Automated Methods for the Detection and Creation of Frontal Systems in Atmospheric Data

Stefan Niebler

September 11, 2024

Automated Methods for the Detection and Creation of Frontal Systems in Atmospheric Data

Dissertation
for the attainment of the degree
"Doctor of Natural Sciences"
at the Department of Physics, Mathematics, and Computer Science
of Johannes Gutenberg University
in Mainz

Stefan Niebler

born in Bayreuth, Germany.
Mainz, September 11, 2024

1. Reviewer	Prof. Dr. Bertil Schmidt
2. Reviewer	Prof. Dr. Peter Spichtinger
Date of oral examination	January 30, 2025

Abstract

Synoptic scale weather fronts are ubiquitous elements of extra-tropical weather. Their connection to severe weather conditions such as extreme precipitation, cyclones or thunderstorms highlights the importance of being able to reliably detect these structures to conduct analyses surrounding their effect within the atmosphere. Nonetheless, it is still common to draw fronts via manual analyses by weather services, inevitably introducing biases or being restricted to the corresponding weather service's analysis region. Common numerical, objective methods on the other hand have been developed for a lower resolution than the currently available datasets. While adjustments to make these algorithms applicable to high resolution datasets exist, fine details may be lost in the process. Furthermore, these traditional methods all only provide frontal information at a single pressure or model level, reducing them to a line in a $2D$ slice, neglecting their overall three-dimensional structure. Implications surrounding the three-dimensional shape are therefore still scarce, as objective methods for the creation of their three-dimensional structure are very limited. Due to their connection to severe weather conditions, which can lead to serious damages, it is important to provide methods that are able to both localize these fronts on current datasets and being able to recreate their three-dimensional structure.

In this work, we present a convolutional neural network, which is capable of locating and classifying atmospheric fronts on the current ERA5 dataset. Our method outperforms a common numerical approach when detecting fronts within both the North American continent and Western Europe. These methods are evaluated with respect to manual front analyses of two weather services, however, our model may also be applied globally, where we could show a high correlation of our detected fronts and extreme precipitation.

Additionally, we propose a novel algorithm for the creation of three-dimensional fronts outside of case studies, being - to the best of our knowledge - the first method to allow for statistical evaluation of frontal shape in three dimensions. We further provide an improved implementation of this algorithm which utilizes modern graphics processing units (GPU) to drastically reduce compute time. We show that our implementation can create three-dimensional fronts for our dataset region within the Northern Atlantic in less than 500 seconds, which is vastly faster than

our version, that only utilizes central processing units (CPU). The latter took several hours on multiple nodes. Further, we show that both of our implementations exhibit structural features commonly associated with fronts, such as maximum baroclinity and steep temperature gradients. Similar to our deep learning model, we are able to provide information for all commonly used types of fronts: *warm*, *cold*, *occluded* and *stationary* fronts.

Contents

1 Introduction

1.1 Motivation and problem statement

Atmospheric science as a topic has gained more interest in the public discourse over the last years. Especially with regard to climate change, it has become more important to gain a better understanding of atmospheric processes and their respective implications. In the same vein, researching weather, a more short term view on the atmospheric condition, is of importance. We can already observe an increase in extreme weather situations or natural hazards linked to climate change [27]. Due to their severe impact on our lives, researching extreme events and accompanying atmospheric phenomena is important, since it allows us to get a better understanding of their causes. Eventually, this will aid in the development of better short-term or long-term predictions and potential countermeasures. Apart from these extreme situations, providing better, more accurate weather predictions can provide meaningful information which can be leveraged for short term planning, for example for scheduling flights or predicting electricity generation.

Weather fronts are such accompanying events that are often correlated to extreme situations, such as cyclones, thunderstorms or extreme precipitation. These structures exist around the globe. Although, they are mainly being researched within the midlatitudes, where their impact is most dominant. Since a front is defined as the boundary layer between two differing air masses, they are often accompanied by adiabatic processes, leading to temperature swings and the forming of clouds. They are also closely related to the formation of cyclones, as represented by common cyclone models, such as Shapiro-Keyser [71] or the Norwegian [9, 18] cyclone model (see also [66]). As fronts have a certain longevity spanning multiple hours, potentially even days, they can be a good indicator for the location of future extreme weather occurrences.

Over the past years the amount of available data in the natural sciences has grown [23] and will continue to grow even further with increasing model capabilities, the inclusion of additional resources, and higher resolutions [@7, 2]. Evaluation and processing of these datasets is faced with an ever-increasing need for computational power. Single-threaded performance of CPUs, however, did not increase at the same

rate as the amount of data. To efficiently process the ever-increasing amount of data, parallelization and in return specialized hardware such as multi-core CPUs or GPUs is necessary. However, to exploit the capabilities of these parallel, possibly multi-node systems, it is necessary to implement and design specialized algorithms which are able to fully utilize the hardware. Apart from numerical algorithms accelerated on GPUs, novel machine learning techniques such as deep neural networks emerged, leveraging the availability of the large amount of observational data. These models usually rely on parallel hardware to efficiently train their respective models. This demand for new efficient hardware and algorithms has also been acknowledged by various weather services, which usually have their own super computer hardware [@13, @6]. Even though many algorithms are still CPU based, there are efforts to utilize GPUs for specialized tasks [@1]. Furthermore, the European Centre for Medium-Range Weather Forecasts (ECMWF) also adapts machine learning techniques, for example in form of their novel AIFS model [@2]. We can expect that the amount of GPU based numerical algorithms as well as machine learning and deep learning based models used in weather forecasting and related fields will increase. This further enhances the importance of researching suitable methods.

Common conventional algorithms for the detection of weather fronts have several limitations. First, they have been developed for lower resolved atmospheric grids. Approaches, such as smoothing of higher resolution data, are applied to allow for the applicability of these algorithms to modern higher resolved datasets. However, this smoothing destroys fine scale information that may be beneficial for the overall capability of both detecting and researching weather fronts. Another current limitation is, that many methods only use data from a single vertical level as an input, even though the vertical neighborhood of a frontal surface may also hold important information. Due to this, the common representation and detection of frontal data is limited to single level fronts, ignoring the actual three-dimensional shape that such fronts possess.

Within this work, we present novel methods to create and classify frontal structures at the high $0.25°$ resolution of ERA5. We show how this tool can be applied to research correlations between extreme precipitation and fronts at this resolution. Furthermore, we propose an algorithm to create three-dimensional representations of fronts, which allows us to provide statistical evaluations of the latter. To be able to process all this data in a timely manner, efficient, ideally parallel algorithms are necessary. As model data can commonly be obtained as gridded datasets, GPUs are an ideal choice to provide a significant acceleration of algorithms that process the vast amounts of atmospheric data. By utilizing GPU hardware, we are able to create

three-dimensional fronts for every hourly timestamp of 2016 in less than 500 seconds on a single GPU, instead of several hours on multiple CPUs.

1.2 Thesis structure

In this thesis, we will present a deep learning based method for the detection of surface $2D$ fronts and two methods for locating three-dimensional fronts based on given surface fronts.

In Chapter 3 we provide general information about weather fronts, deep learning, as well as the Compute Unified Device Architecture (CUDA) programming model.

The main content of the thesis is divided into two parts. In Chapters 4 to 8 our deep learning based method for the detection of two-dimensional surface fronts is presented. In the second part, consisting of Chapters 9 to 14, we present our novel algorithm to create three-dimensional weather fronts out of surface fronts. Finally, we provide a conclusion and possible future work in Chapters 15 and 16.

1.3 Publications

This thesis is based on the following publications and uses content directly taken from these papers.

Our deep learning based method for the detection of two-dimensional surface fronts was previously published in Weather and Climate Dynamics (WCD) [44]. The network provides a global detection and classification of surface fronts at a $0.25°$ resolution latitude-longitude grid. Our method can be directly applied to this grid without the need for previous smoothing.

We developed an algorithm to extend the surface fronts by our neural network to a three-dimensional representation, being - for the best of our knowledge - the first to enable statistical evaluations of three-dimensional fronts. A Python implementation of this method was published and presented at ICCS 2023 [48]. This algorithm was later adapted to run on GPUs, while simultaneously providing several improvements on the overall pipeline and published at PASC 24 [47]. The GPU implementation was able to drastically improve performance of the algorithm, allowing to process each time step of 2016 in only a few minutes on a single GPU, whereas the first implementation took several hours utilizing multiple CPU nodes.

Automated detection and classification of synoptic-scale fronts from atmospheric data grids
Stefan Niebler, Annette Miltenberger, Bertil Schmidt, Peter Spichtinger
Weather and Climate Dynamics (WCD), Volume 3, Issue 1, 2022, Pages 113-137
http://doi.org/10.5194/wcd-3-113-2022

Automated Identification and Location of Three Dimensional Atmospheric Frontal Systems
Stefan Niebler, Bertil Schmidt, Holger Tost, Peter Spichtinger
First published in Lecture Notes in Computer Science (ICCS 2023), Volume 14074, 2023, Pages 3-17 by Springer Nature
https://doi.org/10.1007/978-3-031-36021-3_1
Reproduced with permission from Springer Nature

Scalable GPU-Enabled Creation of Three Dimensional Weather Fronts
Stefan Niebler, Bertil Schmidt, Peter Spichtinger, Holger Tost
In: Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '24), 2024, Association for Computing Machinery
https://doi.org/10.1145/3659914.3659916

I developed, implemented and evaluated the software presented in the papers. Furthermore, I wrote several parts of the manuscripts.

Related work

2

Objective weather front analysis and detection has emerged during the second half of the 20^{th} century. Since then, several implementations and methods for the detection of weather fronts have been developed (Table 2.1). In 1965 Renard and Clarke [58] presented their numerical method to detect fronts within weather maps where they introduced a method later described as the thermal front parameter (TFP) (Eq. 2.1).

$$TFP = -\nabla \left|\nabla\theta\right| \cdot \frac{\nabla\theta}{\left|\nabla\theta\right|} \tag{2.1}$$

This formula describes the derivative of the thermal gradient in the direction of the thermal gradient at a given evaluation point. It is applied to a single height or pressure level and does not take temperature changes across levels into account. This method searches for a characteristic feature of fronts, where a steep temperature change across the front is expected. It can be used to either locate the center or the leading edge of a front by locating the front at the zero pass of the TFP or the gradient of the TFP, respectively [64, 29].

Another thermal gradient based method, building from the thermal front parameter, was later proposed by Hewson [20] who used this approach to automatically detect fronts without human intervention in gridded atmospheric datasets. Hewson locates fronts at the zero contour of $\frac{\delta\nabla|\nabla\tau|}{\delta\hat{s}}$, where τ denotes a thermal quantity and $\hat{s} = \pm\nabla\left|\nabla\tau\right| / \left|\nabla\left|\nabla\tau\right|\right|$ [20, 3]. For a one-dimensional front, referred to as a type I front by Hewson [20], this formula translates to the third derivative of τ [63]. This *localization criterion* should be met at the leading edge of a front. This criterion is however not sufficient to determine fronts. Other pixel, which are not part of a front, may fulfill the condition as well and, thus, need to be removed. For this, Hewson [20] apply two additional filters, based on other features connected to fronts. The first filter, removes all candidates, where the TFP does not exceed a filtering threshold. Essentially, points with a low TFP value are not considered fronts and should therefore not be further considered. The second filter masks all points, where the thermal gradient in the adjacent baroclinic zone does not exceed a given threshold, as a notable change in the thermal gradient across the frontal surface is expected. Berry et al. [4] reimplemented this method and recently Sansom and Catto [63] proposed an enhancement of the latter implementation. According

to Sansom and Catto [63], Berry et al. [4] first applied the TFP based mask and then searched for the *localization criterion* near possible candidate regions. This procedure could lead to cases, where a front was not correctly represented as the filtered regions were too narrow and possible frontal points were not identified. Therefore, Sansom and Catto [63] first identify the zero passes of the localization criterion and join them into lines with a contouring algorithm. Then the filter masks are applied at the interpolated positions to filter invalid fronts. This procedure was also originally proposed by Hewson [20]. They found that this order of operations generally increased the amount of detected fronts in gridded datasets. Overall, methods utilizing the TFP have been used by various studies to create and evaluate climatologies of global or regional frontal distribution [29, 4].

There is an ongoing discussion which thermal quantity is best suited for the detection of fronts (see e.g. [75]). Generally, numerical methods, which use the TFP or similar methods to detect fronts, do not rely on the temperature T but rather a derived thermal quantity. Renard and Clarke [58] use the potential temperature θ instead of T as a thermal field. The former corresponds to the temperature that a dry air parcel would have, if adiabatically brought to a certain pressure level, usually $1000\ \mathrm{hPa}$. However, this quantity is not conserved during moist-adiabatic processes, which generally occur near fronts, as, for example, clouds may form when moist air is lifted. Therefore, Hewson [20] and Berry et al. [4] instead use the wet-bulb potential temperature θ_w. Jenkner et al. [29] as well as Schemm et al. [64] use the equivalent potential temperature θ_e, instead. θ_e is conserved during reversible phase transitions in adiabatic processes, whereas θ_w is conserved for reversible pseudo-adiabatic processes. Therefore, they are commonly used for investigations of fronts.

Similarly to the choice of thermal field, the choice of altitude varies between studies. The most common choice is the $850\ \mathrm{hPa}$ pressure level. Jenkner et al. [29] differ from this by using the $700\ \mathrm{hPa}$ pressure level. Hewson [20] propose to use $900\ \mathrm{hPa}$ for surface fronts, although Berry et al. [4] and later Sansom and Catto [63] also apply this method at the $850\ \mathrm{hPa}$ pressure level.

Other numerical methods using non-thermal quantities exist. Simmonds et al. [73] proposed a method that investigates the Eulerian time rate of change of wind direction and speed in the lower troposphere to detect fronts. Schemm et al. [64] later compared this method against a TFP based method. While they found agreement between both methods, they also identified regional differences between both methods.

Finally, Névir [43] defines the dynamical state index (DSI), which is a parameter that indicates deviations from an adiabatic and stationary base state of an atmospheric flow field. It is used by Mack et al. [36] to localize fronts. This method is comparably rather new and has not been subject to extensive testing to the best of our knowledge. However, the results presented by the authors are promising.

In the next paragraph, we will shortly describe some common drawbacks of the numerical TFP based methods and how these problems are handled by our deep learning approach.

A general drawback of the mentioned TFP based methods is that they were originally developed for coarser datasets and cannot easily be applied to high resolution data. While tuning the parameter to improve results on higher resolution data is a possibility, such tuning is expensive and will not always be able to provide the desired results. Another common approach is to smooth the input dataset in a preprocessing step [3, 63]. However, by doing so, it reduces the benefits of using a high resolution dataset in the first place. Our proposed network on the other hand is directly trained on ERA5 and is therefore capable of providing output directly on the high resolution dataset. Further, as we do not need to handcraft the desired features of a front, we can easily use a broader spectrum of input variables and pressure levels. In comparison, the numerical methods generally locate the fronts based on a single atmospheric quantity at a single pressure level. With regard to the choice of thermal quantity, we provide the temperature T, pressure and specific humidity, which could be used to approximate any of the derived thermal quantities used by numerical approaches, if necessary. Potentially, the neural network is even able to learn a latent thermal representation, which better represents frontal features, instead. Since we include the horizontal and vertical wind, other, non-thermal features may be inferred by the network as well. All of this enables our network to use more of the available information for the classification and detection of weather fronts. Furthermore, one could theoretically use sensitivity studies to further investigate the importance of different atmospheric variables on the expression of frontal structures.

Due to the increased popularity and possibilities introduced by machine learning and especially deep learning, several other learned frontal detection methods have been developed as well. Biard and Kunkel [7] or Lagerquist et al. [32] used the MERRA-2 and NARR datasets, respectively, to detect fronts over the United States. They used labels provided by the National Weather Service (NWS) as ground truth. Both methods found that the results of the network were relatively wide compared to the lines provided as labels by the weather services. Lagerquist et al. [32] used skeletonizing as post-processing to thin the identified fronts to be able to output

thin lines. Another deep learning based method which used a loss based on the intersection over union was proposed by Matsuoka et al. [38]. They used frontal data located over Japan to analyze the Mei-Yu front. A different machine learning approach was introduced by Bochenek et al. [10], who used a random forest to detect weather fronts over Central Europe using data from the Deutscher Wetterdienst (DWD). They utilize the ERA5 reanalysis data as the atmospheric dataset, which we also utilize for our work. In contrast to the other deep learning based methods, our method utilizes frontal analyses of multiple weather services, namely the DWD and NWS. We are therefore able to train our model on different regions, which the other methods have not done. Furthermore, this allows us to evaluate how training on different regions affects the skill of the model.

