid visually distort the vortices. Hence not sufficient for the usage of b_0 as bounding box dimensions. We therefore only focus on LiDAR scans in the form of cartesian coordinates. The conversion is presented in the following Section 6.1.2. The formula to transform polar coordinates to cartesian coordinates is given by

$$y(R, \varphi) = R\cos(\varphi), \quad z(R, \varphi) = R\sin(\varphi). \tag{6.1}$$

Before normalizing the center coordinates and the bounding box dimensions as YOLO requires, another transformation from the standard cartesian coordinate system towards the coordinate system used in images is needed. By mirroring the coordinate system along the z-axis and shifting the origin of a LiDAR scan $(y_{\mathrm{min}}, 0)$ to $(0, 0)$, illustrated in Figure 17, we transform the coordinate system accordingly.

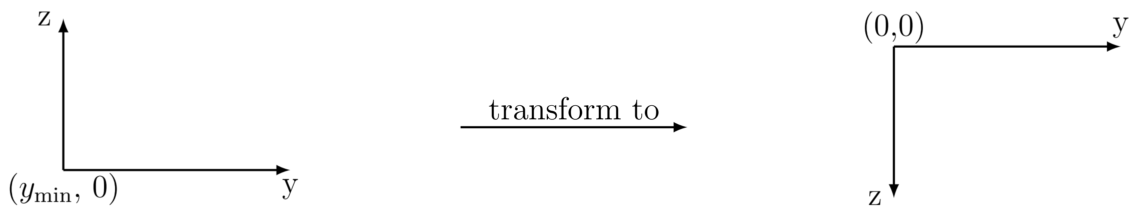


Figure 17: Coordinate transformation from a cartesian LiDAR scan to image coordinates.

Let $(y, z) \in [y_{\mathrm{min}}, y_{\mathrm{max}}] \times [z_{\mathrm{min}}, z_{\mathrm{max}}]$ be the domain of the LiDAR scan with $y_{\mathrm{max}} - y_{\mathrm{min}} = W_{\mathrm{lid}} \in \mathbb{R}$ being the width and $z_{\mathrm{max}} - z_{\mathrm{min}} = H_{\mathrm{lid}} \in \mathbb{R}$ being the height of the LiDAR scan. Let further $(y, z) \in [0, W_{\mathrm{im}}] \times [0, H_{\mathrm{im}}]$ be the domain of an image with image width $W_{\mathrm{im}} \in \mathbb{N}$ and image height $H_{\mathrm{im}} \in \mathbb{N}$. Since the LiDAR scans, in most cases, start at an elevation angle of zero, z_{min} can be considered zero. With the mirroring of the z-axis according to Figure 17, we get the transformation of the LiDAR coordinates to image coordinates by calculating

$$\begin{aligned} y_{\mathrm{im}}(y_{\mathrm{lid}}) &= \frac{y_{\mathrm{lid}} - y_{\mathrm{min}}}{W_{\mathrm{lid}}} W_{\mathrm{im}} \\ z_{\mathrm{im}}(z_{\mathrm{lid}}) &= \frac{z_{\mathrm{max}} - z_{\mathrm{lid}}}{H_{\mathrm{lid}}} H_{\mathrm{im}} \\ w_{\mathrm{im}}(w_{\mathrm{lid}}) &= \frac{w_{\mathrm{lid}}}{W_{\mathrm{lid}}} W_{\mathrm{im}} \\ h_{\mathrm{im}}(h_{\mathrm{lid}}) &= \frac{h_{\mathrm{lid}}}{H_{\mathrm{lid}}} H_{\mathrm{im}}. \end{aligned} \tag{6.2}$$

The last step is to normalize the bounding box coordinates with respect to the image width W_{im} and image height H_{im}. Let $(y_c, z_c) \in \mathbb{R}^2_{\geq 0}$ be the vortex center according to the RV method. We use the transformation (6.2) to transform the labels created by the RV method to bounding box labels $(y_{\mathrm{YOLO}}, z_{\mathrm{YOLO}}, w_{\mathrm{B}}, h_{\mathrm{B}}) \in \mathbb{R}^4_{\geq 0}$ for YOLO by calculating

$$\begin{aligned} y_{\mathrm{YOLO}}(y_c) &= \frac{y_{\mathrm{im}}(y_c)}{W_{\mathrm{im}}} = \frac{y_c - y_{\mathrm{min}}}{W_{\mathrm{lid}}} \\ z_{\mathrm{YOLO}}(z_c) &= \frac{z_{\mathrm{im}}(z_c)}{W_{\mathrm{im}}} = \frac{z_{\mathrm{max}} - z_c}{H_{\mathrm{lid}}} \\ w_{\mathrm{B}} &= \frac{w_{\mathrm{im}}(b_0)}{W_{\mathrm{lid}}} = \frac{b_0}{W_{\mathrm{lid}}} \\ h_{\mathrm{B}} &= \frac{h_{\mathrm{im}}(b_0)}{H_{\mathrm{im}}} = \frac{b_0}{H_{\mathrm{lid}}}. \end{aligned} \tag{6.3}$$

6.1.2 Conversion of LiDAR Scans to Image Format

As already stated, the input data format YOLOv4 requires is an image in the form of ".jpg" or ".png". We use the ".png" format, which supports 16-bit pixel values and uses lossless compression. Thus, more details are kept after transformation, and no information is lost. Before converting the LiDAR scan to an image, we employ feature engineering. LiDAR scans are transformed to a universal measurement grid, low LOSs and crosswind are removed, and a CNR filter is applied [10].

A universal measurement grid is required as different LiDAR position measurements result in different grids (see Table 2). The largest scan originates from LiDAR position L5 with a domain of approximately $[70\ \mathrm{m}, 530\ \mathrm{m}] \times [0\ \mathrm{m}, 256\ \mathrm{m}]$ and average step sizes of approximately $\Delta y = 2.87$ m and $\Delta z = 1$ m. Based on those values and YOLOv4's requirement for an image to have dimensions which are multiples of 32, we use a universal equidistant image grid domain of $[70\ \mathrm{m}, 530\ \mathrm{m}] \times [0\ \mathrm{m}, 256\ \mathrm{m}]$ and feature according step sizes. To choose a suitable resolution, we consider the recommendation that the smallest object to detect should be larger than 16×16 pixels [96]. The smallest wake vortex in the data set defined by b_0 is produced by aircraft A320 and A20N with a value of $b_0 = 26.8$ m. We already match that requirement if we keep the step size z-direction at $\Delta z = 1$ m, leading to an image height of $H_{\text{im}} = 256$. In the y-direction, we would only have approximately 9 pixels with the given $\Delta y = 2.87$ m. Hence we have to choose a smaller step size. As the width of our domain is 460 m we choose the closest multiple of 32 as the number of pixels used in the y-direction, which is $W_{\text{im}} = 448$, such that we have a step size of approximately $\Delta y = 1.03$ m.

As a result of the LiDAR measurement starting either at the highest or lowest elevation angle, we first have to interpolate the radial velocities from a LiDAR scan onto a general grid matching each LiDAR position values from Table 2 by nearest-neighbor interpolation [97, p. 45]. Nearest-neighbour interpolation is employed considering the number of radial velocities in each measurement position is consistent. Given that, after the transformation from polar coordinates to cartesian coordinates, the measurement grid is no longer equidistant, we use linear interpolation to get the radial velocities at the image grid points.

To obtain a complete image, the empty space above the LiDAR scan, which can be seen in Figure 10b, has to be filled with values. It can be assumed that the wind speed is vertically stratified [98]. Consequently, those missing values are filled constantly with the average radial velocity at a given height. Given that we further assume the wake vortices to be parallel and in pairs in most scans, an impact on the average radial velocity can be considered negligible. Since the maximal height z_{max}, of LiDAR scans at the positions L1-L4 are lower, we also have to fill up values above z_{max}. To fill those values, we pad with the edge radial velocities given at z_{max}.

Removing the crosswind in LiDAR scans has been shown advantageous [10, 32]. A LiDAR scan measured before the overflight of an aircraft is used as a background scan. This background scan is also interpolated onto the universal image grid. Under the assumption of vertically stratified wind speeds, we calculate an average wind speed $\overline{V}(z)$ according to (6.5) with respect to the height z. From the LiDAR scan, we subtract the average wind speed as follows

$$\widetilde{V}_r(y_i, z_j) = V_r(y_i, z_j) - \overline{V}(z_j) \quad i \in \{0, 1, \ldots, W_{\text{im}}\},\ j \in \{0, 1, \ldots, H_{\text{im}}\}, \tag{6.4}$$

$$\overline{V}(z_j) = \frac{1}{W_{\text{im}}} \sum_{i=0}^{W_{\text{im}}} V_r(y_i, z_j). \tag{6.5}$$

To keep the notation clean, we expect $\widetilde{V}_r$ from (6.4) as given and ignore the tilde in the further course of this thesis.

The reason to remove low LOSs is to mitigate the impact of plate lines and secondary vortex structures in the measurements since they might lead to large velocity gradients, which could overshadow patterns of the wake vortices [10]. As a threshold, we choose a minimum height above the runway centerline of 7 m. Depending on the LiDAR position, all values below a linear function with a slope of $7/d_{\text{runway}}$, with d_{runway} being the distance of the LiDAR to the runway center, are removed. Those removed values are filled horizontally with the average radial velocity at the respective height.

To mitigate measurement errors, we use a CNR-filter to remove all radial velocities measured with a CNR higher than a given threshold T_{CNR}. We assume a threshold of $T_{\text{CNR}} = 0.02$ to be sufficient [99].

After those preprocessing steps are done, we convert the resulting radial velocities into pixel intensities. The ".png" image format supports 16-bit pixel values, such that we have to transform the radial velocities from an interval of $[V_r^{\min}, V_r^{\max}]$ to an interval of $[0, 2^{16} - 1]$. We first apply a *Min-Max normalization* to scale the radial velocities onto an interval of [0,1]. This normalization method is suggested in [38]. After that, we scale the values to a maximal value of $2^{16} - 1$ and round them to the next integer value. The transformation is given by

$$V_r^{\text{pixel}}(V_r) = \frac{V_r - V_r^{\min}}{V_r^{\max} - V_r^{\min}}(2^{16} - 1) \tag{6.6}$$

6.2 YOLOv4 Evaluation

6.2.1 Training

Three different data set groups introduced in Section 5.1 are utilized for the training of YOLOv4. LiDAR scans are expected to be homogeneous inside a group only containing LiDAR scans from one plate line scenario and heterogeneous combined. Therefore, the patterns found in LiDAR scans from different plate line scenarios might differ. First, the success of the different data set groups is evaluated to decide whether differently trained YOLOv4 versions should be used for the different plate line scenarios or whether only one model for both cases is sufficient. Subsequently, the final YOLOv4 model is evaluated in terms of accuracy and different detection parameters.

Before training on the LiDAR scans, we train YOLOv4 on a small data set of proxy scans. The data set contains 1000 proxy scans, with 500 scans having only a single port vortex and 500 scans having a port and starboard vortex pair. We split that data set into 800 proxy scans as the training data set and 200 proxy scans as the validation data set. Supplementary test data sets are employed to evaluate the capability of YOLOv4 to detect more than two vortices, despite being trained with proxy scans containing only one or two vortices. We use two test data sets with 200 proxy scans each. The first contains two vortex pairs, and the second contains a vortex pair and an additional port vortex.

Since using a pre-trained version of Faster R-CNN on the Microsoft Common Objects in Context data set (COCO) [100] gave promising results [44], we use a version of YOLOv4 pre-trained on the COCO data set as well. COCO data set consists of 328 000 images containing 2 500 000 labeled instances of common objects from 91 classes [100].

The results after 345 epochs of training can be found in Table 5. The mAP at an IoU threshold of 50% (mAP@50) for scans containing three or four vortices is found to be above 90%, despite training YOLOv4 only with scans containing one or two vortices. Consequently, we assume that YOLOv4 is sufficient for the wake vortex detection task and generalizes well on a different number of vortices in a LiDAR scan. The accuracies in Table 5 also represent a benchmark for the training on the LiDAR data set.

Table 5: The mAP of YOLOv4 trained on proxy scans.

	Validation Data Set	Three Vortices Test Data Set	Four Vortices Test Data Set
mAP@50	94.91%	90.22%	91.81%

6.2.2 Data Set evaluation

As explained in section 5.1, the data set can be categorized into three different data set groups. One data set group with only scans having the plate lines used, one data set group without using plate lines and the combined data set group. To evaluate which training works best for YOLOv4, we train it with those three data set groups separately. The three resulting YOLOv4 models generated by training with each training data set group, are validated with each validation data set group. We employ the mAP@50 as accuracy metric, as it is the standard choice for YOLOv4.

The mAP for all different scenarios can be found in Table 6. As we already have trained YOLOv4 on proxy scans, we can utilize this model as a pre-trained version of YOLOv4 and continue training with the LiDAR data set groups. The YOLOv4 model trained with the combined data set group - containing both plate line scenarios - has the best mAP for all validation data sets. We thus continue improving that model. This result is the same for YOLOv4 pre-trained on COCO and proxy scans. In the further course of this thesis, we only consider the data set group containing both plate line scenarios for training and validation. The difference between the mAP on the model pre-trained on COCO and the model pre-trained on proxy scans is negligible. For further training, we use YOLOv4 pre-trained on COCO as it was also utilized by [45].

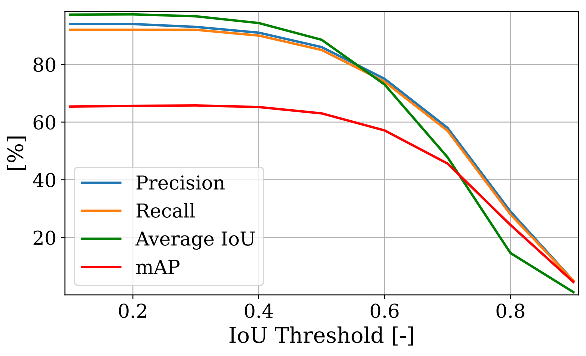
Table 6: The mAP of YOLOv4 for the three different validation data sets after training with the respective training data sets. The highest mAP for each validation data set is marked in green.

Pre-trained on COCO			
Validation Data Set / **Training Data Set**	**Both**	**Down**	**Up**
Both	87.96%	86.79%	89.15%
Down	84.61%	85.55%	83.96%
Up	83.47%	84.22%	82.92%
Pre-trained on proxy scans			
Validation Data Set / **Training Data Set**	**Both**	**Down**	**Up**
Both	87.96%	86.24%	89.80%
Down	83.19%	84%	82.60%
Up	82.72%	83%	82.66%

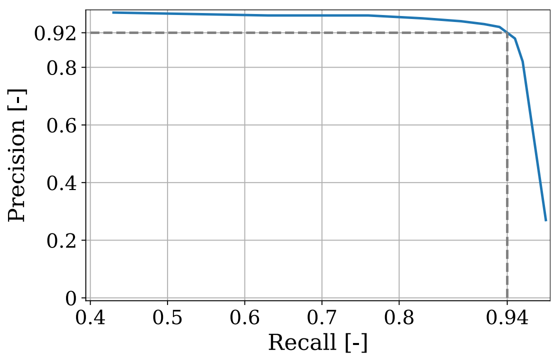
By increasing the number of training steps from 14700 to 20000, we push the mAP of the chosen setting even further to 88.53%, which is already close to the mAP of YOLOv4 trained with proxy data (see Table 5).

6.2.3 Detection Parameter Setting

Since we use the YOLOv4 bounding box predictions as input for the regression network, we have to investigate the parameters used for detection. In particular, the confidence threshold used for detection has to be considered. We use the regression CNN to enhance the vortex center prediction. Hence a lower IoU can be considered for validation. So far, we have always used the mAP for an IoU threshold of 0.5 to evaluate our model. In Figure 18a, precision, recall, mAP, and average IoU for different values of IoU thresholds can be seen. Since the precision and recall are the highest at an IoU threshold of 0.25, we evaluate this case further for different confidence thresholds. In Figure 18b, the Precision-Recall curve is plotted for an IoU threshold of 0.25. The best point is marked at a precision of 0.92 and a recall of 0.94, with the confidence threshold being $T_c = 0.25$. If we favor recall



(a)

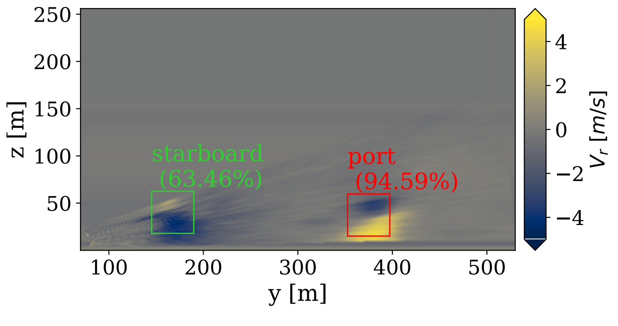


(b)

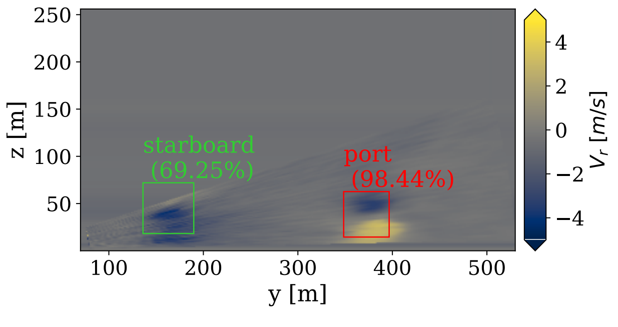
Figure 18: (a) A graph of the precision and recall, as well as a graph of the average IoU and the mAP with respect to the IoU threshold used. (b) Precision-Recall curve with the optimal precision-recall pair marked.

over precision, a confidence threshold of 0.2 would be the choice. Given that a lower recall explains increasing numbers of TP predictions and decreasing numbers of FN predictions, more vortices are captured. In the case of wake vortex prediction, the conservative choice, of capturing more vortices, is preferred. Therefore we apply a confidence threshold of 0.2 to create the bounding box predictions that are fed into the regression net.

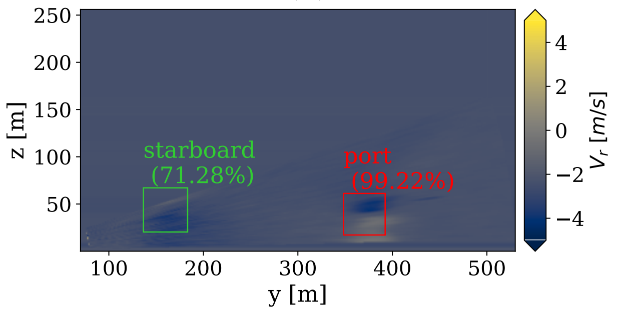
Considering the RV method creating the ground-truth labels also has inaccuracies, we assume the confidence threshold of 0.2 to be the better choice and continue to use it for detection. Figure 19 shows an example of wrong YOLOv4 prediction according to the RV label. The first scan is predicted correctly according to the RV label. In the two consecutive scans, the RV method no longer detected the starboard vortex, but YOLOv4 has. As previously discussed, the vortex trajectory is a hyperbola without crosswind, but with a crosswind, the upwind vortex can be expected to stall over the runway [1]. This is the case seen in Figure 19. Hence we assume the YOLOv4 prediction to be correct and that the RV method could not detect the starboard vortex. Therefore, leads to the assumption that the actual mAP, precision, and recall are higher than evaluated as we validate using labels generated by the RV method.