We created an adjusted loss, based on the intersection over union, which allowed our network to provide the detected fronts as lines rather than areas. Differently to Lagerquist et al. [32], we are therefore able to omit heuristic post-processing methods, to obtain frontal lines.

These previously mentioned methods are restricted to providing the frontal location at a single vertical level. Regarding the three-dimensional shape of atmospheric fronts - as far as we know - only a single method has been previously created, which allows the user to investigate three-dimensional fronts [30, 3]. To detect fronts in a three-dimensional grid, they use Hewson's method. The method is applicable to various datasets of different resolution, although, similar to many two-dimensional front detection algorithms, smoothing of high resolution data is performed. Beckert et al. [3] also show, that their thresholds need to be adjusted according to the chosen smoothing length scale. The method is presented as a tool for the investigation of case studies. Further, it allows the user to statistically evaluate the properties of individual fronts. They use filtering, with user defined thresholds, that can be adjusted as necessary. The authors present investigations of individual case studies, however, they do not provide insights regarding a statistical evaluation of three-dimensional fronts in a climatological sense, e.g. aggregated results over large datasets.

We therefore developed a novel method, with the ability to create three-dimensional frontal structures out of an atmospheric grid and surface front positions. Our method is intended for the statistical evaluation of three-dimensional fronts over larger datasets. By using results of two-dimensional front detection methods as a prior, we already have a good estimate of potential front locations within our dataset. This allows us to omit filtering using thermal thresholds, as potential fronts are already identified. Further, we infer the position of a front using a local optimization

approach rather than a localization criterion. For this, we utilize the horizontal gradient information of a θ_e thermal field. This method was initially implemented as a single process Python program and later improved to run on GPUs using CUDA, which allows us to create three-dimensional fronts over the Northern Atlantic in less than 10 minutes for a complete year.

2D	
Author	Criterion
Jenkner [29]	TFP
Schemm [64]	TFP & Wind
Simmonds [73]	Wind
Hewson [20]	Hewson (TFP)
Catto [63]	Hewson (TFP)
Mack [36]	DSI
Lagerquist [32]	DL
Biard [7]	DL
Matsuoka [38]	DL
Bocheneck [10]	Random forest
3D	
Kern, Beckert [30, 3]	Hewson (TFP)

Tab. 2.1.: Selection of different front detection methods and the criterion they are based on. Hewson's method implicitly uses TFP through its filtering. Deep Learning (DL) and Random Forest based methods use machine learning to deduce the criterion.

3 Background

3.1 Frontal systems

3.1.1 General background

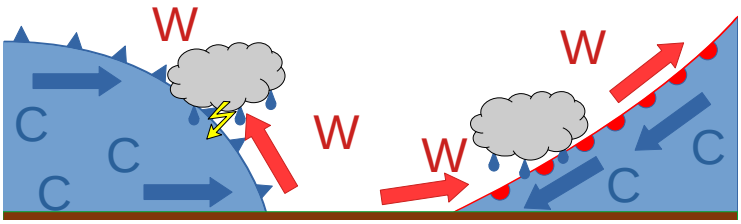
Atmospheric fronts are defined as narrow transition zones between air masses, differing in density and/or temperature [see e.g. 74]. They play an important role in synoptic weather systems within the midlatitudes. These fronts are considered synoptic scale phenomena, as each can span a length of several hundred kilometers. Synoptic scale meteorology usually works on large scale observations spanning several hundreds or thousands of kilometers. For instance, the National Oceanic and Atmospheric Association (NOAA) refers to the synoptic scale as the scale at which weather patterns between $1000\ \mathrm{km}$ to $2500\ \mathrm{km}$ are observed [@12]. Unlike the large extent of a front, the size across its frontal surface is comparably narrow, spanning only a few dozen kilometers up to a few hundreds of kilometers [@4].

The frontal life cycle starts at the frontogenesis stage, at which the thermal gradient between air masses intensifies. Different models account advection caused by deformation to lead to the creation of a front [25, 77]. Recent research is also taking into account other influences, such as sea surface temperature fronts [57, 52, 51]. The frontal lifecycle then ends at the frontolysis, at which point the front is dissipating. During its lifetime, a front may move or be (quasi-)stationary. It further may collide with other fronts, potentially merging into a different type of front. We will explain this more in the next section. Typical characteristics associated with a front during its lifetime are cloudiness, temperature swings or wind shifts, when the front passes through a region [42, 64, 73]. They can, however, also be associated to more significant weather such as high gust speeds, thunderstorms, extreme precipitation or cyclones [16, 14, 37]. While several objective front detection algorithms have been developed, there is not a single clear physical definition of what makes a front. As a result, most frontal detection algorithms rely on common features associated with fronts, choosing appropriate thresholds [75].

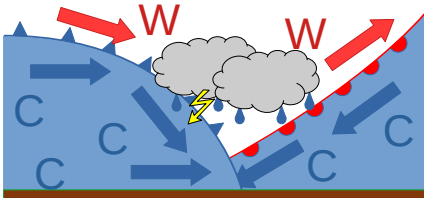
3.1.2 Classification and characteristics

There exist several types of fronts. The most common classifications are *warm*, *cold*, *occluded* or *(quasi-)stationary*. The mentioned classifications are exclusive, although classification may not always be clear and a front may change its classification during its lifetime. A *warm* front occurs, when a warmer air mass catches up to a cold air mass. The warm air mass has a lower density than the cold air mass and therefore, once the warm air reaches the cold air mass it begins to ascend, while the cold air stays at lower levels. The warm air slides over the cold air mass along an ideally shallow inclination path. Due to the warm advection, clouds may form and precipitation possibly occurs ahead of the surface level front. At the surface level, a warm front typically manifests in a sudden rise of temperature, once the front has passed. Additionally, a change in wind direction may be observed. Warm fronts are typically depicted as red lines with half circles indicating the direction of movement of the front. Differently, a *cold* front occurs when the cold air mass moves onto the warm air mass instead. Due to the difference in density, the cold air does not ascend over the warm air but rather forces the warm air upwards, such that the cold air can move below it. Again we observe warm air advection, which lets clouds or convection manifest near the front, thus potentially leading to storms and precipitation. In contrast to the warm front, the temperature decreases with the passing of the front. Similar to the *warm* fronts, a change in the wind direction may occur. A *cold* front is typically depicted as a blue line with triangles, indicating the direction of travel of the front. There exist subtypes of *warm* and *cold* fronts, called high altitude *warm* or *cold* fronts. As the name indicates, these are *warm* or *cold* fronts, however, the temperature change commonly associated with the passing of a front only occurs at higher altitude or lower pressure levels than it would for a general *warm* or *cold* front. These are usually depicted in the same fashion as *warm* or *cold* fronts, although the shape indicating the frontal motion is hollow instead. For the work presented in this thesis, we simply considered these front to be regular *warm* or *cold* fronts.

Another important type of fronts are *occluded* fronts or *occlusions*. Differently to both *warm* and *cold* fronts, which form at the boundary of two differing air masses, an *occlusion* is defined as the overlapping of a *warm* and *cold* front. Generally, such an *occluded* fronts occurs, when a faster moving *cold* front catches up to a leading *warm* front. This usually occurs near cyclones or low pressure zones. Both fronts rotate around the low pressure center. However, the *cold* front rotates faster and therefore catches up to the *warm* front. The warm air sector between these two fronts is lifted, and the two opposing cold air masses begin to overlap at the lower altitudes. Due to



(a) Sketch of a cold (left) and a warm (right) front



(b) Sketch of an occluded front

the warm advection, once again, we often see cloudiness and storms accompanying *occlusions*.

As the cold air masses may also differ in temperature and density, we may additionally observe one cold air mass to be advected as well. If the cold air mass of the *cold* front is cooler than the cold air mass of the *warm* front, the latter will be advected, and the *occlusion* is referred to as a cold *occlusion*. In the opposite case, it is called a warm *occlusion* instead. Due to the strong advection, *occlusions* are often accompanied by storms and rain. Furthermore, as they oftentimes form at low pressure zones and exhibit a strong rotational force, they are often associated with cyclones.

As a fourth type, there are *(quasi-)stationary* fronts. These fronts are mainly characterized as fronts, whose movement speed is below a threshold of $5\mathrm{kn}$, thus being nearly stationary. As a result, the thermal advection is also comparatively low, and the other effects are less dominantly expressed.

3.2 Deep learning

Deep learning (DL) is a machine learning technique that found increasing attention in recent years. While often used synonymous, deep learning is a subset of artificial intelligence (AI) and machine learning (ML). Deep learning models have shown impressive results, exceeding state of the art in natural sciences, natural language processing (NLP), computer vision (CV) and other domains. For example, modern deep learning based models are capable of interpreting human-readable text and

respond with their own textual response [50]. Other models can take such a textual prompt and generate high fidelity images in various styles [60, 5]. Within the atmospheric sciences, recent models have proven to be capable of providing multi hour global scale forecasts, exceeding the skill of numerical weather prediction algorithms [6, 33].

These networks consist of several million parameters, which are usually trained on large datasets. During training, the parameters of a neural network are adjusted in a way to optimize the overall skill of the neural network to solve a given task. Due to their vast amount of parameters, these models are able to automatically learn complex correlations and features within the data during training, which can be used to solve the given task. These implicit features oftentimes exceed human crafted features in terms of complexity and level of detail, which is a reason why neural networks are capable of outperforming other non machine learning based methods.

Despite their good results, there are certain drawbacks when using deep learning based models. For once, deep learning models lack good explainability. While deep learning models are skillful in solving certain tasks, one currently cannot answer why the network provided a certain result. As a result, the potential of deep learning, to provide new insights, with respect to why a certain phenomenon such as a weather front was detected, is limited. Furthermore, to achieve good results, modern deep learning architectures rely on huge amounts of data necessary for training, validation and testing. With an increasing complexity and size of a model, the necessary compute resources increase as well. For example, training GraphCast took approximately four weeks on 32 Cloud TPU v4 devices [33]. Nevertheless, in the given case, the high resource usage during training can be justified by the comparatively low resource cost for productive use of the trained model compared to numerical weather prediction models.

In the following sections, we will present some basic information regarding the structure and training of neural networks.

3.2.1 Basic structure

The most simple representation of a deep learning model as in Figure 3.2 is the multilayer perceptron (MLP), where the model consists of few consecutive layers. The input layer - the first layer - gets the raw data as input and processes it, before sending the data to the next layer. Next, there are usually one or multiple hidden

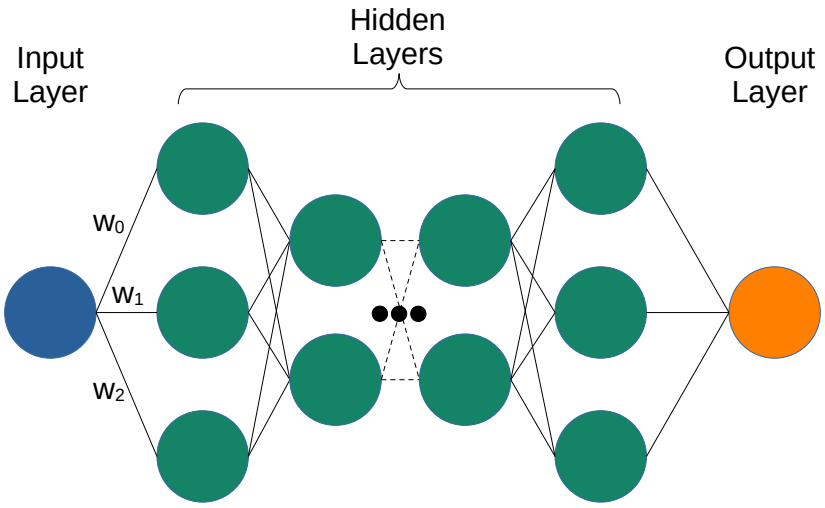


Fig. 3.2.: Multilayer perceptron sketch

layers, each of which taking the output of its previous layer. Again, the data is processed and sent to the next layer. The final layer then provides the model's inferred value. A simple representation of such a layer is that it consists of multiple neurons, the basic building blocks of a neural network. Each neuron of a layer in our MLP is connected to all neurons of the previous layer, whereas each connection has a certain weight. A neuron obtains all inputs, weighted by the connection's weight, processes the data based on internal values and applies an activation function to provide an output value. During training, a neuron's parameter, for example the connection weights, internal bias values, etc., are adjusted until the training is finished. The resulting parameter in combination with the basic architecture then describe the fully trained model.

While the multilayer perceptron is a very simple feed forward network (FFN), several other architectures have been developed for specific tasks. E.g. Convolutional neural networks (CNNs), which use convolution operations to restrict a neuron's possible predecessors to a spatial grid. These types of networks are often applied for image processing tasks. Recurrent neural networks (RNNs) have been applied for language processing. These networks contain feedback loops, where processed information may be reused in a future iteration to model time or spatial correlations. A recent highly performant architecture are transformer networks, which dynamically modify their weights using an attention mechanism, which calculates correlations or dependencies between different inputs. For several tasks these models are currently state of the art. There has also been work which showed that CNNs are capable of providing similar results to transformer networks, given that several of the insights gained from training and tuning transformer networks are applied [35].

3.2.2 U-Net architecture

We previously mentioned some common architectures of neural networks. Within this thesis, we restrict ourselves to a certain type of neural network, called the U-Net, which we will explain more in detail. Our implementation of the U-Net is based on convolutional neural networks.

As image data can contain thousands of pixels, training a fully connected network, may become infeasible unless a very shallow architecture is chosen. In addition, in several tasks it is not important, where certain objects or features are positioned globally in an image, but rather only a local dependency is of importance. For such tasks, convolutional neural networks were designed. As the name indicates, these networks use convolution operations. Kernels, usually much smaller than the input data, are convoluted over the input image, evaluating the underlying patch and providing a new output. As a result, the model size only depends on the number of convolutional kernels and their respective kernel size, independent of the input. Furthermore, a fully convolutional network can process data of varying sizes, as the convolutional operation works independent of the spatial extent of the respective data.

A specialized CNN architecture, originally introduced for biomedical image segmentation [61] is the U-Net, which gains its name from the U-shaped data flow chart. This architecture consists of an encoding and a decoding phase. During the encoding phase, several blocks of layers, such as CNNs, are applied to the data before the results are downsampled to a lower resolution. However, when segmenting data, usually a segmentation mask at the input resolution is desired, as such during the decoding stage the U-Net needs to apply upsampling at the end of each resolution stage. To include fine scale information of the data, skip connections are introduced, where data from the encoding blocks are directly concatenated to the input of each decoding block of the same resolution. While initially the layers within the encoding and decoding blocks were CNNs, newer versions of the U-Net architecture may consist of transformer networks or other architectures, which often provide even better results.

3.2.3 Training

As in most machine learning methods, a neural network needs to be trained on a dataset, commonly referred to as the training dataset. During training, the network randomly selects a batch of data from the training dataset, calculates a result and

evaluates this result with a given loss function. To actually train the network, its parameters need to be adjusted. This is done by optimizing the trainable parameter based on gradient information obtained from evaluating the loss function. An optimization algorithm such as gradient descent is applied, propagating the gradient backwards through the network using the chain rule. During this backpropagation step, all trainable weights are updated accordingly, ideally improving the network's skill. Once the update is complete, the next iteration can start, where the next samples are processed using the updated model weights. Processing all batches of data is referred to as a training epoch. The overall training procedure consists of multiple epochs, processing batches from the training dataset within each epoch. Training is usually repeated until convergence or some user defined threshold is reached (e.g. maximum number of training iterations).

Generally, the choice of loss function is an integral part of a network's training, as the networks strives to minimize the loss. As such, the loss function should be chosen, such that minimizing it yields good performance for the desired task. Commonly used basic loss functions are for example the mean root squared error (MRSE), binary cross entropy (BCE) or intersection over union (IoU).

A common problem that may arise during training of neural networks is overfitting. This describes the fact, that a machine learning algorithm may be able to perform well on the training dataset, while being unable to yield good results on previously unseen data. This can for example occur if the training data is relatively small compared to the network's modeling capability, such that instead of learning a general model, the network rather learns a mapping from the limited training input to its corresponding output. Once the network infers data not present in the training set, this mapping no longer works and the network fails to create useful results. It is therefore important to evaluate the network's performance with other data, to measure how well the network is able to generalize outside the training dataset.

To do so, one usually uses two additional datasets, disjoint from the training dataset and each other. The validation dataset is evaluated during training after a certain amount of training epochs. The score of this validation step is used to determine the network's potential on data, which was not used for training. As a result, this score is a better measurement on the network's overall quality, as it should be less prone to overfitting. Therefore, this score is oftentimes used to identify when to stop training. Since the network implicitly optimizes against the validation dataset during training, a test dataset is used to provide an evaluation of the network's skill, independent of any data which has been seen during the training process. This dataset is only

evaluated once training has been completed, providing an evaluation on data, which was not used during the training procedure in any way.

3.3 General-purpose computation on GPUs

3.3.1 GPU

While originally designed to speed up graphical pipelines, GPUs have by now been adapted to several other use cases. In contrast to many-core architectures such as CPUs, which often only contain a few dozens of cores, like the AMD EPYC 7713 with 96 cores, a GPU contains more than ten thousands cores which allow for a high degree of parallelization. In addition, GPUs have access to high bandwidth local memory, speeding up memory accesses. As a drawback, the available memory on GPUs is usually more limited than on a CPU. For example, current GPUs such as the NVIDIA H100 have around $80\mathrm{GB}$ of memory per GPU. In comparison, current CPU hardware supports several TB of memory. It is however possible to use multiple GPUs per node to increase the available GPU memory per node, although this usually increases the complexity of code. To overcome this problem, specialized chips such as the recently introduced NVIDIA Grace-Hopper compute chip GH200 are developed. The GH200 provides a very large memory of up to $624\mathrm{GB}$ which can be directly accessed by the GPU, notably increasing the available fast access memory per GPU. The application of GPUs for general problems not limited to graphics is called general-purpose computation on GPUs (GPGPU). To create GPGPU programs running directly on GPU hardware, several frameworks such as CUDA (compatible with NVIDIA GPUs) or ROCm (compatible with AMD GPUs) exist. In this thesis, only NVIDIA GPUs and correspondingly the CUDA programming framework have been used, which we will present in more detail in the next sections. Although, the parallelization concepts could also be applied to other GPU hardware.

3.3.2 CUDA programming model

While code executed on GPUs can achieve significant speedups over sequential and even parallel CPU code, one needs to provide an efficiently parallelized code to achieve this potential. Within the CUDA C++ programming model, a programmer can write code, which can be executed on NVIDIA GPUs. The general workflow of a CUDA program is to copy data from the host (the CPU) to the device (the

GPU), process the data on the GPU before eventually copying the results back to the host to finalize the program. To differentiate whether a function within the code has to be compiled for the host or device, CUDA provides so-called *execution space identifier*, `__host__`, `__device__` and `__global__` , which indicate where the code snippet is supposed to be executed. For example, a function declared as `__device__` can only be executed on the GPU not on the CPU. To issue a function to run on the device, the host processor needs to call a kernel. A kernel is a function, which serves as an entry point for the device execution. A function that should be called as a kernel needs to have the `__global__` execution space identifier. The call to the function needs to be parametrized using the *execution configuration syntax* `<<< ... >>>` which, among other, defines how many threads should be used for the execution of the kernel. Within a kernel, a function may freely call any `__device__` functions.

A kernel is executed based on the *execution configuration parameters* issued with the call. All spawned threads together form the compute *grid*. The *grid* itself is organized in several subgrids, called thread *blocks*, each of which containing a certain, different subset of threads. The size of a block, i.e. how many threads are grouped within a block, is again determined by the execution configuration syntax. The amount of blocks within the grid is defined in the same way. The threads can be spawned in a $1, 2$ or 3 dimensional grid and each thread is automatically issued a unique position within the grid, which can be queried from the kernel, using the provided `threadIdx`, `blockIdx`, `blockDim` and `gridDim` structs. Each thread belongs only to a single block. Note that the value of `threadIdx` only corresponds to the thread's position within its corresponding block and not the global position, thus threads belonging to different blocks can have the same value. To obtain the global position, one needs to include the offset determined by the `blockIdx` of the block as well. Threads within a block are further organized in groups of 32, called *warps*. Up to the $Volta$ GPU architecture, all threads within a warp were executed in lock step. Starting from $Volta$ threads within a warp can be scheduled independently. Furthermore, a kernel can contain more than $2^{31} - 1$ blocks. However, a GPU contains far less streaming multiprocessors (SMs) (e.g. 132 for the `H100 SXM5` [@11]). During execution of device code, each SM can execute only a limited number of blocks simultaneously, applying context switches to other blocks, when possible. Since compute capability 9.0, it is also possible to define thread block clusters, which currently allow up to k thread blocks to form a cluster, which are then guaranteed to be scheduled concurrently on the device. The supported portable cluster size k is set to 8, although this limit is stated to be architecture-specific.

3.3.3 Memory management

GPUs contain multiple levels of device memory, with differing sizes and access speeds. Global memory is the main memory used for data storage on GPUs. It is larger than the other memory types on a GPU, although, access is usually slower. For example, the global memory of an NVIDIA H100, has a capacity of up to $80\mathrm{GB}$ with a bandwidth of more than $2\mathrm{TB/s}$. The main memory, as well as the L2 cache, can both be accessed by all threads on the device. However, the programmer needs to ensure thread-safe access. Shared memory, which resides on chip, is a local low latency cache shared among all threads within a single block. It allows exchanging data between threads within a block at a higher speed than global memory. Static shared memory is limited to $48KB$ per thread block. On newer GPU architectures, dynamic shared memory can be configured to use up to $227KB$ per thread block, instead [@10]. Threads within a thread block cluster may use distributed shared memory to communicate utilizing the shared memory of all blocks within the cluster. Lastly, each thread has up to 255 registers, which can be used to store thread local variables. Although, the maximum number of registers per SM is limited to $64k$ [@10]. A high register usage per thread may therefore reduce occupancy of the device, as fewer blocks may be scheduled in parallel, due to insufficient registers per SM. This could impact performance as computational resources are not fully utilized. Within a warp, we can use warp-wide collectives such as shuffles or votes to communicate between threads. This communication works on register level memory and is the fastest inter thread communication on the device.

The processing speed of a GPU typically vastly exceeds the performance of memory operations, which is why the memory to compute ratio, the fraction of non-register memory operations (e.g. reading or writing from main memory) per calculation, should be generally low. Nevertheless, memory accesses are not always avoidable. If such accesses are necessary it is recommendable to use the faster low latency memory such as shared memory, constant memory or texture memory where possible. Most preferably, computation should rely on register memory using warp shuffles for inter thread communication, whenever possible. For optimal performance, memory accesses should be coalesced to reduce the amount of data loads and stores from global memory. This is important as memory transactions per warp happen in aligned batches of 32 bytes. As such, if a warp for example accesses 32 consecutively stored 4 byte values such as single precision floating point values, only four memory transactions are needed if the access pattern is aligned. Strided access to global memory on the other hand may require up 32 memory transactions, which notably decreases performance [@9].

Part I

Detection and classification of atmospheric fronts

4 Front detection

Chapters 4 to 8 are based on the following paper. Several text passages and images are taken directly from the mentioned paper. I designed and implemented the presented software. I performed all evaluations and wrote several parts of the manuscript.

Automated detection and classification of synoptic-scale fronts from atmospheric data grids

Stefan Niebler, Annette Miltenberger, Bertil Schmidt, Peter Spichtinger
Weather and Climate Dynamics (WCD), Volume 3, Issue 1, 2022, Pages 113-137
http://doi.org/10.5194/wcd-3-113-2022

Abstract Automatic determination of fronts from atmospheric data is an important task for weather prediction as well as for research of synoptic scale phenomena. In this paper, we introduce a deep neural network to detect and classify fronts from multilevel ERA5 reanalysis data. Model training and prediction is evaluated using two different regions covering Europe and North America with data from two weather services. We apply label deformation within our loss function which removes the need for skeletonize operations or other complicated post-processing steps as used in other work, to create the final output. We obtain good prediction scores with Critical Success Index higher than 66.9% and an Object Detection Rate of more than 77.3%. Frontal climatologies of our network are highly correlated (greater than 77.2%) to climatologies created from weather service data. Comparison with a well-established baseline method based on thermodynamic criteria shows a better performance of our network classification. Evaluated cross-sections further show that the surface front data of the weather services as well as our network classification are physical plausible. Finally, we investigate the link between fronts and extreme precipitation events to showcase possible applications of the proposed method. This demonstrates the usefulness of our new method for scientific investigations.

4.1 Introduction and related work

Atmospheric fronts are ubiquitous structural elements of extra-tropical weather. The term *front* refers to a narrow transition region between air masses of different density and/or temperature [see, e.g., 75]. These air mass boundaries play an important role for understanding the dynamics of midlatitude weather and are usually related to clouds. Further, fronts are often associated with significant weather, such as intense precipitation and high gust speeds[see, e.g., 16, 14, 37]. Hence, fronts in the sense of separating polar from more subtropical air masses play a vital part of the communication of weather to the public and the public perception of weather in general, although this aspect may have lost some attention due to the use of colorful apps. Frontal surfaces exist also on smaller scales, e.g. in the context of sea-breeze circulation or local circulation patterns in mountainous regions. Even tropical weather systems might indeed produce similar features of transition regions of different air masses, but due to other mechanisms than in extra-tropical weather systems. The focus here and in much of the literature is on larger-scale fronts that can extend over several hundred kilometers and are often associated with extra-tropical cyclones [65]. In addition, also quasi-stationary fronts can extend over large distance, but they do not move strongly over time, e.g. the Mei-Yu front [e.g. 26]. These stationary fronts are as well foci of significant surface weather. Unfortunately, there is no generally accepted front definition, see, e.g. the discussion in Schemm et al. [65] and Thomas and Schultz [74]. Thus, the detection of fronts often relies on different measures, usually based on physical variables and including physical hypotheses or theories, as detailed below. Additionally, it is still on debate if a front detection should be guided by determining surface fronts (as, e.g., on the analysis charts of weather services), or even more on the physical (horizontal and vertical) structure [see also the summary in 76, 62]. We present tools for the analysis of the vertical structure of fronts in the second half of this thesis [48, 47].

Nevertheless, determining the position and propagation of surface fronts plays an important role for weather forecasting, and, of course, for research on synoptic scale phenomena. The traditional manual approach to front detection is based on the expertise of weather analysts at operational meteorological services, along some (mostly empirical) guidelines. With the advent of large, gridded datasets, e.g. reanalysis from different weather centers, as e.g. ECMWF or NCEP, in the second half of the past century the drive for objective means to detect fronts automatically set in [see, e.g., 22]. Currently used methods are typically relying on detecting strong gradients in either temperature and humidity fields (e.g., by using equivalent potential temperature or wet-bulb temperature) or in wind fields [64]. The former

methodology goes back to the work by Renard and Clarke [58] and is represented by Hewson [20], who suggested an automatic method to detect fronts in fairly coarse datasets based on the so-called "thermal front parameters", derived from thermodynamic variables. In these and subsequent studies this is often related to the second spatial derivative of the temperature, and one or more "masking parameters", i.e. thresholds of thermal gradients along the front or in adjacent regions. This or conceptually similar methods have been used in numerous studies to determine the global or regional climatological distribution of fronts [e.g. 4, 29].

For the investigation of fronts on the Southern Hemisphere Simmonds et al. [73] suggested an alternative approach that investigates the Eulerian time rate of change of wind direction and speed in the lower troposphere at a given location. A comparison of the two methods to identify fronts on a global climatological scale by Schemm et al. [64] revealed some agreement between the fronts detected, but also regional difference and systematic biases in the detection of certain front types by both algorithms: For example, the "thermal" method detects more reliably warm fronts than the method based on lower tropospheric wind speed and direction. In addition, the orientation of detected fronts differs in general between the two methods. In consequence Schemm et al. [64] also find differences in the global distribution of fronts and the amplitude of seasonal variations in front occurrence frequency.

While it is well known that different front detection methods provide different outputs [e.g. 64, 24], an objective ground-truth is difficult to find. Most studies developing or testing automatic detection schemes rely on manual analysis as the "gold standard" to test the accuracy and for tuning free parameters in the automatic detection schemes [e.g., 20, 4, 8]. However, it should be noted that manual analysis is affected to a large degree by subjective decisions, and hence the focus, interest and expertise, of the person conducting the analysis. Shakina [70] reports results from an inter-comparison study of different manual front analysis carried out independently in different divisions of the Russian Meteorological service up until the 1990s. Comparing the different archives, agreement on the presence or absence of a front in any one $2.5^\circ \times 2.5^\circ$ box was found in 84.8 % of cases. However, if only the presence of fronts in any one grid box is considered, the agreement dropped to 23 % to 30 % depending on the type of front. Shakina [70] further suggests that disagreement mainly arises from the detection and positioning of secondary or occluded fronts, which typically are associated with less marked changes in surface weather. It is likely that the differences between manual analysis by different forecasters in the meantime have not reduced, but they may potentially be reduced by strict guidelines for forecasters on the key decision features for positioning fronts.

Despite a none negligible subjectivity of manual analysis, it still offers many advantages over automatic methods:

1. In contrast to most automatic detection methods many different aspects, including temperature, wind, and humidity fields, surface pressure, but also surface precipitation and wind, are taken into account.

2. Manual analysis does not rely strongly on the choice of (arbitrary) thresholds that are needed in most automatic front detection algorithms.

3. Experience of analysts can be taken into account, especially on regional scales (e.g. with complicated terrain as in the Alps)

In order to address the over-reliance on specific variables some recent studies have suggested methods that combine not only temperature and humidity data but also include information on the wind field [e.g. 59, 53], or information on Eulerian changes in mean sea-level pressure [e.g. 17]. Nevertheless, these extended algorithms that are so far mainly used in regional studies still rely on choosing appropriate thresholds for the magnitude of thermal gradients or changes in the wind direction and speed.

The necessity of manually designing metrics and selecting thresholds for automatic front detection can be at least partly overcome by employing statistical methods and machine learning approaches. The key idea with this approach is that based on manual analysis a complex statistical method retrieves as much consistent information on patterns, important variables, and thresholds as is available in manual analyses and coinciding state of the atmosphere, e.g. from reanalysis datasets. Previous attempts on using machine learning approaches for front detection are discussed in more detail in the following section.

Bochenek et al. [10] used a random forest to predict fronts over Europe using data from the German Weather Service (Deutscher Wetterdienst, DWD). Their results indicate that it is possible to detect fronts with this method, however it does not seem to be very robust, as the probability of object detection varies greatly between the shown samples.

Recently, different groups have used Artificial Neural Networks (ANNs) to predict frontal lines from atmospheric data. Biard and Kunkel [7] used the MERRA-2 dataset to predict and classify fronts over the North American continent. Their network also classifies their predicted fronts using the four types: warm, cold, and stationary fronts, as well as occlusions. They used labels provided by the North American Weather Service (NWS).

Lagerquist et al. [32] used the North American Regional Reanalysis (NARR) dataset [40], to predict synoptic cold and warm fronts over the North American continent also using the NWS labels. While the network of Biard and Kunkel [7] creates an output on the input domain, the network of Lagerquist et al. [32] predicts the probability for a single pixel and needs to be applied to each pixel consecutively. Both methods rely on post-processing steps like morphological thinning to create their final representation of frontal data. Additionally, both methods only use a 2D mask for each input variable, not making use of multiple pressure or height levels. Matsuoka et al. [38] used a U-Net architecture [61, 72] to predict stationary fronts located near Japan.