(a)



(b)



(c)

Figure 19: Three consecutive LiDAR scans from the same overflight with YOLOv4 bounding box predictions. (a) Correctly predicted bounding box according to RV method. (b)-(c) False detection of starboard vortex according to RV method.

6.2.4 Accuracy

To evaluate the accuracy of the YOLOv4 network with a confidence threshold of 0.2 even further, we assume that the center of the bounding box matches the center of the predicted vortex. The rationale is that the vortex center is used for the bounding box center label. We use the validation data set for all those tests containing both plate line scenarios. After evaluating the validation data set, we find the mean confidence to be 84.54% and the median confidence to be 91.63%. With that result, we can confirm that the bounding box predictions are of good quality even with a low confidence threshold. To evaluate the accuracy of the center prediction from YOLOv4, we use the Euclidean distance of the bounding box center to the vortex center label, i.e., MADE (5.11). Furthermore, we separately evaluate the MAE (5.10) in the y- and z-directions to understand which coordinate prediction is more accurate. The results can be found in Table 7. We can see that the accuracy of the z-coordinate predictions is significantly higher. This can be explained by the larger velocity gradient present in the z-direction, caused by the perpendicular measuring of radial velocities only measuring the velocities in the direction of the LiDAR's LOS. Based on these values, we can evaluate the enhancement of vortex center predictions by the following regression CNN.

Table 7: MADE as well as MAE for y- and z-coordinates evaluated separately for each plate line scenario.

Plate Line Scenario	MADE	MAE (y)	MAE (z)
Both	6.26 m	5.86 m	1.41 m
Up	6.54 m	6.18 m	1.43 m
Down	5.96 m	5.52 m	1.39 m

The difference between the MADE with plate lines and without is 0.58 m, the MAE difference in y-coordinates is 0.66 m, and for z-coordinates, it is 0.04 m. Those differences are negligible, normalized by the minimal initial vortex separation $b_0 = 26.8$ m for aircraft A320 and A20N giving a percentage error. We end up with a rounded percentage error of 0.02 for the MADE and the MAE in y-coordinates and an even lower percentage error for the z-coordinates.

To further evaluate the quality of the vortex center prediction, we compare each prediction's absolute distance error (ADE) with the circulation strength target Γ_t of the respective vortex in Figure 20. We can see that YOLOv4 is more accurate at predicting the vortex center of stronger vortices. Those vortices, in most cases, belong to aircraft of the heavy category.

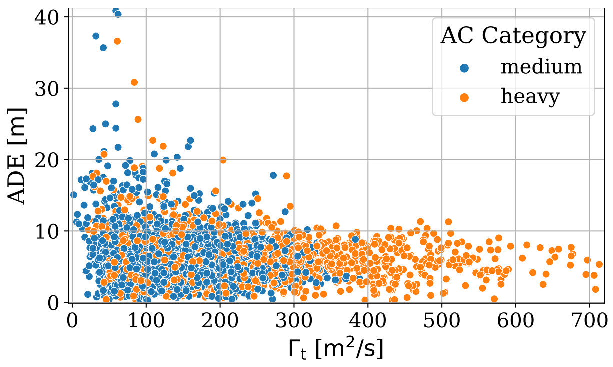


Figure 20: Relationship between the ADE of the YOLOv4 prediction and circulation Γ_t of the respective vortex.

7 Regression Network for Wake Vortex Characterization

After we obtain the bounding box predictions, we employ a regression CNN based on [10] to enhance the localization and perform a vortex circulation strength estimation. As input for the CNN, we use the cut-out vortices. In the following, the term vortex is used equivalently to the cut-out vortices from bounding boxes. Before the network can be trained, we must preprocess the vortices and labels. The original labels refer to the whole scan, so we must translate the labels to match the coordinate system of the vortices.

7.1 Data Preprocessing

The YOLOv4 network uses scans converted to images as input. Due to that conversion, we assume to lose important physical information needed to estimate the circulation strength [101]. For that reason, we use the original LiDAR scans as base data. As the predicted bounding boxes can vary in size, but the regression CNN needs a constant input dimension, we have to choose a sufficient dimension as input. The bounding box size depends mainly on the initial vortex separation b_0, given that we labeled the data based on that (see Section 6.1.1). The largest initial vortex separation of the data set is $b_0 = 62.7$ m (A380). Therefore, a maximal width and height of 64 m, including a safety factor, is assumed to be sufficient. To test that assumption, we check the width and height of every bounding box prediction on the complete data set, including both training and validation data. The result is summarized in Table 8. Since more than 99% of predicted bounding boxes have a width and height below 64 m, we see the assumption confirmed and use matrices $\mathbf{x} \in \mathbb{R}^{64\times64}$ as input for our regression network. Besides that, we already know the MADE of YOLOv4, which is 6.26 m on the validation data set. Hence the vortex center is contained.

Table 8: Evaluation of predicted bounding box width and height. 29 829 bounding box predictions are used for evaluation.

	Mean	**Median**	**99-th percentile**	**Maximum**	**Below 64 m**
BB width	35.16 m	32.73 m	61.02 m	86.68 m	99.43 %
BB height	34.6 m	32.41 m	57.04 m	74.36 m	99.78%

Given that our goal is to individually characterize vortices, we do not only consider the center of the predicted bounding box and cut out a patch of dimensions 64 m × 64 m. As previously mentioned, we chose the bounding box width to be the initial vortex separation b_0 to ensure only one vortex center is contained in a bounding box. Accordingly, the predicted bounding box width and height are used to cut out the vortices.

If a bounding box is larger than 64 m in any dimension, we crop the bounding box symmetrically to have the required dimension. If a bounding box is smaller than 64 m in any dimension, we pad the vortex with the mean vortex radial velocity symmetrically, such that the vortex is centered.

The problem of YOLOv4 predicting more vortices than labeled, is addressed by matching labels with bounding box predictions. For each scan, we match each predicted vortex to the closest ground-truth vortex with respect to its center. Considering the largest bounding box, being 56 m $\times$ 56 m, the farthest point on the edge of that bounding box is 39.6 m away from the center. The prediction is ignored if the closest ground-truth vortex center is further away than a threshold of 40 m.

In the data set with both plate line scenarios, we have 25 315 vortices labeled in the training data set and 3 389 vortices labeled in the validation data set. After matching the vortices with the correct labels, we end up with a data set of 24 693 vortices in the training data set and 3 249 vortices in the validation data set. After the bounding box prediction, 95.86% of vortices in the validation data set remain, and 97.54% of the vortices in the training data set remain. These values match the accuracy of YOLOv4, showing that the threshold used for matching was sufficient. The reason for losing a higher percentage in the validation data set is that ANNs usually perform better on the data they have been trained with than on the data they have never seen before [50].

7.2 Radial Velocity Preprocessing

It is advantageous to incorporate prior knowledge of the data utilized for training and validation, and transform it respectively to simplify the job for the machine-learning model [59]. To gain prior knowledge of the data set, we take a closer look at each vortex's radial velocities. A common preprocessing strategy is to normalize the data set to have a mean of zero and a standard deviation of one [59]. Since this was useful for other wake vortex estimation CNNs, we normalize the complete data set to have a mean of zero and a standard deviation of one [10]. We achieve this by calculating the overall mean radial velocity $\overline{V}_r$ and the standard deviation σ_{V_r}. We then subtract the mean and divide by the standard deviation to obtain

$$\widetilde{V}_r = \frac{V_r - \overline{V}_r}{\sigma_{V_r}}. \tag{7.1}$$

Another feature taken into account is the maximal radial velocity in each vortex. Large values in the input can lead to large gradient updates that will prevent the network from converging [59]. To remove radial velocity outliers, we find a threshold value at which radial velocities are clipped. A histogram of the absolute maximal radial velocity per scan can be found in Figure 21. Based on that, we assume a threshold of 20 m/s to be sufficient since 99% of all absolute radial velocities lie below 17.81 m/s.

To evaluate that threshold, we test the effectiveness of clipping radial velocities on the base CNN introduced in section 5.4. The training results, employing radial velocity clipping, can be seen in Figure 22a and 22b compared to a lower threshold of 10 m/s and no threshold. In the localization prediction clipping, the radial velocity at a threshold of 20 m/s works best. The impact it has in the case of circulation strength prediction is

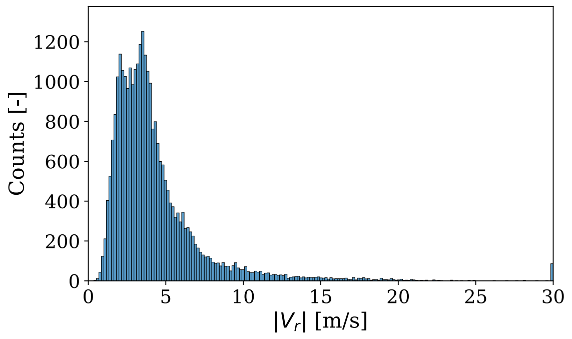


Figure 21: Histogram of the maximal absolute radial velocity in every vortex of the data set. The spike at 30 m/s originates from clipping all velocities above 30 m/s for illustration purposes.

negligible. To keep the networks consistent, we incorporate a radial velocity clipping at a threshold of 20 m/s in the localization and circulation strength prediction.