In this study, we present a new method for automatic front detection based on machine learning using meteorological reanalysis as input data and trained with information on surface fronts provided by two different weather services (NWS and DWD). The overall aim is to investigate the degree to which machine learning approaches are able to replicate manual analysis on a case-study and climatological scale and the degree to which the learned features are consistent with meteorological expectations on the physical properties characterizing a frontal surface. Our provided network uses the U-Net approach to predict and classify all four types of fronts, and it does not require morphological post-processing. We evaluate our approach similar to Lagerquist et al. [32] using an object based evaluation method. Unlike the previous methods, we incorporate data from two different weather services, NWS and DWD, and also evaluate on the two different regions covered by these datasets. We additionally compare our predicted fronts against the method developed by Schemm et al. [64], using a thermal front parameter (TFP) as an example of a conventional automatic front detection method. We refer to it in the following as "baseline method". As input data we use the ERA5 reanalysis data [19] from the European Centre for Medium-Range Weather Forecasts (ECMWF) at a $0.25°$ grid at multiple pressure levels for each variable. This dataset exhibits a higher resolution than the NARR data ($32\ \mathrm{km}$ grid) used by Lagerquist et al. [32] and MERRA-2 data ($1°$ grid) used by Biard and Kunkel [7]. In contrast to these studies, we also used multiple pressure levels to refine our results.

Although we are aware of the conceptual differences between determining surface fronts and the complex 3D structure of fronts, we use the surface maps as a ground truth, i.e. as a proxy for the complex structures called fronts. However, in the later evaluation, it turns out that the detected surface fronts represent the expected physical properties of air mass boundaries in a meaningful way.

In Section 5 we describe our used data, network architecture and evaluation methods, respectively. In Section 6 we explain our evaluation methods and display our evaluation results on the training and test dataset. In addition, we showcase applications in terms of determining the variation of physical properties across fronts (Section 6.2) and relating fronts to extreme precipitation events (Section 7.1). We close with a summary of the study and a short outlook for future improvements as well as further applications of the new method for scientific purposes.

5 Data and Method

For each spatial grid point, our proposed algorithm predicts a probability distribution, describing how likely it is that the point belongs to each of our possible five classes: warm front, cold front, occlusion, stationary front, or background. Our method predicts that probability from a 4-dimensional input consisting of multiple channels located on a 3-dimensional multilevel geospatial grid, which was flattened to a 3-dimensional input by combining the atmospheric channel and level dimension. For this task, we use a convolutional neural network (CNN) architecture to automatically learn atmospheric features that correspond to the existence of a weather front at spatial grid points. We use a supervised learning approach, in which we provide ground truth data of frontal data sampled from two different weather services (surface fronts). We adjust hidden parameters of the CNN in order to optimize a loss function measuring the quality of our weather front prediction. CNN architecture and training will be explained in further detail in this section. Our network was implemented, trained, and tested using Pytorch 1.6 [54]. Parallel multi-GPU training was implemented using Pytorch's DistributedParallel package. The provided code was run using Python 3.8.2 and is freely available [46].

5.1 Data

We will briefly describe which channels and grid points were used as training input from the ERA5 reanalysis data [19]. Furthermore, we will describe the format of the corresponding label data of fronts obtained from NWS and DWD; in the case of the DWD label data, we additionally describe the preprocessing of the DWD data.

5.1.1 ERA5 Reanalysis Data

Our model input consists of a multichannel multilevel spatial grid provided by ECMWF's ERA5 reanalysis dataset. Each channel denotes a different atmospheric variable, while levels consist of a subset taken from the $L137$ vertical level definition [@5]. Data is represented on a spatial grid with a grid-spacing of $0.25°$ in both

Tab. 5.1.: Mean and variance of the individual variables used for normalization of input data.

variable	(unit)	mean	variance (in unit^2)
T	K	275.355461	320.404803
q	$\mathrm{kg\,kg^{-1}}$	$5.57926815 \cdot 10^{-3}$	$2.72627785 \cdot 10^{-5}$
u	$\mathrm{m\,s^{-1}}$	1.27024432	67.4232481
v	$\mathrm{m\,s^{-1}}$	0.10213897	43.6244384
w	$\mathrm{Pa\,s^{-1}}$	$5.87718196 \cdot 10^{-3}$	$4.77972548 \cdot 10^{-2}$
sp	hPa	865.211548	1494.6063
kmPerLon	$\mathrm{km}/^\circ$	0.64	0.09

latitudinal and longitudinal direction. Since we do not expect to obtain relevant information from high altitude level data, we decided to restrict ourselves to every fourth level within the inclusive interval $[105, 137]$, representing 9 model levels between the surface and about $700\ \mathrm{hPa}$. This range contains both the ground level information and the $850\ \mathrm{hPa}$ pressure level information, both of which are commonly used to detect fronts. Pressure values are defined as parameters of an affine transformation of the surface level pressure, which is why we manually added the surface pressure field to the data using the merge operation of the Climate Data Operators (CDO) [68]. This allows us to calculate the pressure at each grid point and level. We further only use 5 ERA5 multilevel variables as input for our network: temperature (T), specific humidity (q), zonal wind velocity (u, East-West), meridional wind velocity (v, North-South), and vertical velocity (w). In addition, the surface pressure (sp) and longitudinal distance per pixel in km relative to $27.772\ \mathrm{km}$ (kmPerLon) are considered. The distance between two pixels at a certain degree latitude is derived by assuming a spherical shape of the globe and is only used as a single level variable. Surface pressure on the other hand is used to estimate the pressure at each model level using the corresponding level parameter to create another multilevel network input. All resulting data is normalized with respect to a global mean and variance sampled from data of the year 2016. The resulting mean and variance values are listed in Table 5.1.

While ERA5 reanalysis data is available for the whole globe, the available ground truth labels only reside within the analysis region of their corresponding weather services. We therefore cannot use ERA5 data outside these regions. For this reason, we decided to restrict our usage of ERA5 data to rectangular subgrids, each of which being completely within the analysis region of the respective weather service analysis.

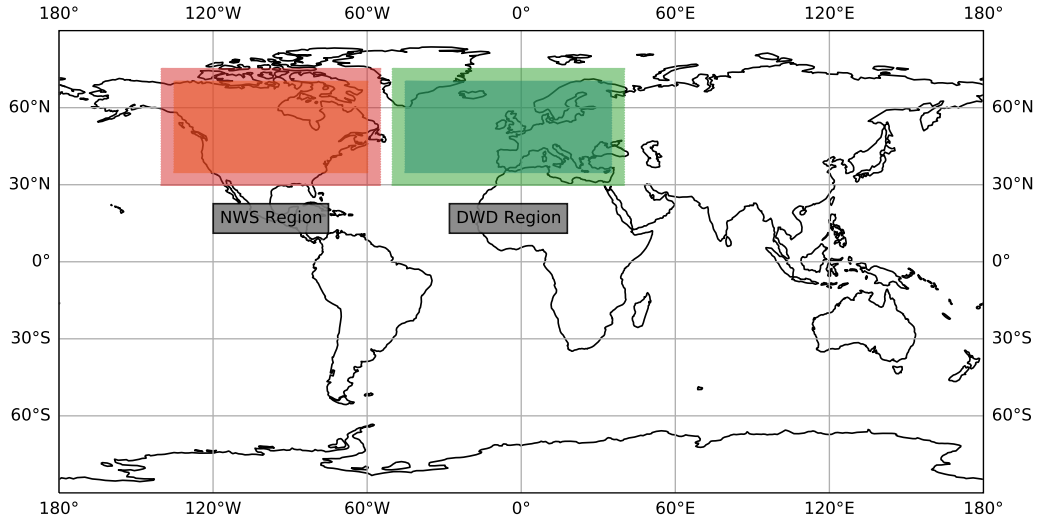


Fig. 5.1.: Bounding boxes for the two regions used for training and evaluation against the weather service labels. The brighter area is used as input, but is not used for evaluation.

The extent of these regions is described in Tab. 5.2 as $\text{DWD}_{\text{input}}$ and $\text{NWS}_{\text{input}}$. Pixels at the border of our input may lose critical information to successfully identify a front due to the input crop. As a result, detections on the outer $5°$ (20 pixel) of the input domain are not evaluated during training. While the network still outputs these pixels, they do not contain valid detections and should therefore be removed from the evaluation. As a result, the effective output region is smaller than the input region, as indicated in Tab. 5.2. This is also shown in Fig. 5.1 as the difference in shade within each weather service region. Prior to evaluation, we create detections for each sample using the global input data. Evaluations against the weather service labels are performed using the corresponding output regions. Comparisons against the baseline method use the same regions restricted to latitudes spanning $[35°, 60°]N$ instead. The evaluation in Section 7.1 does not rely on the weather service data and is therefore evaluated within $[-60°, 60°]N$ and $[-175°, 175°]E$. The restriction of the longitudes is caused by the smaller output regions, as explained in this section.

5.1.2 NWS Front Label Data

For training on the North American continent, we use the HiRes Coded-Surface-Bulletins (csb) of the North American National Weather Service [41]. This data ranges from 2003 up to 2018 and was previously used by Biard and Kunkel [7] and Lagerquist et al. [32]. Each front in a csb file consists of an identifier, describing the type of front, followed by a series of coordinate pairs on a $0.1°$ grid, defining

Tab. 5.2.: The input and output regions for the respective weather service analysis dataset used during training and the global input region. Levels are only used for network input. The output regions are also used during evaluation against the weather service labels. Every fourth vertical level between levels 105 and 137 is chosen to reduce the amount of input data, also in terms of redundant information.

Weather Service	Latitudes	Longitudes	Levels
DWD_{input}	$[30°N, 75°N]$	$[-50°E, 40°E[$	$[105, 137, 4]$
DWD_{output}	$[35°N, 70°N]$	$[-45°E, 35°E[$	-
NWS_{input}	$[30°N, 75°N]$	$[-140°E, -55°E[$	$[105, 137, 4]$
NWS_{output}	$[35°N, 70°N]$	$[-135°E, -60°E[$	-
Global	$]-90°N, 90°N]$	$[-180°E, 180°E[$	$[105, 137, 4]$

a polyline of the front. We do not perform any preprocessing on this data. In accordance with our available data, we restricted the use of the latter to the years 2012 to 2017 using only snapshots in a 6-hour interval to keep the amount of data balanced compared to the DWD data during training. The NWS dataset contains labels for the following front types: warm front, cold front, occlusion, and stationary front.

5.1.3 DWD Front Label Data

For training over Europe and the Northern Atlantic, we use label data extracted from the surface analysis maps of the Deutscher Wetterdienst (DWD) for the years 2015 to 2019. Unlike the Coded-Surface-Bulletins, these maps are not provided as polylines, but rather as PNG images of a region containing both the North Atlantic and Western Europe (see Fig. 5.2 (a)). Each of these images has a resolution of 4389×3114 pixel. To use the labels we extract each individual front, by creating coordinate pairs, which describe the front as a polyline, similar to a csb. Within an image, different types of fronts are color coded, which allows us to easily separate them from the background. Our algorithm first filters all fronts of a specific type by filtering all pixel of the corresponding color. In a second step, we erase all additional symbols on each line. This includes symbolic identifiers like half-circles and triangles, indicating the propagation direction of a front, as we do not need this information. Also, otherwise, these symbols could create false positive coordinate points in the label data. Subsequently, latitude and longitude coordinate pairs along each line are extracted in order to describe each front in terms of a polyline. In Fig. 5.2 (b) we show an example of a processed image file, redrawn onto the same projection as the input image. Blue and red lines in both panels correspond to cold and warm fronts

respectively, while green lines correspond to occlusions, which are pink in the left panel.

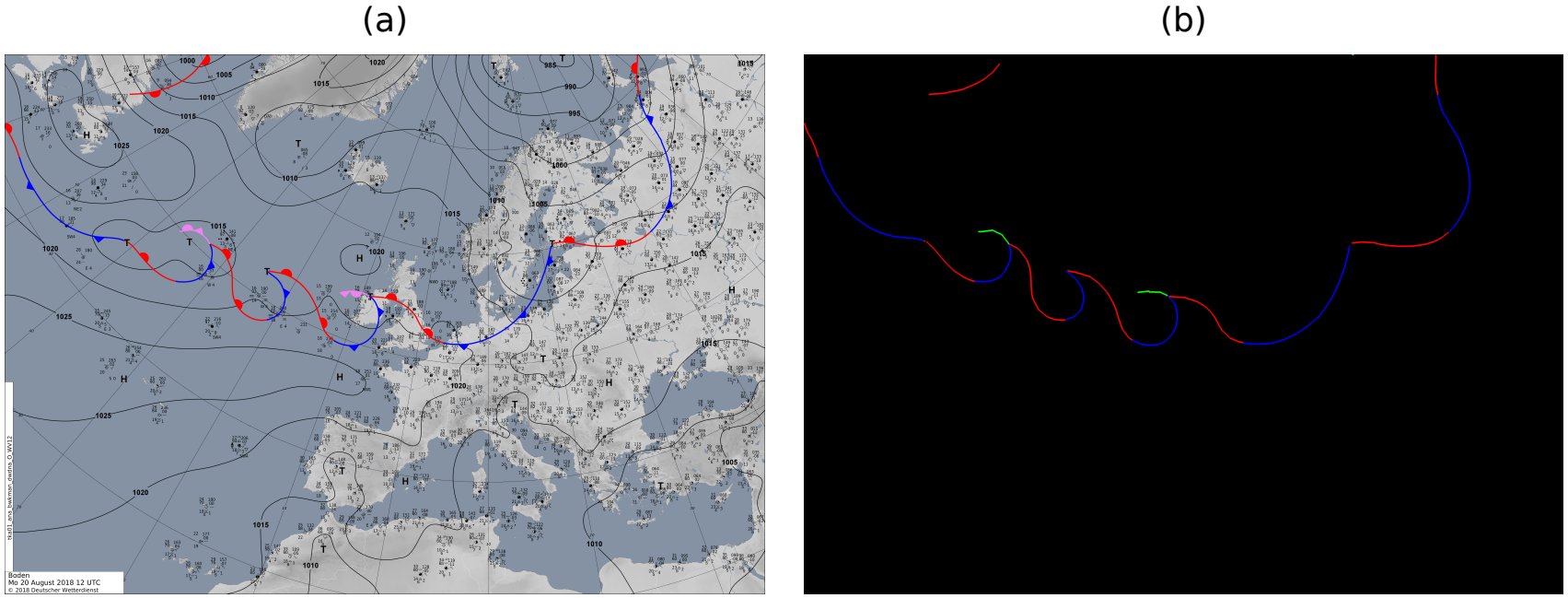


Fig. 5.2.: Example of well extracted fronts (b) from an image provided by the DWD (a) (Source: [@3]). In panel (b) blue and red lines correspond to cold and warm fronts as in the original image (a), green lines correspond to occlusions which are pink in the input image. Note that stationary fronts are originally depicted as alternating warm and cold fronts. For this reason, we cannot distinguish those from regular cold and warm fronts.

In certain cases, our method fails to correctly extract the frontal lines. These cases lead to gaps within a front, wrongly extracted objects or wrongly connected fronts. Gaps originate from two factors. One is that another object is drawn on top of a frontal line, effectively splitting the front into two parts. The other is an aggregation of multiple front-symbols on a short segment. As our method removes sections where a symbol is placed before reconnecting the remaining points, crowded placement of these symbols may make the remaining part of the front too short to be considered relevant and as such will be omitted. Wrongly extracted objects occur mostly due to tropical storm symbols that are depicted in the same color as a warm front. As such, our extraction method wrongly extracts these objects as well. Finally, errors can occur when we try to sort the extracted coordinate pairs of a single front. In some cases, the sorting method may end up stuck in a local minimum, resulting in a wrong order of points. An example of such a faulty extracted image is shown in Fig. 5.3. However, these are relatively rare, only account for a small portion of fronts within a sample and many are going to be masked by the lower resolution of ERA5, which is why we ultimately decided to ignore these cases for this work.

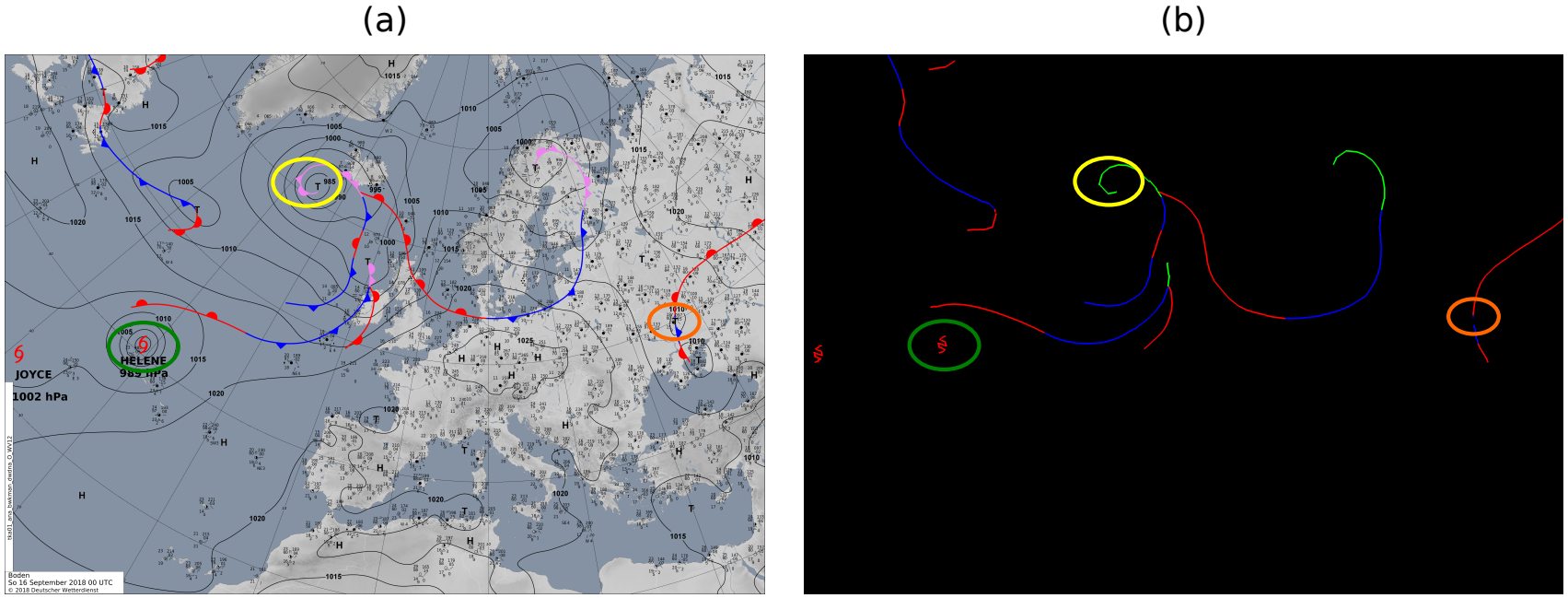


Fig. 5.3.: Example of badly extracted fronts (b) from an image provided by the DWD (a) (Source: [@3]). **Green Circle:** Object that is not a front, but has the same color coding, is wrongly extracted as a front. **Orange Circle:** Unrelated symbol is drawn over the front. The front could not be extracted completely. **Yellow Circle:** Frontal symbol is placed in an area with high curvature. The curvature is not extracted exactly, as the symbol is removed during the procedure and the loose ends are connected with a straight line.

We can extract information for the following front types: warm front, cold front, and occlusion. Since stationary fronts are indicated by alternating warm and cold fronts, we cannot extract this information from the images as obtained from DWD; this would interfere with the classification of warm and cold fronts.

5.2 Network design and training

5.2.1 Network architecture

Neural networks are a machine learning technique where a network consisting of several layers is used to extract feature representations of an input at different levels. Each layer transforms its input into an output map, the layer's feature map. These feature maps can then be used as an input for consecutive layers, which enables the network to learn more detailed features within the data. In a Convolutional Neural Network (CNN) the most common transformation function is a convolution of the input image with a convolution mask where each entry is a trainable, latent parameter of the network. During training, these parameters are adjusted to optimize a loss function, which measures the quality of the output of the network. In our case, we use a U-Net Architecture originally introduced by [61] for biomedical segmentation. The proposed architecture consists of several consecutive blocks that gradually extract features from the data and reduce the spatial dimension of the input data to extract features on multiple scales (Fig. 5.4). These blocks are followed by a number of expansive blocks which gradually increase the resolution up to the original scale. Additionally, at each resolution scale a so-called skip connection allows the final feature map of an encoding block to directly serve as additional input to the corresponding decoding block, displayed as gray arrows in Fig. 5.4. These skips improve the network's ability to localize the features, as the upsampled features only hold coarse localization information. In our network we use convolutional layers as explained before. Additionally, we use Rectified Linear Unit (ReLU), Batch Normalization, Pooling, Upsampling and 2D-DropOut layers, whose functionality we will briefly explain. The dropout chance at each 2D-DropOut layer is set to 0.2.

- ReLU layers are used to introduce non-linearity into the network. They transform each input x as $\text{ReLU}(x) = \max(0, x)$
- Batch Normalization layers normalize the batched input to a mean of 0 and variance of 1. They can have additional learnable affine parameters.

- Pooling layers transform several input grid points to a single output gridpoint. Common operations are averagePooling or maxPooling where the grid points are combined calculating the average or maximum of the input, respectively. This operation is used to reduce the resolution of the feature map.
- Upsample layers are a simple upsampling of a grid point to increase the resolution of the feature map.
- 2D-Dropout layers randomly set all values in a channel to 0 to reduce overfitting.

A sketch of the used architecture is shown in Fig. 5.4. We use Pytorch's DistributedParallel package to enable training on multiple GPUs in parallel. Training is performed on a single node, with each GPU acting on a fixed shard of the available data.

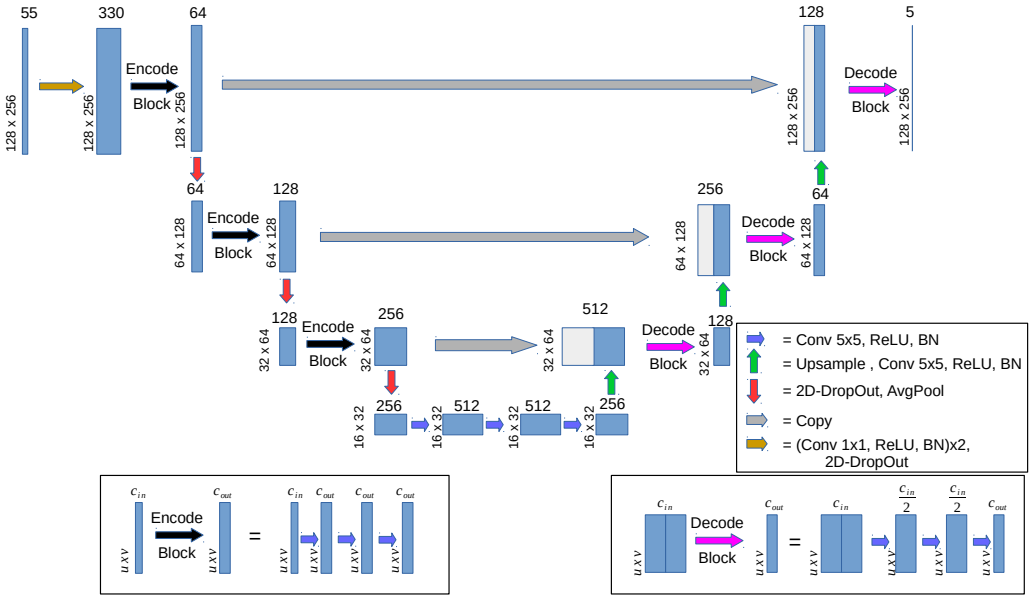


Fig. 5.4.: U-Net architecture used in this paper. The first convolution of the input data uses a 1×1 sized kernel instead of 5×5. Decode and encode blocks are explained in the boxes at the bottom of the image. Each Decode and Encode block consist of 3 sequential blocks: convolution, ReLU and BN. $U \times V$ describes the image size per channel. C_{in} and C_{out} describe the number of channels of the input and output of an encode or decode block. The copy operation simply copies the blue box at the start of the arrow into the white box at the end. The white and blue boxes then describe the concatenation of the output from the copy and upsample operations. The number at the left-hand side of each block denotes the spatial input dimension. The shown sizes are those used during training, however the initial spatial dimension can be chosen freely as long as it is divisible by 8. At each red (green) arrow, the dimension is divided (multiplied) by 2. The number on top of each block denotes the number of channels for each block and must not be changed.

Tab. 5.3.: Distribution of our data into training, validation and test datasets. For each dataset, the covered time frame and number of labels are shown. All models use the same validation and test data.

Dataset	years	samples
test data	2016	1464
validation data	2017	1460
training both	2012-2014, 2015/03 - 2015/12, 2018, 2019	8526
training NWS	2012-2014, 2015/03 - 2015/12	5608 (only NWS label)
training DWD	2015/03-2015/12, 2018, 2019	4142 (only DWD label)

5.2.2 Dataset augmentation

In each epoch and for each timestamp, we randomly select one of the available weather service labels for the given timestamp. Depending on which weather service was chosen, we crop a 128×256 pixel sized sub-grid residing within the corresponding weather service's input region (see Table 5.2) from the ERA5 data. We use this smaller crop instead of the complete region to increase the number of training samples, reduce the memory footprint on the GPU during training, and to ensure that all input dimensions are multiples of 8. The extracted label data is also cropped by removing each vertex, where neither the vertex itself nor a neighboring vertex is located within the extent of the ERA5 crop. To further increase sample count via data augmentation, we also perform random horizontal and vertical flips on the data. It is important to note that, whenever data is horizontally (vertically) flipped the sign of the input variable v (u) has to be flipped as well, as these variables describe a vector field rather than a stationary value. Flipping of the data might also lead to a better representation of fronts in the Southern Hemisphere, which are "mirrored" at the equator (see video supplement [@8]).

5.2.3 Training

Our model is trained using stochastic gradient descent with Nesterov momentum of 0.9 to minimize the loss function. The initial learning rate is set to $0.005 \cdot \#Ranks$, where $\#Ranks$ corresponds to the number of processes used for the parallel training. We train the network for several epochs. Within each epoch, the algorithm randomly trains on a permutation of the complete training dataset. Every 10 epochs, we measure the training loss. If the test loss does not improve for 10 test phases, we divide the learning rate by 10 up to a minimum of 10^{-7} and reset the count, if the learning rate was changed. If the test loss does not improve for 20 test phases

(200 epochs) and we cannot reduce the learning rate anymore, we stop training. Additionally, we set a maximum of 10000 training epochs or 3 days time as stopping criteria. At each test step, we save a snapshot of the network if the test loss is better than the currently best test loss. Our final network is the resulting network which yielded the lowest test error.

5.2.4 Label extraction

As described by Lagerquist et al. [32], the frontal polylines are subject to two non-negligible causes of bias: inter- and intra-meteorologist. The first bias describes the effect that two meteorologists may disagree on the exact location of a front, the occurrence of a front at all, or which exact shape the frontal curve follows. The second bias describes the effect that the same meteorologist may be biased on the placement of a frontal line due to fronts placed at previous analysis times by the same person.

The transformation of these curves into poly-lines and the application onto a different resolution is subject to creating additional label displacements. While these problems are present in most human labeled data, it is more peculiar in this specific case because

the ideal poly-line should have a width of only *a single pixel*. As a result, each ever so slight displacement introduces a large per pixel disparity between two fronts, as the intersection of the sets of pixels that describe these fronts ends up being close to non-existent. This has at least two negative effects. First, the gradient information is really sparse, as a close prediction will be considered false positive just as a far off prediction, as can be seen in the example of Fig. 5.5 a. Further translating the green line to the right, will barely affect the count of intersecting pixels with the red line, even though one would consider the detection becoming worse the further it moves from the label. Secondly, the previously mentioned label offset due to personal bias may lead to the case that a labeled front is not located exactly at the physical frontal position, essentially creating a false label with wrong underlying atmospheric properties. Due to the low intersection count, a correctly placed detection will now score badly.

One way to handle this might be to widen the extracted front labels. While this approach introduces further false positive labels, slight translations in the detection are less penalized as they are more likely to be covered due to the larger width of the labeled data. Additionally, the network is inclined to also detect wider frontal lines,

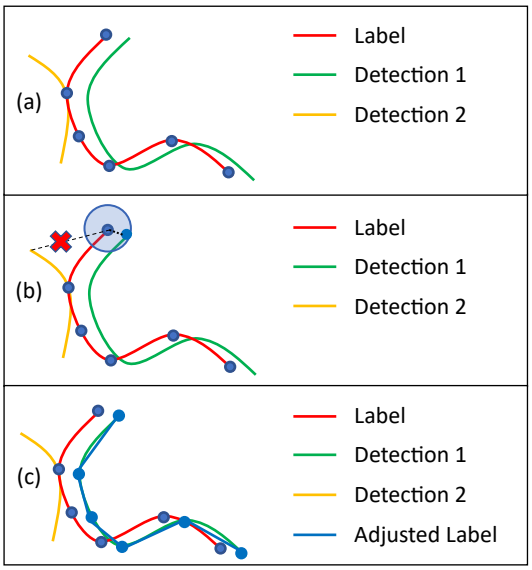


Fig. 5.5.: Sketch of our label adjustment method. (a) Initial Weather Service label with polyline vertices (blue dots) and 2 possible detections. Detection 1 initially scores lower than Detection 2 due to a lower intersection with Label. (b) Display of how a vertex of Label might be adjusted within a search radius for Detection 1. The possibly optimal position for the vertex regarding Detection 2 is not within the search radius of the vertex. Deformation will therefore not be able to create a good intersection of the upper part of Detection 2 and Label. A similar situation occurs for the three vertices at the bottom right of Label. (c) Possible resulting Adjusted Label after each Vertex was adjusted. The Label was deformed onto Detection 1 as it creates the best matching score. Detection 2 is too far from several vertices of Label and cannot score a similar matching score with any deformation of Label. As a result, Detection 1 now scores higher than Detection 2.

making it even easier to create intersections. In the same way, the effect of positional bias of the label placement is also reduced as the widened label is more likely to cover the physically correct location, if a small translational bias exists. However, this bias is not completely negated. From our studies and the results of previous studies [e.g., 38, 32, 7], it seems apparent that a deep learning architecture learns that a bias in label placement exists and, as a result, tends to predict enlarged lines, trying to cover the uncertainty caused by the bias. Using enlarged labels further enhances this effect, leading to even larger line width, which in return leads to a low spatial accuracy of the detections. To regain positional accuracy, previous work used a morphological post-processing step to extract thin lines from wider network predictions.

In this work we use a different approach, as illustrated in Fig. 5.5 panels b and c, to counteract this initial loss of positional accuracy. Instead of widening the label, we deform the given polylines prior to evaluation, by translating the vertices within a restricted search radius (panel b). All possible deformations are considered and evaluated according to a matching function, and the highest scoring deformation is then used for evaluation (panel c). This approach encourages the network to predict fronts with a high spatial certainty, as the labels themselves remain thin, while the deformation models the positional bias.

A polyline j consists of a series v_j of vertices $v_{j,i}$, where each $v_{j,i}$ describes the coordinate pair of the vertex as it is extracted from the weather service label. Additionally, each deformed polyline contains a series of translations tr_j, consisting of a translation vector $tr_{j,i} = (u_{j,i}, w_{j,i})$, which describes the translation of $v_{j,i}$ within the polyline j. A segment of the deformed polyline j is then edge $e_{j,i}$ connecting $v_{j,i} + tr_{j,i}$ and $v_{j,i+1} + tr_{j,i+1}$. We calculate the matching score of a segment as follows:

- calculate the positions of pixels of the line connecting $v_{j,i} + tr_{j,i}$ and $v_{j,i+1} + tr_{j,i+1}$
- sum the values of all pixels in the network output that are on this line
- weight the sum by $1 + \exp(-0.5((\frac{u_{j,i+1}}{\sigma})^2 + (\frac{w_{j,i+1}}{\sigma})^2))$
- reduce the result by the number of pixels in the line connecting $v_{j,i}$ and $v_{j,i+1}$

The matching score of a polyline is considered the sum of the matching scores of each line segment of the deformed polyline.