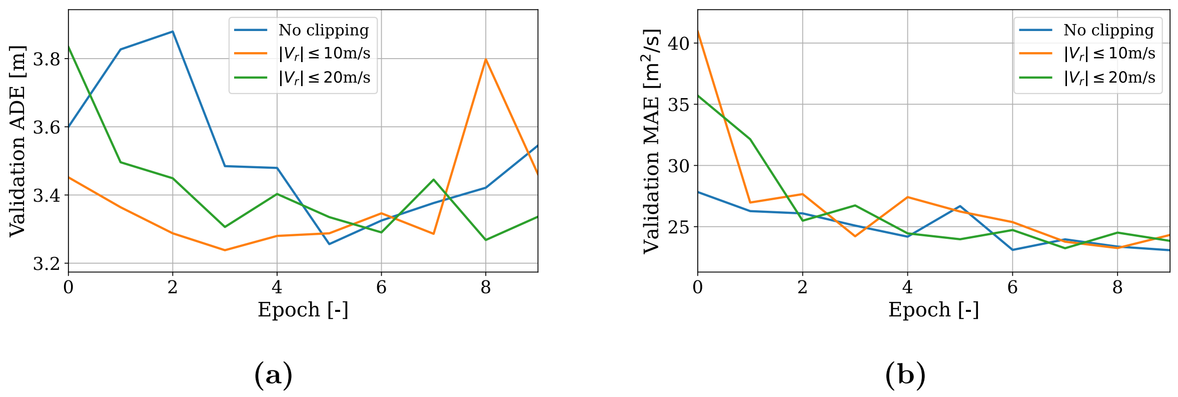


Figure 22: Radial velocity clipping experiment on localization CNN training (a) and circulation CNN training(b).

7.3 Regression Network Evaluation

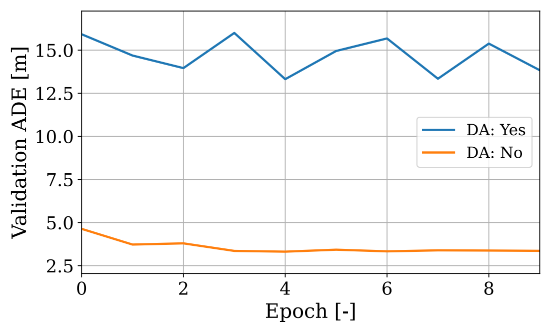
7.3.1 Hyperparameter Studies

We already studied the effect of radial velocity clipping in Section 7.2. In this section, we further investigate the effect of *data augmentation* (DA) during training times, which is a powerful technique to mitigate overfitting in computer vision tasks [59]. Standard DA includes rotation, translation, zooming, shearing, and horizontally flipping [59]. The translation is the only DA process that does not alter the physical information a vortex contains. Thus only translation is considered for DA. Since we use a canvas of 64×64 pixels as an input for the CNN and most of the bounding box predictions are smaller, we can shift the vortex horizontally and vertically in that canvas and adjust the vortex center label, respectively.

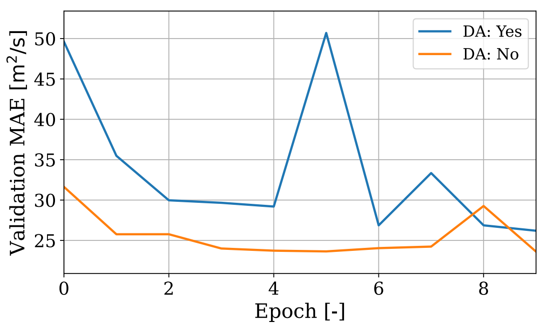
Given that we estimate vortex parameters, the idea is to have a larger filter size in the first layer such that we capture more information about the vortex at first sight. To evaluate this hypothesis, we conduct training of the basic CNN but change the filter size in the first layer to 9×9 instead of 3×3. A summary of the training processes with all those different hyperparameter settings can be found in Figure 23.

We can see that the training process without DA is highly superior in the case of vortex center prediction (Figure 23a) but not so much in the case of circulation strength prediction (Figure 23b). We assume this is the case since the circulation strength of a vortex is independent of its position. However, when shifting the vortex, we also shift the vortex center, making it harder for the network to learn the vortex center parameters. For that reason, we chose not to use DA.

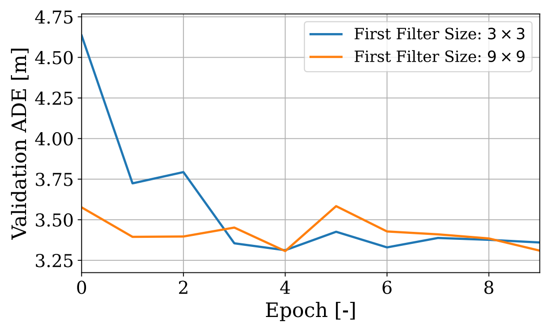
The impact of the filter size in the first layer can be seen in Figure 23c and Figure 23d. We can tell that the training process does not improve using a larger filter in the first layer. Hence we stick to the initially proposed CNN.



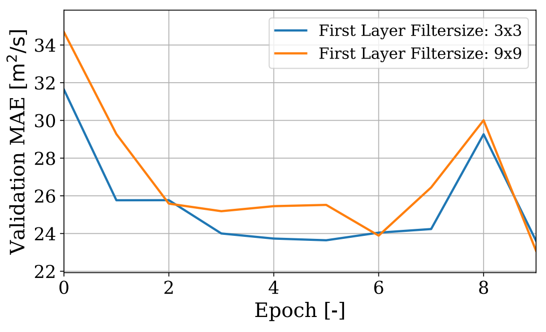
(a)



(b)



(c)



(d)

Figure 23: Training evaluation with validation data and the respective metric. DA experiment on localization (a) and circulation (b). First layer filter size experiment on localization (c) and circulation (d).

A test of different activation functions, e.g., ReLU and Mish, also showed no difference in training. Therefore we keep using ReLU as an activation function in our CNN.

Batch normalization after the convolutional layers also showed no improvement. Since we could not see an advantage we omit batch normalization to not distort the physical properties.

For the final training we use the initially proposed network but employ radial velocity clipping as an additional preprocessing step. The CNN for vortex center estimation has 453 570 and the CNN for vortex circulation strength estimation has 453 505 trainable parameters. As a regularization factor, we use an early stopping mechanism during training [59]. This method was shown to be sufficient in a foregoing application of CNNs for wake vortex parameter characterization with a maximal epoch number of 100 and an early stopping after 30 epochs without improvement [10]. As a batch size we employ 128, such that we can fully utilize the GPU at hand.

7.3.2 Circulation

The CNN model to predict circulation strength only differs from the CNN model for localization in the output dimension. Circulation is a scalar property, thus the output dimension of the CNN is set to be one corresponding to scalar regression.

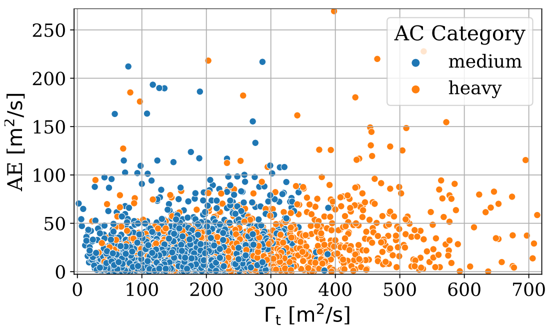
We first test the network's performance on perfectly cropped vortices, i.e., vortices based on the bounding box labels. Then we train another network on the YOLOv4 output vortices. The results can be seen in Table 9. We can see that we could nearly achieve the same accuracy with the YOLOv4 output as with the perfectly cropped vortices. Hence we assume that we can not further improve the accuracy without changing the model architecture significantly. Further investigations are done only considering the output of YOLOv4.

Table 9: MAE of the validation data set from the data set cropped by labels and the data set cropped by YOLOv4's prediction.

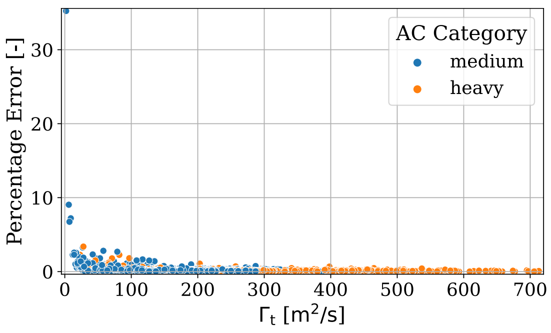
	Label Crop	**YOLOv4 Crop**
MAE	22.31 m^2/s	22.97 m^2/s

As previously mentioned, we found a correlation between the circulation strength and ADE of the YOLOv4 vortex center prediction (Figure 20). In the case of circulation strength prediction, we do not have a correlation between the absolute error (AE) of the circulation CNN and the corresponding circulation strength target. This can be seen in Figure 24a. That we can still have a comparison here, we consider the percentage error (PE), which is the AE divided by the annotated circulation strength. The percentage error can be seen in Figure 24b. Here we can see a slightly higher PE for low circulation with outliers, i.e., the accuracy is higher for strong vortices.

The goal, in the end, is to detect hazardous wake vortices. The hazard depends on the aircraft encountering and the path through the wake vortices [32], which can not be considered here, but also on the vortex circulation strength. Since 100 m^2/s was chosen as a suitable threshold in [10], we classify a vortex as hazardous if the circulation is above



(a)



(b)

Figure 24: (a) Relationship between AE and circulation Γ_t of the respective vortex. (b) Relationship between PE and circulation Γ_t of the respective vortex.

that threshold. With this threshold, we can evaluate the precision and recall by applying equation (5.9) to evaluate the CNN on the validation data set. The precision of the circulation CNN is 96.11 %, and the recall is 95.28%

7.3.3 Localization

Since we do not use DA, the vortex center of a perfectly cropped vortex is always in the center of the input data. Hence the CNN only trains to predict that the vortex center coincides with the center of the input data for perfectly cropped vortices. We, therefore, can not get benchmark values on how a network would behave on perfect bounding box predictions.

The results of the CNN for localization can be found in Table 10. To evaluate the improvement in comparison to the YOLOv4 center prediction, we use Table 7. The CNN improved the overall MADE with the CNN by 3.18 m, which is approximately half the error in comparison to solely using YOLOv4. If we take a closer look at which improvement accounts for the decrease in the localization error, we see that the MAE for the y-coordinate was more than halved from 5.86 m to 2.29 m. The MAE of the z-coordinate increased in the CNN prediction by 0.16 m. Hence the MADE improvement is due to a better y-coordinate prediction. The slightly higher MAE in z-coordinates of 0.16 m for both cases is negligible as normalized by the minimal initial vortex separation $b_0 = 26.8$ m is only, rounded to three decimals, 0.006.