The third step models the assumption that the provided labels are generally placed correctly, and that strong deformations are less likely. Therefore, a low deformation

is preferred to a strong deformation if the intersection with the network output is the same. This matching procedure operates ignorant of the classification results and only takes the presence or absence of any type of front at a given pixel into account. We restricted ourselves to deformations where $-k \leq u_{j,i}, w_{j,i} \leq k$ with $k = 3$, keeping the deformation radius small to only counteract the positional bias of the label, which we expect to be small. Additionally, we chose $\sigma = k$. We do not change classification information of the labels during the procedure. Thus, each front is extracted as the class provided by the weather service. This matching procedure was implemented using C++ and Pybind11 [28].

This method comes at the risk, that instead of predicting the position of the front, the network may end up detecting a systematic displacement of the front within the range of the $(2k + 1) \times (2k + 1)$ grid. We believe this could happen for two possible reasons: (i) the label bias exhibits a systematic displacement itself, and (ii) k is chosen too large. In the first case, the error lies within the labels, and it is generally questionable whether or not these labels are suitable for training at all. The parameter k controls at which distance from the labeled front the detection may still be considered correct. With increasing k, the incentive to place the detection close to the provided label reduces, diminishing the spatial accuracy of the predictions. Therefore, we have chosen $k = 3$, allowing each vertex to displace itself up to 3 pixels in each direction, limiting the scope of movement to a sensible range.

As an example, Fig. 5.5 shows how this algorithm can help to solve the problem of a correct detection being penalized by a biased label. We assume that the green line (Detection 1) is a correct detection with appropriate underlying atmospheric properties, while the yellow line (Detection 2) is an artifact caused by unfinished training of the network. Additionally, the red line was drawn biased and is therefore not located at the appropriate position, regarding the underlying atmospheric features. In panel (a) the correct prediction has very few pixels intersecting with the label, similar to the wrong prediction. Not performing any deformation would wrongly count several pixels of the green detection as false positives, while only resulting in a similarly low number of pixels considered true positive, similar to the yellow detection. However, when using the deformation algorithm most pixels of the green detection correctly count as true positives, while the yellow detection is correctly classified as false positive. A deformation towards Detection 2 does not occur in this example, as the yellow line is out of range for most vertices. Most segments will therefore not intersect with the yellow line, leading to generally lower matching scores than the displayed blue line. The latter further displays the importance of the choice of k for preventing the label from deforming onto a wrong detection.

5.2.5 Loss functions

During training, we extract the label lines as described in Section 5.2.4. As a loss function we decided to use a loss based on *Intersection over Union* (IoU), which we evaluate for each output channel individually, before combining them by a weighted average. This loss function inherently circumvents the problem that in each channel most of our output belongs to the background as it does not contain a front. While the original formulation of IoU is used for sets and therefore a strictly binary labeling, we used an adjusted version that works with floating point probabilities. This loss function is also used by Matsuoka et al. [38]. However, they only evaluate it on a single output channel. The definition of loss for a single output channel is given by the following equation:

$$L(p,x) = 1 - \frac{\sum_i p_i \cdot x_i}{\sum_i p_i \cdot p_i + \sum_i x_i \cdot x_i - \sum_i p_i \cdot x_i} \tag{5.1}$$

Here L denotes the loss function, x is the extracted label image, and p the prediction of our network. p_i and x_i are the i^{th} pixel of either p or x. We subtract the loss function from 1, as we will minimize our loss function during training, as the IoU normally increases the better the prediction becomes. $L(p,0)$ always evaluates to 1 regardless of p, which means we do not obtain much information from such a label. When combining our network's output channels, we try to adjust for this problem. We define a variant of L, denoted as L^0, that simply omits evaluation for all $L^0(p,0)$ by setting the result to 0. In all other cases $L^0 = L$. These omitted cases therefore will not influence the training gradient. As our network generates a multichannel output, we calculate a loss for each channel individually and combine the results. The first output channel corresponds to the background label, which corresponds to the absence of fronts. We invert this output, by subtracting it from 1, to get a value describing the presence of fronts. As a result, we obtain 5 output channels describing fronts (front, warm front, cold front, occlusion, stationary front) denoted as $k \in 0,1,2,3,4$. Additionally, in each batch b we have $batchsize$ samples b_n and for each b_n we have a detection p_{b_n} and a label x_{b_n}. The respective data in the channel k is then denoted as $p_{b_n,k}$ and $x_{b_n,k}$. For each b_n, we calculate $L_{b_n,0} = L(p_{b_n,0}, x_{b_n,0})$. For the classification channels $k > 0$, we calculate $L^0(p_{b_n,k}, x_{b_n,k})$ instead and denote these results as $L^0_{b_n,k}$ correspondingly. By doing so, we may omit some samples where no label is present within the respective channels. To compensate, we define a weight $s_{b,k} = \frac{batchsize}{nz_{b,k}}$ for $k > 0$, where $nz_{b,k}$ is the number of samples in b where there is no label in channel k. This weight is used to balance the potentially different

counts of labels for the individual channels. The resulting loss for one $b_n \in b$ is calculated according to

$$E_{b_n} = 0.2\ L_{b_n,k} + 0.8\ \frac{\sum_{k=1}^{4} s_{b,k} L^0_{b_n,k}}{4} \tag{5.2}$$

The values 0.2 and 0.8 are chosen to formulate a weighted average over all channels. In the case of $nz_{b,k} = 0$, we set $s_{b,k} L^0_{b_n,k} = 0$. In this case, channel k will not be evaluated at all within the current batch. The loss for the complete batch can then be calculated as the mean of all E_{b_n} within the batch b:

$$E_b = \frac{\sum_{b_n \in b} E_{b_n}}{batchsize} \tag{5.3}$$

5.3 Baseline method

We compare our results against a baseline method developed and used at ETH Zurich. The method introduced by Jenkner et al. [29] and later modified by Schemm et al. [64] uses thermal gradients and other information to predict fronts. While the method was originally designed to work on a $1°$ resolution grid, we adjusted the hyperparameters of the method to allow it to run on a $0.5°$ grid[1]. In the baseline method, i.e. that designed for the ERA-Interim dataset with a grid spacing of $1°$, a minimum equivalent potential temperature gradient of $4 \cdot 10^{-2}\ \mathrm{K\,km^{-1}}$, a minimum advection velocity of $3\ \mathrm{m\,s^{-1}}$, and a minimum front length of $500\ \mathrm{km}$ is used. We decided to keep these physical values identical to the original algorithm to retain similar physical properties of the front. However, we have altered parameters used for the a-priori smoothing of the equivalent potential temperature gradient field (number of filter applications as described in Jenkner et al. [29] increased from 5 to 7), the smoothing of frontal lines (smoothing parameter changed from 5 to 15), as well as the minimum size of front objects in number of grid-points (increased from 15 to 20). The largest impact comes from adjusting the smoothing of the equivalent potential temperature gradient field. Using these altered settings, the number of fronts detected in the northern and southern extra-tropics increases by about 30 %, but the spatial distribution of fronts is very similar to the original ERA-Interim dataset with some exceptions in the vicinity of steep terrain (not shown). Our network works on a $0.25°$ resolution grid and outputs on the same domain.

[1] A tuning of the method for the $0.25°$ resolution was not possible, since features on small scales disturb the evaluation of the gradients

Therefore, when comparing against the baseline method, we resample the network output to a $0.5°$ resolution using a 2D maximum pooling operation. The authors of the baseline method mention that the provided baseline should only be applied to the midlatitudes. When comparing against the baseline, we therefore restrict ourselves to the midlatitudes of the Northern Hemisphere for a fair evaluation.

5.4 Evaluation methods

We will briefly explain how the data is processed for the evaluation and how the Critical Success Index (CSI) is calculated.

5.4.1 Trained models and dataset distribution

We distribute our data into a test (year 2016) and a validation (year 2017) dataset, and create 3 training datasets as described in Tab. 5.3. We train a total of 3 models, one for each training set. The models trained using *training NWS* (*training DWD*) are additionally restricted to only use label data from the NWS (DWD) during training. Each model is trained using 6 GPUs on a single node of the Mogon II cluster of the Johannes Gutenberg University. Each node contains 6 NVIDIA GeForce GTX 1080 Ti GPUs and an Intel Xeon CPU E5-2650 v4 with 24 cores and hyperthreading. Data was staged in prior to training to enable reading from a local SSD rather than the parallel file system. The models trained using *training NWS* and *training DWD* are only used in Section 6.1.1 with results presented in Tables 6.1 and 6.2 as well as in the Appendix A in Tables A.1 and A.2. In all other cases, the model using *training both* is applied.

5.4.2 Test data processing

For the evaluation, we process each input file in the test dataset as follows:

- Apply the respective model on the global input region of the current sample.
- Apply a softmax activation function to the raw network output to generate a probability mask for the sample.
- Create a binary mask by setting each entry in the probability mask to 1 if it is greater than 0.45, else to 0.

- Use one iteration of 8-connected binary dilation and calculate all different connected components. Each connected component is considered an individual front.
- Filter the labeled image with the undilated binary mask to remove the dilation effect.
- Remove all fronts that consist of less than 2 pixels.
- Write the binary mask to disk.

During evaluation, we then load the corresponding binary mask from disk and crop it to a sub-region when necessary. Results of the baseline method and the weather service labels are already provided in binary format.

5.4.3 Front to object conversion

Prior to evaluation, the generated binary masks of our network output are transformed into front-objects in two steps.

- Use one iteration of 8-connected binary dilation and calculate all different connected components. Each connected component is considered an individual front.
- Filter the labeled image with the undilated binary mask to remove the dilation effect.

The same transformation is applied to the provided weather service fronts. Note that some provided weather service fronts are separate lines in the label file, but end up as a single longer front due to being connected due to the coarser grid used in our analysis.

5.4.4 Front-Object matching

A predicted front F_p is considered to be matched to the weather service label if the median distance of each pixel of F_p to the nearest labeled pixel of the same class in the weather service's label image is less than a detection radius of D. The same is applied vice versa for the weather service fronts compared against the network output. Each class of front can only be matched to pixels of the same class, however each frontal object is matched against the whole set of pixels of the same class, rather than just a single other object.

For the evaluation we define two distinct regions, namely (i) the evaluation region, which is the region out of which we take the fronts we want to match against any other fronts, and (ii) the comparison region, which is the region in which the algorithm checks for possible matches for the fronts within the evaluation region. In our evaluation, the comparison region is the same as the evaluation region with an additional extension of $10°$ in each direction. The advantage of looking for matches within this comparison region instead of the evaluation region, is to reduce false results caused by the crop of the evaluation region: For example, fronts at the edge of the evaluation region may be split into multiple fronts due to the crop skewing the count of individual fronts. Alternatively, a front located at the edge of the evaluation region may be counted as unmatched, because the possible match was cropped out. Using the comparison region we will resolve most of these cases. A sketch of this procedure is shown in Fig. A.1. Note that using this larger region for the matching purposes does not add any fronts to the evaluation nor does it affect the matching radius D. This change only allows each front to better use its search radius D to find possible matches unaffected by input crop.

5.4.5 Critical Success Index calculation

We evaluate the detection quality of our network and the baseline method by calculating the Critical Success Index (CSI) similar to Lagerquist et al. [32]. As ground truth the provided weather service labels of surface fronts are used. We define n_{MWS} as the count of fronts provided by a weather service, that could be matched against the prediction, while n_{WS} is the count of all provided fronts. Similarly, n_{MD} describes the count of all detected fronts, that could be matched against the weather service fronts, while n_D describes the total count of detected fronts. With these values we can then calculate the *Critical Success Index* (CSI), *Probability of Object Detection* (POD), and *Success Rate* (SR) as described in Eq. 5.4, 5.5, and 5.6, respectively.

$$POD = \frac{n_{MWS}}{n_{WS}} \tag{5.4}$$

$$SR = \frac{n_{MD}}{n_D} \tag{5.5}$$

$$CSI = \frac{1}{\frac{1}{POD} + \frac{1}{SR} - 1} \tag{5.6}$$

As mentioned by Lagerquist et al. [32], these measurements are also applied in other scenarios, like the verification of tornado warnings by the NWS [13]. The SR describes the probability that a predicted front corresponds to an actual front from

the labeled dataset, while the POD describes the probability that an actual front is detected by the network. SR and POD could easily be maximized at the cost of the other, by either not predicting anything or classifying each pixel as a front instead. The CSI serves as a measurement that penalizes such degenerate optimizations, as it maximizes only when both values yield good results. Generally speaking, a high CSI score is preferable. Whether it is more important to have a high POD or SR depends on the task at hand and whether it is more important that the detection is more sensitive or more accurate.

6 Results and Evaluation

In this section, we first evaluate the CSI of our network detections against the weather services data and compare the detections from the network to those from the baseline method (Sections 6.1.1 and 6.1.2). We additionally create climatologies for both automatic methods and calculate the Pearson correlation against climatologies created from the weather service data (Section 6.1.3). Secondly, we present further results of our network's output, where we look into physical quantities across the frontal surface to infer physical plausibility of our network's detections. Finally, we evaluate the relation of fronts to extreme precipitation events to highlight a possible scientific application scenario for the presented method (Section 7.1).

6.1 Performance evaluation and comparison against baseline

6.1.1 Front detection quality

In Fig. 6.1 we provide an image showing an example of the network's output compared to the label of the corresponding weather service. The image shows that the network tends to create thin fronts, as desired. The detections also appear to have a generally smoother shape compared to the weather service labels. The general shape of the fronts appears plausible, even though there are disagreements between the detections and labels regarding both the shape and class of fronts. For a better impression of the network's output, we also provide a video supplement showing the network output on a global scale [@8]. Further details are provided in Section A.4.

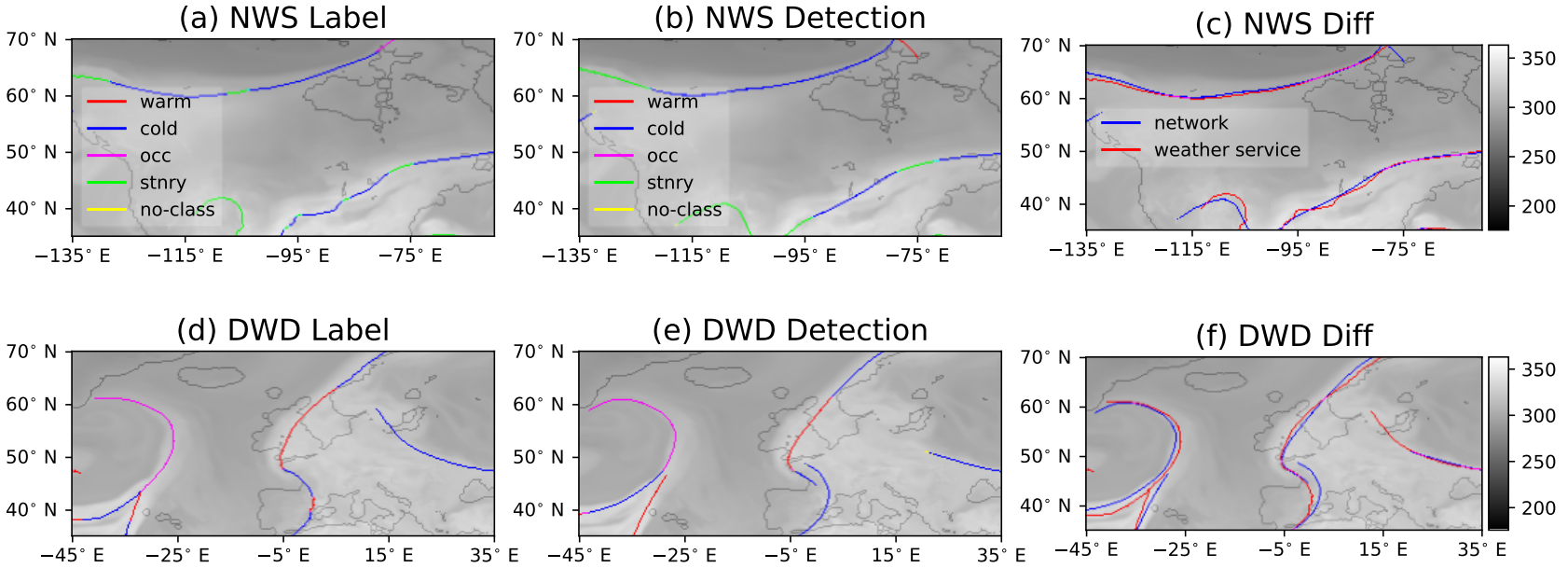


Fig. 6.1.: Fronts from provided labels of the NWS (a) and DWD (d) as well as the corresponding network generated outputs ((b) and (e)) displayed on top of equivalent potential temperature. Colors indicate the frontal type, whereas unclassified fronts are displayed yellow. The labels are the same for both rows. The difference images (c, NWS) and (f, DWD) show a direct comparison of frontal placement by the weather service (red) and the network (blue) ignoring classification. All displayed examples are at 14 September 2016, 00:00:00 UTC.