Table 10: MADE as well as MAE for y- and z-coordinates separately for each plate line scenario by the CNN regression.

Plate Line Scenario	**MADE**	**MAE (y)**	**MAE (z)**
Both	3.08 m	2.29 m	1.57 m
Up	3.10 m	2.30 m	1.58 m
Down	3.06 m	2.27 m	1.56 m

Again we can see that the prediction in the case of no plate line usage is insignificantly better. While for YOLOv4 center prediction, we had a MADE difference between the up and down cases of 0.58 m, we now have a difference of 0.04 m. Hence we can assume that the plate line scenario has no significant impact on the quality of the vortex center prediction. The relationship of the ADE and the circulation strength target Γ_t looks similar to the one of YOLOv4 (see Figure 20) and can be seen in Figure 25. The main difference is that we now have a lower mean, and we see less outliers.

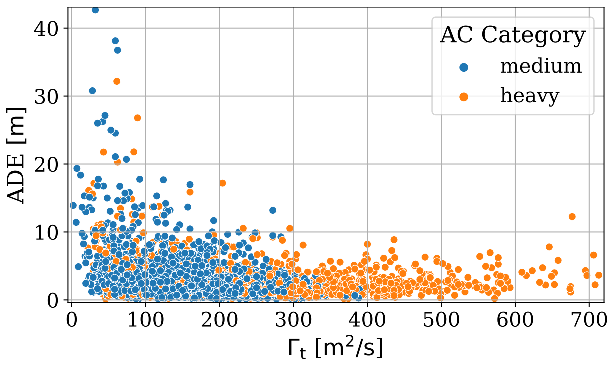


Figure 25: Relationship between the ADE of the localization CNN and circulation Γ_t of the respective vortex.

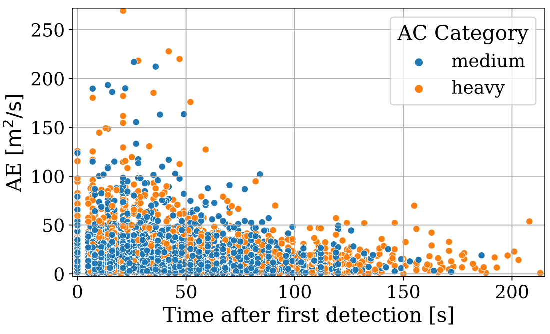
7.3.4 Physical Evaluation

In this section, we evaluate how far the physical properties of a vortex play a role in the prediction accuracy. We focus on the prediction accuracy in relationship with the time after vortex detection, i.e., the vortex age. As the circulation strength of a vortex decreases with age [1], we could expect from previous results that the localization error increases with the vortex age. Furthermore, as we could not find a correlation between the circulation strength and the circulation strength absolute error, we expect the circulation strength prediction to not correlate with vortex age. In Figure 26, we can see the localization and circulation error compared to the time after first detection of the vortices.

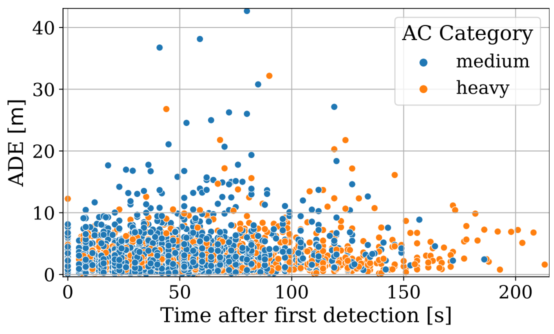
The MAE for circulation strength prediction interestingly decreases with the vortex age. As explained in Section 2.2, a wake vortex evolves in four phases. Hence a reason for decreasing circulation MAE with relative vortex age could be that the roll-up process has to finish to detect the vortex circulation better. We do not see any correlation with the vortex age for the localization error.

7.4 Explainable AI

EASA defines AI explainability as the “Capability to provide the human with understandable, reliable, and relevant information with the appropriate level of details and with appropriate timing on how an AI/ML application is coming to its results.” [38, p. 56]. Multiple techniques exist to visualize what a CNN learns. We focus on the first layer



(a)

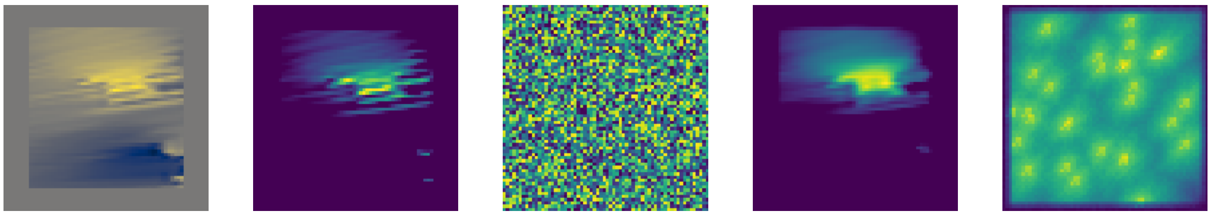


(b)

Figure 26: (a) Relationship between the AE of predicted Circulation Γ and vortex age relative to first detection. (b) Relationship between the ADE of the predicted vortex center and vortex age relative to the first detection.

of both regression CNNs and visualize the intermediate output of the first layer and the filters of that layer, following [59, pp. 160 - 172].

In Figures 27b and 27d, we can see the output of the first convolutional layer given the starboard vortex, depicted in Figure 27a, as input of both the localization 27b and circulation 27d CNN, respectively. We notice that the activations of both networks look similar in the first layer. The output of the circulation CNN seems to look smoother and capture more general vortex information. Although we pad the vortex, it can be seen that the networks only use the vortex for their information gain. In general the first layers carry basic information, and the last layers carry more pieces of information toward the target [59, p. 166]. As the input is the same on both networks and the physical information to extract is related, it is reasonable that both networks first look for gradients inside a vortex.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 27: (a) The starboard vortex employed to visualize first layer activations. Localization CNN:(b) The first feature map activation output and (c) the input maximizing the response of the first feature map. Circulation CNN: (d) The first feature map activation output and (e) the input maximizing the response of the first feature map.

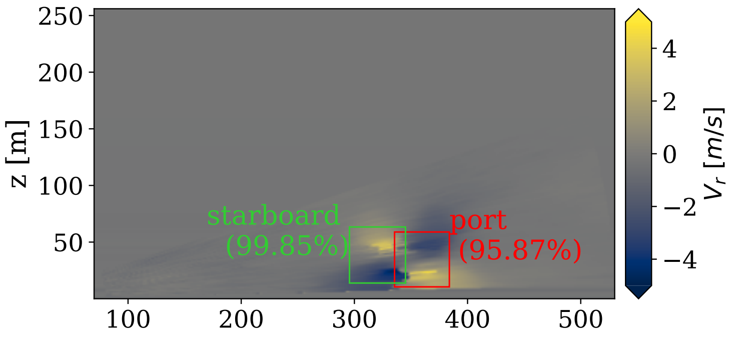
A technique to visualize what the first layers of a CNN are looking for is to maximize the activation by finding the best input. As training a network is done with gradient descent methods, we can use gradient ascent to maximize the response of a layer given a blank input to start with [59, p.167]. The output then provides an input to which each filter is maximally responsive. The visualization of the first filter of the first convolutional layer can be seen in Figure 27c for the localization CNN and Figure 27e for the circulation CNN. The localization CNN seems to respond to high radial velocity patches in a vortex, but at a smaller scale than the circulation CNN. This explains why the activation of the circulation CNN looks more smoothly than the one of the localization CNN.

7.5 Complete Prediction Pipeline

After training, the prediction time of the complete pipeline can be evaluated. An example of one LiDAR scan being processed can be found in Figure 28. Although the bounding box prediction is made on an image converted from a LiDAR scan, we use the radial velocities and not the pixel intensities for illustration purposes. The original input LiDAR scan has previously been presented in Figure 10a and converted to cartesian coordinates in Figure 10b.

The pipeline includes the preprocessing and transforming of a LiDAR scan to an image. After that, the bounding box prediction by YOLOv4 is made, illustrated in Figure 28a. Based on the bounding boxes, the vortices get cropped from the original scan (Figure 28b,28c) and preprocessed according to previously explained feature engineering. Those vortices are fed into the circulation prediction CNN and the localization prediction CNN. The predictions of the CNNs can be found in Figure 28d and Figure 28e, for the starboard and port vortex, respectively. The combination of the YOLOv4 prediction and the CNN prediction is depicted in Figure 28f.

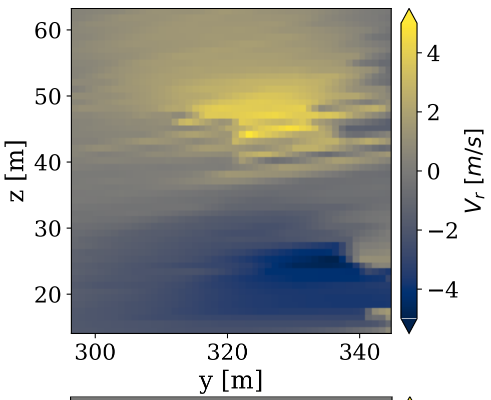
The average time this pipeline takes is 0.13 seconds on the HoreKa supercomputer with an NVIDIA A100-40 GPU and an Intel Xeon Platinum 8368 CPU. The CNN-only approach took 0.16 seconds for evaluation but was also performed without the usage of a GPU [10].