To quantify the quality of our predictions, we evaluate the CSI, POD, and SR for a matching radius of $D = 250\ \mathrm{km}$ on our test dataset. The results are listed in Tables 6.1 and 6.2 for the binary task, which only considers the classes front and no-front, as well as the individual scores for each of the four frontal classes. As evaluation region we use the corresponding weather services output region as defined in Tab. 5.2.

The scores show that the network excels at the pure front detection task, with CSI scores of 66.9% (DWD) or 68.3% (NWS). At the same time, the network evaluates with a POD and SR exceeding 77.3%. POD tends to be higher than SR for the NWS data, while on the DWD data SR tends to be higher than POD. Considering individual front classes, the classification scores are overall lower, with a class CSI ranging between 36.4% and 56.8%. Across all tests, warm and stationary fronts appear to be harder to classify for the network than cold fronts or occlusions. This effect is more pronounced on the NWS dataset. A possible explanation is the lack of a clear distinction of these two front classes from the DWD data, which in return leads to more false classifications due to the ambiguity. We can further see that training on a single region does not provide a good generalization onto the other region, which is expressed by a lower CSI scores when training on only the DWD (NWS) data and evaluating on the respective other region, i.e., NWS (DWD) data. At the same time, training on both regions yields comparable scores as the networks trained on a single region. This clearly shows that using the network trained on both regions is preferable. We will therefore continue our evaluation with only this model. The difference between the regions may be originating in different synoptic structures of cyclones and their associated fronts over the North American continent and over the North Atlantic. This implies that the inclusion of further datasets, for example datasets used by Matsuoka et al. [38] or generally data of the Southern Hemisphere, may improve the network performance even further. This would also be interesting with regard to a thorough evaluation of the network performance on the Southern Hemisphere. We want to point out here that the inclusion of additional training data of similar structure than the used NWS/DWD data can be carried out easily, the method is designed to be very flexible.

Tab. 6.1.: CSI, POD and SR values for $D = 250\ \mathrm{km}$ evaluated on DWD data for 2016. Warm fronts tend to be detected worse than the other classes, while cold fronts are generally well detected. Stationary fronts are not available for DWD labels and are therefore not listed. Evaluation regions contain latitudes within $]35°, 70°]N$.

Training region	NWS			DWD			Both		
	CSI	POD	SR	CSI	POD	SR	CSI	POD	SR
Binary	51.1 %	65.4 %	70.1 %	68.4 %	78.7 %	84.0 %	66.9 %	77.3 %	83.2 %
Warm	20.3 %	22.8 %	65.1 %	49.3 %	58.1 %	76.6 %	49.2 %	57.6 %	77.0 %
Cold	39.5 %	47.9 %	69.2 %	56.6 %	67.8 %	77.3 %	56.1 %	66.3 %	78.5 %
Occlusion	35.4 %	44.0 %	64.6 %	51.9 %	69.5 %	67.3 %	52.4 %	67.2 %	70.3 %

Tab. 6.2.: CSI, POD and SR values for $D = 250\ \mathrm{km}$ evaluated on the NWS data 2016. Warm fronts tend to be detected worse than the other classes, while cold fronts are generally well detected. The network trained purely on DWD data could not learn stationary fronts, as they are not included in the training data and stationary fronts are therefore not listed. Evaluation regions contain latitudes within $]35°, 70°]N$.

Training region	NWS			DWD			Both		
	CSI	POD	SR	CSI	POD	SR	CSI	POD	SR
Binary	67.3 %	81.9 %	79.1 %	49.7 %	57.0 %	79.6 %	68.3 %	83.4 %	79.1 %
Warm	37.3 %	56.5 %	52.4 %	22.5 %	44.1 %	31.6 %	36.4 %	58.1 %	49.3 %
Cold	55.6 %	70.1 %	73.0 %	41.2 %	51.8 %	66.8 %	56.8 %	73.1 %	71.8 %
Occlusion	48.7 %	72.5 %	59.8 %	36.1 %	62.7 %	46.0 %	49.0 %	73.4 %	59.5 %
Stationary	44.6 %	59.4 %	64.1 %		–		43.2 %	56.2 %	65.2 %

We also evaluated results where each object can only be matched against a single object of the corresponding class, instead of the whole set. The resulting scores are listed in Tables A.1 and A.2. We observe a drop in POD from 77.3% (83.4%) to 70.8% (76.9%) when evaluating on DWD (NWS) data, while SR barely changes. This indicates that our network tends to not fully cover long frontal regions with a single front but rather multiple smaller, disjoint fronts. Each of these can still be matched with the long front, but the long front cannot be matched with any one of them due to their insufficient length, leading to the lower object detection rate. Interestingly, we also do not observe the same change in POD when only considering the classification scores. This further indicates that the previously mentioned fragmentation does not occur within the individual classes but rather at the transition between classes. When the weather service labels several fronts of different classes as connected, the generation of the binary label merges all these fronts into a single long front. If the network then is able to detect the individual fronts, but does not detect them as connected, the conversion to the binary detection will result in several shorter fragments instead. A similar effect may occur if some parts of the long front are simply not detected at all. However, the low change in the classification scores indicates that the first effect is more pronounced. In the bottom row of Fig. 6.1 an example of such a fragmentation can be seen, where the network detects the central front as two separate fronts, while the provided label is a single, connected front. Using the initially introduced matching method, where each front can be matched with the whole set of a class, the fragmentation problem can be overcome. At the same time SR and classification scores are barely affected which shows that this method is suitable for our task.

6.1.2 Comparison against baseline

We additionally evaluated the CSI score on a coarser $0.5°$ resolution grid and compare the results against the baseline algorithm evaluated on the same grid. The used baseline does not classify its results, which is why we only display and compare the task of front detection and forgo any classification results. Due to the previously mentioned fragmentation issues, we only evaluate the results where each front may be matched against the complete set of fronts rather than just a single front object. The baseline algorithm is only designed for application in the midlatitudes and should not detect stationary fronts. Hence, for this comparison, we further restrict our evaluation region to fit within the midlatitudes of the Northern Hemisphere and remove stationary fronts from the labels and network output. There may be an offset between the placement of a front by the baseline and the weather services, as

Tab. 6.3.: Comparison of the CSI, POD and SR of the baseline algorithm against our network for the data of 2016, restricted to the midlatitudes in the Northern Hemisphere. As the baseline algorithm does not classify fronts, we use the binary-classification evaluation for our network. (Quasi-)stationary fronts were removed from the network output as well as the NWS label, because the baseline algorithm should not identify them. For the DWD label, these could not be reliably removed due to the label's ambiguity. We can see that the baseline algorithm is better in predicting fronts in the DWD region than in the NWS region. Evaluation was performed at $D = 250$km for NET and baseline$_{250}$, while $D = 500$km was used for baseline$_{500}$. However, the network performs better in terms of all three measures for both regions.

Method	Evaluation on DWD Region			Evaluation on NWS Region		
	CSI	POD	SR	CSI	POD	SR
baseline$_{250}$	31.2 %	44.4 %	51.2 %	21.9 %	42.7 %	31.1 %
baseline$_{500}$	56.4 %	68.0 %	76.6 %	48.1 %	69.9 %	60.7 %
NET	69.9 %	78.0 %	87.1 %	60.2 %	78.8 %	71.8%

the baseline locates its fronts at the center of a passing front rather than the leading edge. While we believe that the used matching procedure already respects such a difference, we also evaluated the baseline method using $D = 500$ km, i.e. doubling the search radius compared to that used in the evaluation of our network. As shown in Table 6.3 our network (NET) outperforms the baseline algorithm (baseline) in all evaluated scenarios and metrics with a more than twice as high CSI score when using $D = 250$ km. Even when the baseline is evaluated with a larger search radius of $D = 500$ km, the network outperforms it with a difference in CSI scores of more than 10%, even though the network is still evaluated using the smaller search radius of $D = 250$ km.

6.1.3 Comparison of frontal climatologies

To further investigate the soundness of our front detection, we created frontal climatologies for the year 2016 for the provided weather service labels, our network, and the baseline method. While the respective weather services only provide labels within their analysis region, both the network and the baseline can be executed on the entire globe. As in Section 6.1.2 we explicitly remove stationary fronts from both the NWS label dataset and the network output, when creating those climatologies. This is done as the baseline method does not include fronts propagating at less than $3\ \mathrm{m\,s^{-1}}$. The baseline method was designed for application within the midlatitudes and results outside the midlatitudes should be taken with care. We therefore restrict our quantitative evaluation to regions within the midlatitudes. We nonetheless

present the climatology on the global area to emphasize the difference in performance of the network compared to the baseline method outside the midlatitudes. The resulting climatologies are shown in Fig. 6.2.

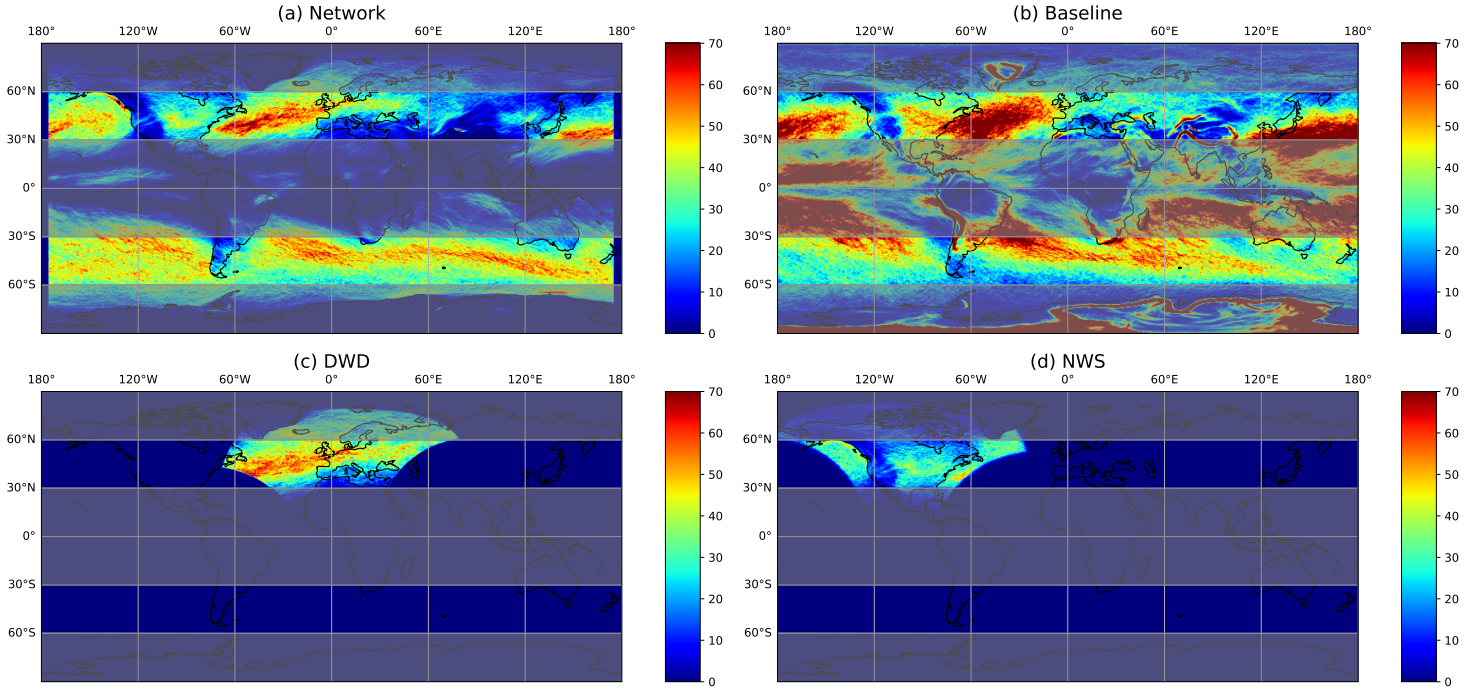


Fig. 6.2.: Global frontal climatologies as derived from the ERA5 data for the year 2016 and climatologies from the weather service datasets. (a) Global frontal climatology from the network executed on the $0.25°$ grid and resampled to $0.5°$ resolution. The network does not provide a valid prediction for the outer $5°$, as the effective output domain is smaller than the input domain. For this reason, no fronts are displayed here. (b) Global frontal climatology of the baseline algorithm. Note that the algorithm is not designed for application outside the midlatitudes and should only be evaluated outside the gray shaded regions. (c) Climatology of the DWD front labels. (d) Climatology of the NWS front labels. The represented front count was clipped at 70 for visual representation, regions with higher front counts are shown in red. Stationary fronts are explicitly excluded from the climatology of network generated data and NWS labeled data. The global climatology from the baseline algorithm does not include fronts propagating at less than $3\ \mathrm{m\,s^{-1}}$. The DWD dataset may include stationary fronts, as we were unable to reliably separate them from warm or cold fronts.

First, we compare the climatology for the North Atlantic / European region from the manually labeled dataset with the climatology of network generated fronts. In the DWD climatology, the North Atlantic storm track is clearly visible as a band of heightened front occurrence stretching from the East coast of North America to the Channel (Fig. 6.2 c). Frontal activity is tampering off inwards of the European west coast. The climatology of the network generated fronts has a very similar overall structure, with a strongly enhanced frontal frequency in the storm track region (Fig. 6.2 a). Frontal frequency is somewhat larger at the beginning of the storm track compared to the DWD climatology. This may be related to the training with North American manual analysis, which naturally has a stronger focus on the early cyclone lifecycle than the European data. Over the Channel and North Sea Coast of Europe, frontal frequency in the network generated dataset is somewhat lower than in the DWD dataset, which may be related to the inclusion of stationary fronts in the latter but not the former. We have seen also in the previous section that very weak warm fronts, as may exist further into the European continent, are often not detected by the network. In both datasets a slightly enhanced frontal frequency around Iceland is evident.

Next, we compare the climatology for the North American region from the manually labeled dataset with the climatology of network generated fronts. The manual labels indicate the onset of the storm track, with enhanced frontal frequencies just off the North American East Coast and secondary peaks in frontal frequencies in the lee of the Rocky Mountains and along the West Coast (Fig. 6.2 d). The climatology of network generated fronts captures all three maxima in the frontal frequency in roughly the same location (Fig. 6.2 a). However, frontal frequency in the lee of the Rocky Mountains and along the West Coast are more pronounced in the network generated climatology. We are under the impression that the network tends to assign labeled warm fronts as stationary and vice versa. These shifts may explain the different frontal frequency.

Finally, we compare the global climatology of network generated front labels to those generated by the baseline algorithm (compare Fig. 6.2 a and b). The striking first difference between the two climatologies is the much larger spatial extent of regions with high frontal frequency in the second dataset. This is evident both in the storm track regions on both hemispheres but also the subtropical regions. In the subtropics, regions of large gradients in equivalent potential temperature exist, and these are picked up by the baseline algorithm. However, their structure and origin differs from fronts in the extratropics. It appears that the network is able to detect this difference in the structure, while focusing solely on equivalent potential temperature and frontal propagation speed is not enough information to differentiate these structures.

In absence of any manual dataset that can serve as ground truth it is difficult to judge the physical meaningfulness of the climatological patterns emerging from either algorithm and indeed in the case of the subtropics may strongly depend on the purpose and definition of what is considered a frontal structure. In the storm track regions on both hemispheres, both datasets show consistently enhanced frontal frequencies over similar geographic regions. They only differ in the zonal extent of the regions with enhanced activity and the absolute values of frontal frequencies. In the only region, where we have an independent, manually generated dataset often considered as the "ground truth", the climatology of network generated fronts is in closer agreement with the former than the climatology from the baseline algorithm. For the Southern Hemisphere or the North Pacific, we currently do not have any such dataset available.