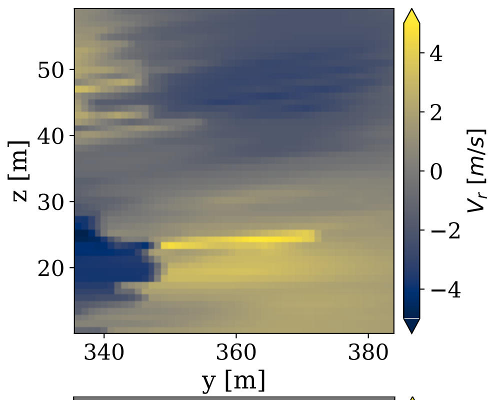


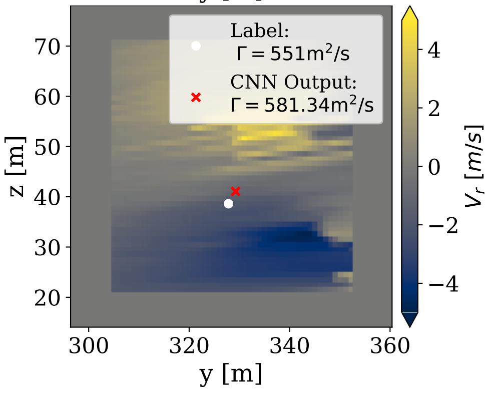
(a) YOLOv4 Predicition

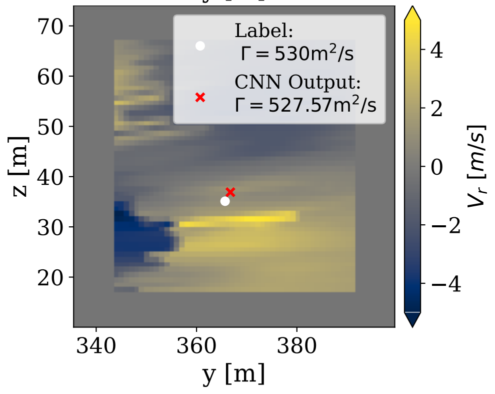
(b) Cropped Starboard Vortex

(c) Cropped Port Vortex



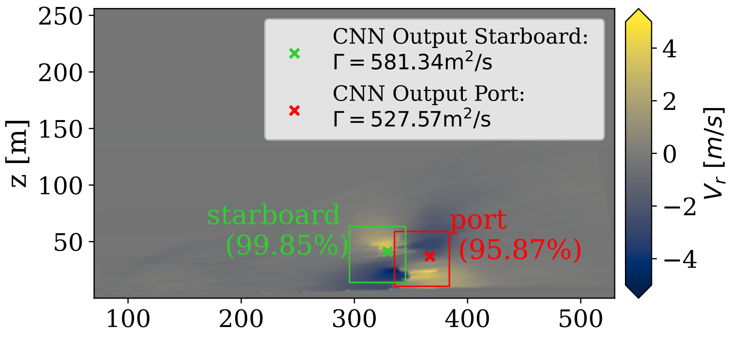






(d) CNN on Starboard Vortex

(e) CNN on Port Vortex



(f) CNN Prediction included in YOLOv4 Prediction

Figure 28: An illustration of the complete prediction pipeline excluding the initial preprocessing step.

8 Comparison with the state-of-the-art

Since the labels for our data set were created with the RV method, we can only give a qualitative comparison. We must expect the accuracy of our approach to be, at most, the one used for labeling the data set. Given that the labels created with the RV method, might contain inaccuracies, and the data used also contains inaccuracies due to the nature of LiDAR measurements, a natural accuracy limit is given.

The traditional wake vortex characterization methods, the RV and VE methods, predict the vortex center in polar coordinates. Only the VE method provides an error in terms of absolute distance that we can compare our approach with. Those errors are only theoretical estimations [35, 36]. The median ADE for the VE method is 7.91 m [10]. YOLOv4 has a median ADE of 5.78 m, and the YOLOv4+CNN approach can reduce this to a median ADE of 2.24 m. As the RV method has an accuracy similar to the VE method for the vortex center estimation, we can compare both methods to our approach. The median ADE is of the same magnitude, so that we can say the prediction is of similar quality as the RV and VE method.

The most recent and only approach to predicting wake vortex parameters with an ANN can be found in [10] with the best results using a CNN. A comparison of the prediction of the herein presented approach - first using YOLOv4 and then using a CNN - with the results of the CNN from [10] can be seen in Table 11. The prediction accuracies were increased in all categories independent of both, the plate line scenario and the vortex class.

Table 11: Comparison of YOLOv4+CNN with the CNN approach from [10] for both plate line scenarios and vortex classes separately.

Plate Lines Up				
ANN	**Port MAE** $\Gamma[\mathrm{m}^2/\mathrm{s}]$	**Starboard MAE** $\Gamma[\mathrm{m}^2/\mathrm{s}]$	**Port MADE** [m]	**Starboard MADE** [m]
YOLOv4	-	-	7.59	5.30
YOLOv4+CNN	21.40	23.88	3.26	2.91
CNN-only [10]	31.08	33.21	46.90	46.70
Plate Lines Down				
ANN	**Port MAE** $\Gamma[\mathrm{m}^2/\mathrm{s}]$	**Starboard MAE** $\Gamma[\mathrm{m}^2/\mathrm{s}]$	**Port MADE** [m]	**Starboard MADE** [m]
YOLOv4	-	-	6.64	4.94
YOLOv4+CNN	20.88	27.22	3.23	2.82
CNN-only [10]	25.76	32.24	21.89	22.70

The absolute and relative improvement of the prediction pipeline in comparison to the CNN only approach is captured in Table 12. We could achieve a significant improvement of more than 93 % in the localization prediction accuracy in the case of plate line usage and more than 85 % without plate line usage. The improvement of the circulation prediction is also more significant in the case of plate line usage compared to the scenario without plate lines. For circulation strength prediction we achieved a minimal accuracy improvement of 15.52% for starboard vortices in the case without plate lines. The maximal improvement of the circrulation strength accuracy was achieved for port vortices and plate line usage with an improvement of 31.15%. Compared to localization accuracy, the circulation strength accuracy improvement is significantly lower. This gives rise to the asssumption, that most of the localization accuracy improvement originates from the YOLOv4 prediction.

Table 12: The absolute and relative improvement of the YOLOv4+CNN approach compared to the CNN only approach.

Improvement (absolute/relative)				
PL	**Port** **MAE** $\Gamma[m^2/s]$	**Starboard** **MAE** $\Gamma[m^2/s]$	**Port** **ADE** [m]	**Starboard** **ADE** [m]
Up	-9.68/-31.15%	-9.33/-28.09%	-43.64/-93.04%	-43.79/-93.77 %
Down	-4.88/-18.94%	-5.02/-15.52 %	-18.66/-85.24%	-19.88/-87.58 %

A significant difference between the approach combining YOLOv4 and CNN for prediction and the CNN-only approach is that it is independent of the plate line scenario and the vortex class in the case of vortex center localization, whereas in [10], they use a different CNN for every possible scenario, and also predict the localization coordinates separately. In the case of circulation strength prediction we can still see a discrepancy between different plate line scenarios and vortex classes. Especially the circulation strength prediction of starboard vortices without plate line usage has the highest MAE with 27.22 m^2/s.

Furthermore, the usage of YOLOv4 made us independent of the number of vortices in each scan. The CNN-only approach had to set unavailable targets for scans only containing one vortex or iterate over scans multiple times with more than two vortices. With the new approach, we can feed each scan through our pipeline independently of the number of vortices.

The hazard detection can also be compared by using the precision value. The precision of the CNN-only approach was 88.6% [10]. With the new approach, we could increase the precision by 7.51% to gain a precision of 96.11%.

9 Conclusion

9.1 Accomplishments

As it was one of the biggest issues of the convolutional neural network (CNN) approach using a complete LiDAR scan, the goal main goal of this thesis was to implement a pipeline for independent vortex analysis. Therefore, a pipeline of YOLOv4 bounding box prediction and CNNs to enhance the localization prediction and additionaly provide circulation strength predictions, was used.

To evaluate the fundamental behavior of YOLOv4, we first used proxy scans. We were able to show that YOLOv4 generalizes well on a different number of vortices contained in a LiDAR scan. This result made the processing pipeline independent of the number of vortices for further characterization. Furthermore, YOLOv4 classifies port and starboard vortices with a precision of 92% and a recall of 94%.

The bounding box prediction of YOLOv4 already gave an improvement on the localization accuracy. The mean absolute distance error (MADE) of vortex center prediction could be improved significantly to 6.29 m . This still left room for further enhancement. Thus, a CNN for vortex center prediction on individual vortices was employed.

Besides YOLOv4 drastically decreasing the localization error, the accuracy could be further enhanced with an additional CNN for localization. The final MADE is 3.08 m, approximately halving the MADE from YOLOv4.

Another wake vortex aspect we looked at was circulation strength. To predict the circulation strength of individual vortices, the same preprocessing routine and CNN architecture as for localization was used. An independent CNN was trained only for circulation strength prediction. As a benchmark value, perfectly cropped vortices were used to investigate whether a similar accuracy on cropped vortices based on YOLOv4 predictions can be achieved. The mean absolute error (MAE) could be improved by a minimum of 15.52 % for starboard vortices in combination without plate line usage and a maximum of 31.15 % for port vortices in combination with plate line usage. The final MAE is 22.97 m^2/s.

Furthermore, we could show that by using YOLOv4 before wake vortex localization, we became independent of the plate line scenario and vortex class, achieving similar accuracies in all cases despite having no different CNNs for each case.

9.2 Outlook

During the evaluation of YOLOv4, the assumption arose that it detects more vortices than the traditional RV method could catch. This assumption still needs to be verified. We need perfect labels to verify this assumption, which the RV method can not give. A data set created with LiDAR scan simulations, where the vortex center and circulation can be set precisely, could be used to validate this assumption. This data set must also be more sophisticated than the proxy scans to represent noise, atmospheric turbulence, and boundary layer effects.

Furthermore, one could enhance the YOLOv4 prediction with employing custom anchor boxes and trying different scan resolutions. Future studies might take a look on how to choose the prefect anchor boxes for the application of wake vortex detection. Including further LiDAR scans from different airports and additional aircraft types would benefit the networks' generalizability.

We saw that the first layers of the localization and circulation CNN behaved nearly the same. An improvement might be achieved by predicting circulation and vortex position in a single CNN, including a custom loss function to separately represent the localization and circulation strength predictions.

Furthermore, we inspected wake vortices as 2D objects when in reality, they can be seen as 4D objects. The idea is to incorporate at least the time axis into training and let the network learn some features like vortex decay and other vortex evolution mechanisms. One could achieve such by using recurrent neural networks like LSTM for training on sequences of LiDAR scans instead [43]. This could improve the accuracy and help to track vortices over runways.

Besides, the application to aircraft wake vortices, the treatment of wake vortices individually makes it possible to adjust to new applications. One might be the evaluation of wake vortices generated by wind turbines to optimize the placement of wind turbines in a wind farm.

References

[1] J. N. Hallock and F. Holzäpfel. “A review of recent wake vortex research for increasing airport capacity”. In: *Progress in Aerospace Sciences* 98 (2018), pp. 27–36. DOI: 10.1016/j.paerosci.2018.03.003.

[2] M. Hoogstraten, H. G. Visser, D. Hart, V. Treve, and F. Rooseleer. “Improved Understanding of En Route Wake-Vortex Encounters”. In: *Journal of Aircraft* 52.3 (2015), pp. 981–989. DOI: 10.2514/1.C032858.

[3] Bundesstelle für Flugunfalluntersuchung. *Bulletin Unfälle und Störungen beim Betrieb ziviler Luftfahrzeuge Januar 2017.* Braunschweig, 2017. URL: https://www.bfu-web.de/DE/Publikationen/Bulletins/2017/Bulletin2017-01.pdf?__blob=publicationFile.

[4] T. Gerz, F. Holzäpfel, and D. Darracq. “Commercial aircraft wake vortices”. In: *Progress in Aerospace Sciences* 38.3 (2002), pp. 181–208. DOI: 10.1016/S0376-0421(02)00004-0.

[5] EUROCONTROL. *Seven-Year Forecast 2021- 2027 - Main report.* 2021. URL: https://www.eurocontrol.int/publication/eurocontrol-forecast-update-2021-2027 (visited on 11/07/2020).

[6] N. L. Wartha, A. Stephan, G. Rotshteyn, and F. Holzäpfel. “Investigating Artificial Neural Networks for Detecting Aircraft Wake Vortices in Lidar Measurements”. In: *ODAS 2022 - 22nd Onera-DLR Aerospace Symposium.* June 2022, pp. 1–19.

[7] EUROCONTROL. *Challenges of growth 2018.* 2018. URL: https://www.eurocontrol.int/publication/challenges-growth-2018 (visited on 11/07/2020).

[8] J. Cheng, A. Hoff, J. Tittsworth, and W. A. Gallo. “The Development of Wake Turbulence Re-Categorization in the United States (Invited)”. In: *8th AIAA Atmospheric and Space Environments Conference.* DOI: 10.2514/6.2016-3434.

[9] F. Holzäpfel et al. “Mitigating Wake Turbulence Risk During Final Approach via Plate Lines”. In: *AIAA Journal* 59.11 (Nov. 2021), pp. 4626–4641. DOI: 10.2514/1.J060025.

[10] N. L. Wartha, A. Stephan, F. Holzäpfel, and G. Rotshteyn. “Characterizing aircraft wake vortex position and strength using LiDAR measurements processed with artificial neural networks”. In: *Opt. Express* 30.8 (Apr. 2022), pp. 13197–13225. DOI: 10.1364/OE.454525.

[11] J. Anderson. *Fundamentals of Aerodynamics.* 5th ed. New York: McGraw-Hill, 2011. ISBN: 978-0-07-339810-5.

[12] P. K. Kundu and I. M. Cohen. *Fluid Mechanics.* 4th ed. New York: Academic Press, 208. ISBN: 978-0-12-373735-9.

[13] A. Stephan. “Wake vortices of landing aircraft”. PhD thesis. Ludwig-Maximilians-Universität München, 2014.

[14] J. Katz and A. Plotkin. *Low-Speed Aerodynamics*. 2nd ed. Cambridge Aerospace Series. Cambridge University Press, 2001. ISBN: 9780511810329.

[15] J. Carlton. "Marine Propellers and Propulsion (Fourth Edition)". In: ed. by J. Carlton. 4th ed. Butterworth-Heinemann, 2019, pp. 141–175. ISBN: 978-0-08-100366-4.

[16] P. K. Kundu and I. M. Cohen. "Aerodynamics". In: *Fluid Mechanics (Second Edition)*. Second Edition. Boston: Academic Press, 2002, pp. 629–660. ISBN: 978-0-12-178251-1.

[17] F. Holzäpfel et al. "The Wake Vortex Prediction and Monitoring System WSVBS Part I: Design". In: *Air Traffic Control Quarterly* 17.4 (2009), pp. 301–322. DOI: 10.2514/atcq.17.4.301.

[18] A. Stephan, D. Rohlmann, F. Holzäpfel, and R. Rudnik. "Effects of Detailed Aircraft Geometry on Wake Vortex Dynamics During Landing". In: *Journal of Aircraft* 56.3 (2019), pp. 974–989. DOI: 10.2514/1.C034961.

[19] S. C. Crow. "Stability theory for a pair of trailing vortices". In: *AIAA journal* 8.12 (1970), pp. 2172–2179. DOI: 10.2514/3.6083.

[20] T. Leweke, S. Le Dizès, and C. H. Williamson. "Dynamics and Instabilities of Vortex Pairs". In: *Annual Review of Fluid Mechanics* 48.1 (2016), pp. 507–541. DOI: 10.1146/annurev-fluid-122414-034558.

[21] P. Lissaman, S. C. Crow, P. MacCready Jr, I. Tombach, E. Bate Jr, et al. *Aircraft vortex wake descent and decay under real atmospheric effects*. Tech. rep. United States. Federal Aviation Administration, 1973.

[22] T. Misaka, F. Holzäpfel, and T. Gerz. "Large-Eddy Simulation of Aircraft Wake Evolution from Roll-Up Until Vortex Decay". In: *AIAA Journal* 53.9 (2015), pp. 2646–2670. DOI: 10.2514/1.J053671.

[23] F. Holzäpfel. "Probabilistic Two-Phase Wake Vortex Decay and Transport Model". In: *Journal of Aircraft* 40.2 (2003), pp. 323–331. DOI: 10.2514/2.3096.

[24] S. Körner. "Multi-Model Ensemble Wake Vortex Prediction". PhD thesis. DLR, 2017.

[25] F. Holzäpfel, A. Stephan, and G. Rotshteyn. "Plate lines reduce lifetime of wake vortices during final approach to Vienna airport". In: *AIAA Scitech 2020 Forum*. 2020. DOI: 10.2514/6.2020-0050.

[26] A. Stephan, F. Holzäpfel, and T. Misaka. "Aircraft Wake-Vortex Decay in Ground Proximity—Physical Mechanisms and Artificial Enhancement". In: *Journal of Aircraft* 50.4 (2013), pp. 1250–1260. DOI: 10.2514/1.C032179.

[27] R. George and J. Yang. "A survey for methods of detecting aircraft vortices". In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 45004. American Society of Mechanical Engineers. 2012, pp. 41–50. DOI: 10.1115/DETC2012-70632.

[28] I. Smalikho, V. Banakh, F. Holzäpfel, and S. Rahm. “Method of radial velocities for the estimation of aircraft wake vortex parameters from data measured by coherent Doppler lidar”. In: *Opt. Express* 23.19 (Sept. 2015), A1194–A1207. DOI: 10.1364/OE.23.0A1194.

[29] C. Weitkamp. *Lidar: range-resolved optical remote sensing of the atmosphere.* Vol. 102. Springer Science & Business, 2006. ISBN: 9780387251011.

[30] N. Wildmann. *Wind and turbulence measurements with multiple Doppler wind lidars.* Innsbruck University, May 2019.

[31] N. N. Ahmad and F. Proctor. “Review of idealized aircraft wake vortex models”. In: *52nd Aerospace Sciences Meeting.* 2014. DOI: 10.2514/6.2014-0927.

[32] N. L. Wartha. “Wake Vortex Characterisation of Landing Aircraft using Artificial Neural Networks and LiDAR Measurements”. MA thesis. University of Glasgow, 2021.

[33] S. Schönhals, M. Steen, and P. Hecker. “Wake vortex prediction and detection utilising advanced fusion filter technologies”. In: *The Aeronautical Journal* 115.1166 (2011), pp. 221–228. DOI: 10.1017/S0001924000005674.

[34] F. Köpp, S. Rahm, and I. Smalikho. “Characterization of Aircraft Wake Vortices by 2-μm Pulsed Doppler Lidar”. In: *Journal of Atmospheric and Oceanic Technology* 21.2 (2004), pp. 194–206. DOI: 10.1175/1520-0426(2004)021<0194:COAWVB>2.0.CO;2.

[35] F. Köpp et al. “Comparison of Wake-Vortex Parameters Measured by Pulsed and Continuous-Wave Lidars”. In: *Journal of Aircraft* 42.4 (2005), pp. 916–923. DOI: 10.2514/1.8177.

[36] I. Smalikho, V. Banakh, F. Holzäpfel, and S. Rahm. “Method of radial velocities for the estimation of aircraft wake vortex parameters from data measured by coherent Doppler lidar”. In: *Opt. Express* 23.19 (Sept. 2015), A1194–A1207. DOI: 10.1364/OE.23.0A1194.

[37] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. “Machine Learning for Fluid Mechanics”. In: *Annual Review of Fluid Mechanics* 52.1 (2020), pp. 477–508. DOI: 10.1146/annurev-fluid-010719-060214.

[38] European Union Aviation Safety Agency. *EASA Concept Paper: First usable guidance for Level 1 machine learning applications Issue 01.* Tech. rep. 2022. URL: https://www.easa.europa.eu/en/easa-concept-paper-first-usable-guidance-level-1-machine-learning-applications-proposed-issue-01pdf.

[39] W. Pan, Z. Wu, and X. Zhang. “Identification of Aircraft Wake Vortex Based on SVM”. In: *Mathematical Problems in Engineering* 2020 (May 2020). DOI: 10.1155/2020/9314164.

[40] W. Pan, H. Yin, Y. Leng, and X. Zhang. "Recognition of Aircraft Wake Vortex Based on Random Forest". In: *IEEE Access* 10 (2022), pp. 8916–8923. DOI: 10.1109/ACCESS.2022.3141595.

[41] W. Pan, Y. Leng, H. Yin, and X. Zhang. "Identification of Aircraft Wake Vortex Based on VGGNet". In: *Wireless Communications and Mobile Computing* 2022 (June 2022). DOI: 10.1155/2022/1487854.

[42] Y. Ai, Y. Wang, W. Pan, and D. Wu. "A Deep Learning Framework Based on Multisensor Fusion Information to Identify the Airplane Wake Vortex". In: *Journal of Sensors* 2021 (Nov. 2021). DOI: 10.1155/2021/4819254.

[43] W.-J. Pan, Y.-F. Leng, T.-Y. Wu, Y.-X. Xu, and X.-L. Zhang. "Conv-Wake: A Lightweight Framework for Aircraft Wake Recognition". In: *Journal of Sensors* 2022 (2022). DOI: 10.1155/2022/3050507.

[44] N. Baranov and B. Resnick. "Wake vortex detection by convolutional neural networks". In: *European Journal of Electrical Engineering and Computer Science (EEACS)* 3 (2021), pp. 92–97.

[45] P. Weijun, D. Yingjie, Z. Qiang, T. Jiahao, and Z. Jun. "Deep Learning for Aircraft Wake Vortex Identification". In: *IOP Conference Series: Materials Science and Engineering* 685.1 (Nov. 2019). DOI: 10.1088/1757-899x/685/1/012015.

[46] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[47] K. Fukushima and S. Miyake. "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition". In: *Competition and Cooperation in Neural Nets*. Springer, 1982, pp. 267–285. DOI: 10.2514/6.2016-3434.

[48] Y. LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[49] I., Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[50] C. F. Higham and D. J. Higham. *Deep Learning: An Introduction for Applied Mathematicians*. 2018. DOI: 10.48550/ARXIV.1801.05894.

[51] URL: https://github.com/IzaakWN/CodeSnippets/tree/master/LaTeX/TikZ (visited on 11/07/2022).

[52] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. "Activation functions: Comparison of trends in practice and research for deep learning". In: *arXiv preprint arXiv:1811.03378* (2018). DOI: 10.48550/ARXIV.1811.03378.

[53] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[54] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. Atlanta, Georgia, USA. 2013, p. 3.

[55] D. Misra. “Mish: A Self Regularized Non-Monotonic Neural Activation Function”. In: *CoRR* abs/1908.08681 (2019). DOI: 10.48550/arXiv.1908.08681.

[56] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. DOI: 10.1007/BF02551274.

[57] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural Networks* 6.6 (1993), pp. 861–867. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80131-5.

[58] M. Everingham, L. Van Gool, J. Winn, and A. Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88 (2010). DOI: 10.1007/s11263-009-0275-4.

[59] F. Chollet. *Deep learning with Python*. Birmingham: McGraw-Hill, 2017. ISBN: 978-1-617-29443-3.

[60] L. Bottou. “Stochastic gradient descent tricks”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436. DOI: 10.1007/978-3-642-35289-8_25.

[61] M. Li, T. Zhang, Y. Chen, and A. J. Smola. “Efficient Mini-Batch Training for Stochastic Optimization”. In: KDD ’14. New York, New York, USA: Association for Computing Machinery, 2014. ISBN: 9781450329569.

[62] C. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, Jan. 2006. ISBN: 978-0- 387-31073-2.

[63] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980.

[64] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[65] V. Dumoulin and F. Visin. *A guide to convolution arithmetic for deep learning*. 2016. DOI: 10.48550/ARXIV.1603.07285.

[66] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. DOI: 10.48550/ARXIV.1502.03167.

[67] Y. Boureau, J. Ponce, and Y. LeCun. “A theoretical analysis of feature pooling in visual recognition”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 111–118. ISBN: 9781605589077.

[68] I. Arel, D. C. Rose, and T. P. Karnowski. “Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]”. In: *IEEE Computational Intelligence Magazine* 5.4 (2010), pp. 13–18. DOI: 10.1109/MCI.2010.938364.

[69] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. “Object detection with deep learning: A review”. In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.

[70] L. Du, R. Zhang, and X. Wang. “Overview of two-stage object detection algorithms”. In: *Journal of Physics: Conference Series* 1544 (May 2020). DOI: 10.1088/1742-6596/1544/1/012033.

[71] R. Girshick, J. Donahue, T. Darrell, and J. Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[72] R. Girshick. “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[73] S. Ren, K. He, R. Girshick, and J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[74] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[75] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[76] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[77] J. Redmon and A. Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[78] J. Redmon and A. Farhadi. *YOLOv3: An Incremental Improvement*. 2018. DOI: 10.48550/ARXIV.1804.02767.

[79] A. Bochkovskiy, C. Wang, and H. M. Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: *CoRR* abs/2004.10934 (2020). DOI: 10.48550/ARXIV.2004.10934.

[80] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal Loss for Dense Object Detection”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[81] W. Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0.

[82] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90. DOI: 10.1145/3065386.

[83] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. *Detnet: A backbone network for object detection*. 2018. DOI: 10.48550/ARXIV.1804.06215.

[84] S. Bouraya and A. Belangour. “Deep Learning based Neck Models for Object Detection: A Review and a Benchmarking Study”. In: *International Journal of Advanced Computer Science and Applications* 12.11 (2021). DOI: 10.14569/IJACSA.2021.0121119.

[85] I. N. Smalikho and V. A. Banakh. “Estimation of aircraft wake vortex parameters from data measured with a 1.5-μm coherent Doppler lidar”. In: *Opt. Lett.* 40.14 (July 2015), pp. 3408–3411. DOI: 10.1364/OL.40.003408.

[86] F. Holzäpfel et al. “Strategies for Circulation Evaluation of Aircraft Wake Vortices Measured by Lidar”. In: *Journal of Atmospheric and Oceanic Technology* 20.8 (2003), pp. 1183–1195. DOI: 10.1175/1520-0426(2003)020<1183:SFCEOA>2.0.CO;2.

[87] A. Bochkovskiy. *Yolo v4, v3 and v2 for Windows and Linux*. 2022. URL: https://github.com/AlexeyAB/darknet (visited on 10/19/2022).

[88] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh. “CSPNet: A New Backbone that can Enhance Learning Capability of CNN”. In: (2020), pp. 1571–1580. DOI: 10.1109/CVPRW50498.2020.00203.

[89] K. He, X. Zhang, S. Ren, and J. Sun. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

[90] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. “Path Aggregation Network for Instance Segmentation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8759–8768. DOI: 10.1109/CVPR.2018.00913.

[91] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. “Distance-IoU loss: Faster and better learning for bounding box regression”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 12993–13000. DOI: 10.1609/aaai.v34i07.6999.

[92] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. “Soft-NMS — Improving Object Detection with One Line of Code”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5562–5570. DOI: 10.1109/ICCV.2017.593.

[93] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022. URL: https://smlbook.org.

[94] F. Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[95] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[96] S. Charette. *Darknet FAQ*. 2022. URL: https://www.ccoderun.ca/programming/2020-09-25_Darknet_FAQ/ (visited on 10/19/2022).

[97] K. Bredies and D. Lorenz. *Mathematische Bildverarbeitung*. Vol. 1. Springer, 2011. ISBN: 978-3-8348-1037-3.

[98] N. L. Wartha and T. Bölle. *Personal communication*. May 2022.

[99] G. Rotshteyn. *Personal communication*. May 2022.

[100] T.-Y. Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48.

[101] F. Holzäpfel, N. L. Wartha, A. Stephan, and G. Rotshteyn. *Wake Vortex Group Meeting*. July 2022.

Acknowledgments

During this thesis, I received a lot of input and support. First, I would like to thank my supervisors, Niklas Wartha, Prof. Dr. Daniel Ruprecht, and Dr. Sebastian Götschel. Thank you, Niklas, for our countless discussions, being open to my concerns, and constantly pushing me to achieve better results. Thank you, Daniel and Sebastian, for always having an open ear to my questions. You always gave me the feeling of asking the right things leading in the right direction in the monthly Zoom sessions. I want to express my gratitude to the entire Wake Vortex Group, Dr. Anton Stephan, Dr. Frank Holzäpfel, Grigory Rotshteyn and Moritz Spraul for always showing me your interest in my research and helping me to understand the engineering point of view on wake vortices. Furthermore, I would like to thank the whole Transport Meteorology department for allowing me to be part of the team. I really enjoyed the daily coffee and lunch breaks together.

I would like to thank all the young researchers at the Institute of Atmospheric Physics, for letting me take part in group activities, despite only being there for my master thesis. I would especially like to thank Niklas, Moritz, and Kianusch for the almost weekly bouldering sessions we had.

Last but not least, I would like to express my personal gartitude to Käthe and my family for the continuous mental and emotional support through the hard times, giving me the energy needed to finish this work.

This work was performed on the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research. As well as the image transformation on CARA and the YOLOv4 training on the TUHH Cluster.