The second striking difference is the high frontal frequency along orographic barriers in the climatology from the baseline algorithm, i.e. along the Andes, Greenland, Himalaya and Antarctic coastline. These maxima in frontal activity are largely absent from the climatology of network generated fronts, consistent with the manually labeled datasets. It appears that the network correctly discriminates between temperature and humidity gradients arising only because of the presence of significant topography and those caused by dynamically generated air mass boundaries. In contrast, focusing solely on the advection speeds in regions of large equivalent potential temperature gradients seems not to suffice.

Overall, the global picture emerging from the extrapolation of the network trained on the North American, North Atlantic and European domain performs well also on a global scale and correctly identifies regions of high frontal activity expected from previous investigations and the known general circulation patterns. While physically plausible, this is of course no vigorous evaluation of the performance of the extrapolation to different regions of the globe. Future work should investigate this aspect in a more quantitative manner with manually labeled datasets from other parts of the globe. However, overall, the investigation of the front climatology agrees well with physically expected patterns and climatologies from manually generated frontal datasets. This lends additional physical credibility to the network generated frontal labels.

A physically plausible global climatological pattern further suggests that the learned frontal identification can be extrapolated from the training region. We found that for this it is necessary to include data from two sufficiently different geographic regions, i.e. North America and North Atlantic / Europe, as well as to augment the dataset by including also zonally mirrored examples of the frontal cases (not shown). The

Tab. 6.4.: Extent of the regions used during comparison of climatologies. These regions correspond to the output regions used during training, limited to $[35°N, 60°N]$.

Weather Service	Latitudes	Longitudes
DWD	$]35°N, 60°N]$	$[-45°E, 35°E[$
NWS	$]35°N, 60°N]$	$[-135°E, -60°E[$

Tab. 6.5.: Pearson correlation coefficient of the climatology computed with the baseline algorithm (baseline) and our trained network (NET) against the climatologies created from the provided labels of the weather services for 2016. The columns denote the weather services, against which the methods were evaluated. Correlations are computed for the midlatitude regions covered by the analysis from the weather services. Stationary fronts were excluded from all climatologies except the DWD labels.

Method	Correlation with DWD	Correlation with NWS
baseline	58.4%	65.7%
NET	79.6%	77.2%

latter was found to be particular important for a good performance in the Southern Hemisphere. This is also visible in the video supplement, where the general shape, composition and motion of fronts detected in the Southern Hemisphere appears plausible. At first, the qualitatively good results on the Southern Hemisphere appear to contradict our claim in the previous section, that training on a single region is insufficient for extrapolation to other regions. However, we believe that this is due to the fact that this region is mostly covered by sea. As a result, there is far less orographic influence in the southern regions. As such, the simple mirroring of data from the North Atlantic may be sufficient to learn a seemingly good model for the sea covered regions of the Southern Hemisphere. Nonetheless, this is only a qualitative observation, that needs to be explicitly evaluated, if appropriate data is available.

To quantify the former qualitative discussion of the climatologies we evaluated the Pearson correlation coefficient of the created climatologies within the regions described in Table 6.4. The resulting correlation coefficients, provided in Table 6.5, show that our network outperforms the baseline algorithm in both regions with correlation coefficients greater than 77.2%. For both regions, the network results are more than 10% higher than those of the baseline. This effect is more pronounced on the DWD dataset, which might be caused by the ambiguity of stationary fronts.

6.2 Variation of physical variables across frontal surfaces

In the previous section we showed that our proposed network can reliably detect fronts as they are provided by the weather services. In this chapter, we evaluate various physical quantities across the detected frontal zone qualitatively, to assess whether the detected fronts express plausible physical features. Since some automatic methods as e.g. the baseline method rely on gradients of certain thermodynamic variables, we investigate these variables for the fronts detected by our network. Thus, we can evaluate if these fronts are detected in a completely different way or feature similar frontal characteristics as those detected by the thermodynamic methods or manual analysis.

For this purpose, we create cross-sections perpendicular to the frontal surface for each pixel that corresponding to a front in 4 steps:

- Estimate the direction normal vector of the front at the given point
- Sample points in the normal direction centered at the given point on the front
- Calculate the mean wind direction along the sampled points
- Use the sign of the dot product of the mean wind direction vector and the normal front vector to sort the sampled points along wind direction

These cross-sections are computed at the $850\ \mathrm{hPa}$ level, since the TFP methods usually are based on variables on this level. For the comparison with the thermodynamic front detection methods, we use the variable equivalent potential temperature (θ_e). Additionally, the variables temperature, relative humidity, and (absolute) wind speed are chosen, showing important features of different front types. These variables are taken from the ERA5 dataset, while the position of the fronts is determined by either our network, the baseline method, or the weather service analyses. We used MetPy v1.0.1 to derive θ_e and the relative humidity [39]. We further used GeoPy v2.2.0 (https://github.com/geopy/geopy) to calculate the position of our sample points.

The mean cross-section for the DWD frontal dataset is presented in Figures 6.3, 6.4 and 6.5. The corresponding plots for the NWS front dataset are shown in the Appendix (Figures A.2 and A.3). In Fig. 6.3 (a) we evaluated the variation in equivalent potential temperature (θ_e) at $850\ \mathrm{hPa}$ based on fronts locations (i) identified by the machine learning algorithm (dashed lines) and (ii) indicated in the surface analysis from the DWD (solid lines). For both front location datasets

θ_e is clearly increasing (decreasing) across the frontal surface for cold (warm) fronts, as would be expected from the physical definition of these features. For the identified cold fronts, the across-frontal temperature variation is on average larger than for the DWD labels. For warm fronts, the across-frontal change in θ_e is similar for both detections, albeit the decrease ahead of the passing front is stronger for the machine learning detections. Warm fronts identified by DWD are on average located at slightly cooler temperatures. This may be explained by the assignment of some warm fronts with weak temperature gradients to the additional category of stationary fronts by our machine learning algorithm, a category non-existent in the DWD dataset. For occluded fronts there is only a small across-frontal variation in θ_e as could be expected and again this is consistent across both datasets.

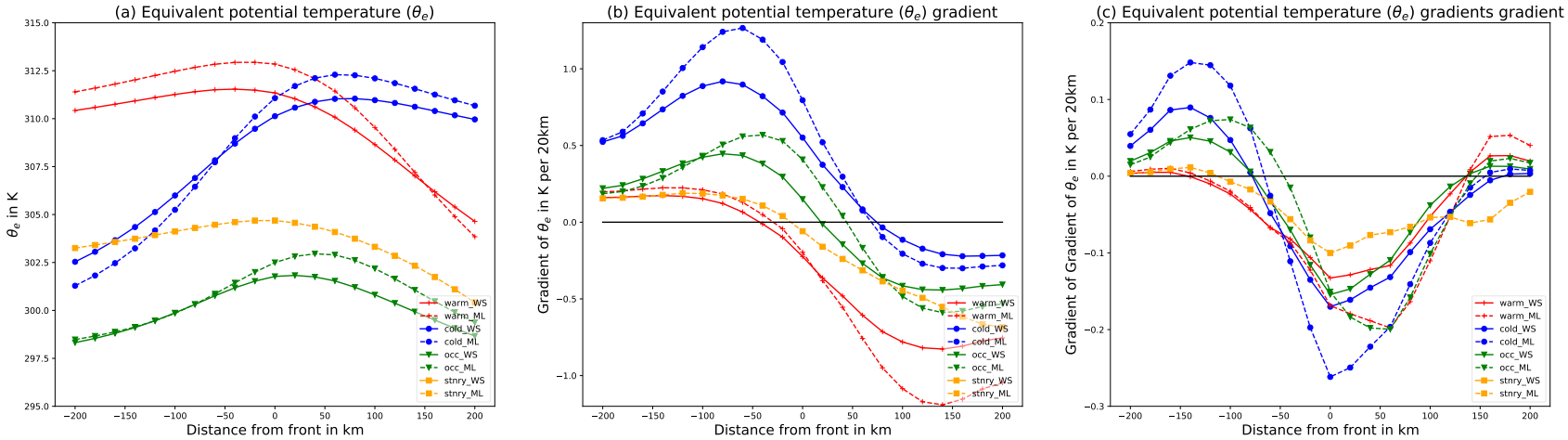


Fig. 6.3.: Average value of variables at 850 hPa across fronts in direction of wind. Mean of (a) equivalent potential temperature (θ_e), (b) θ_e-gradient and (c) gradient of θ_e-gradient with front positions determined by DWD manual analysis (solid, WS) and by our network (dashed, ML). For (b) and (c) we additionally display the 0 level.

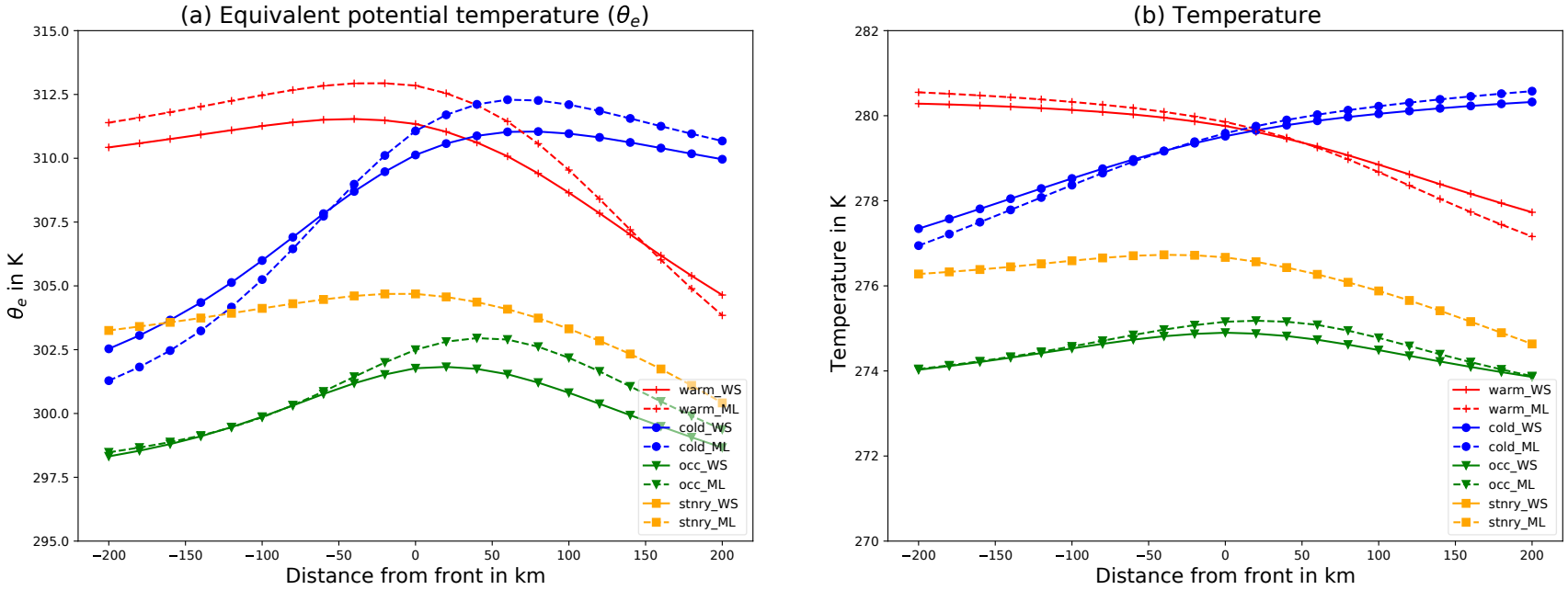


Fig. 6.4.: Average value of variables at $850\ \mathrm{hPa}$ across front in direction of wind. (a) Mean of the equivalent potential temperature (θ_e) and (b) temperature with front positions determined by DWD manual analysis (solid, WS) and by our network (dashed, ML).

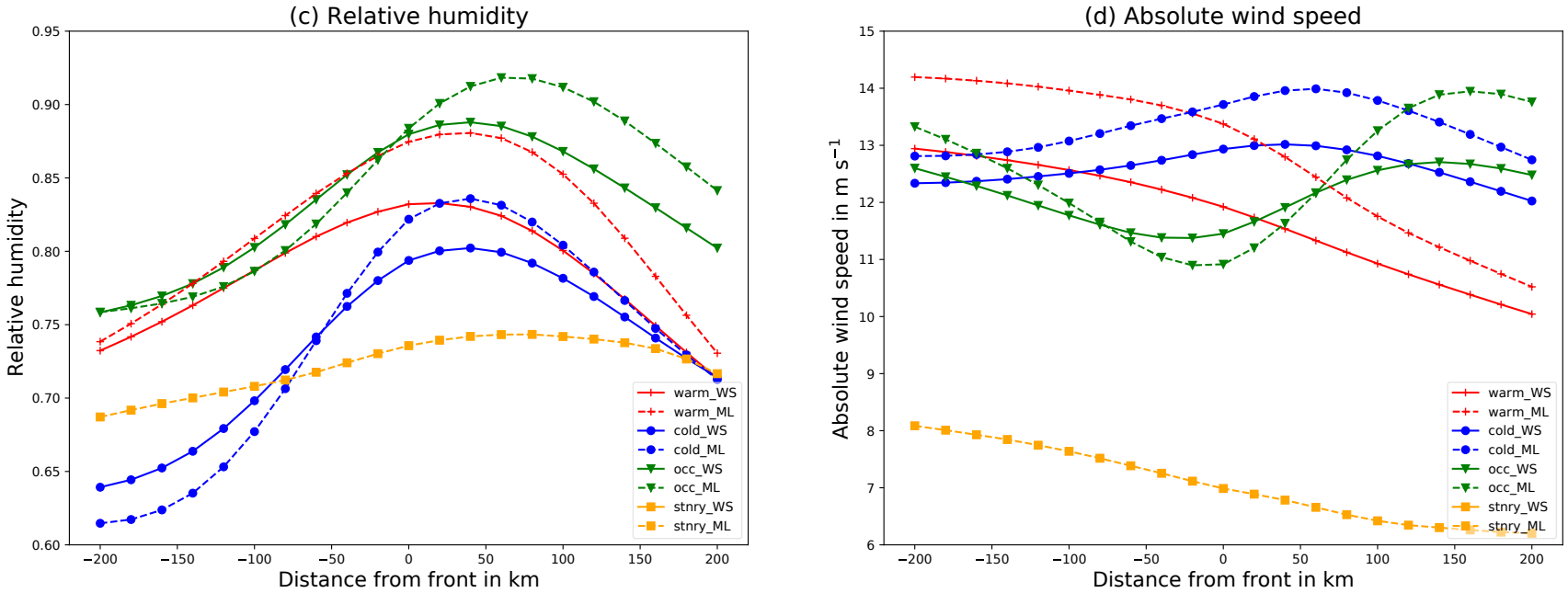


Fig. 6.5.: as Fig. 6.4 but for (c) relative humidity and (d) absolute wind speed.

For most automatic front detection algorithms, the across-frontal θ_e gradient is of importance; this quantity is shown in Fig. 6.3 (b). The θ_e gradient is calculated using finite differences using the sampled temperature cross-sections. Again, we see very similar patterns for both the DWD and our front dataset. In both datasets, the frontal surface is located at the onset of a region with strong change in the horizontal θ_e gradient. This is consistent with the physical definition of frontal zones and agrees with the manually designed automatic front detection algorithms. Generally, the network detected fronts exhibit a stronger gradient compared to those in the weather service analysis for all types of front. Taking the gradient of the θ_e gradient (See Fig. 6.3 (c)) we obtain a magnitude similar to the TFP, where the direction is defined by the normal of our detected front with respect to the wind direction instead of the $2D$ gradient of θ_e. For simplicity, we will refer to it as approximate TFP in the following.

Several conventional methods place the front at the position where the gradient of the TFP is zero. We can clearly see this for the provided DWD labels, where all 3 types of front have a minimum of the approximate TFP at the frontal position. For cold fronts, our network's placement seems to agree with this. For stationary fronts, the signal is less clear, but the front also appears to be located at the extremum of the approximate TFP. Differently, warm fronts and occlusions are placed with an offset of approximately $60\ \mathrm{km}$ to the extremum of the approximate TFP. Nonetheless, we also believe that this offset is reasonable. This shows that both our used labels, but also the network's detections, are plausible with respect to the theoretical background used for TFP methods.

As mentioned before, fronts are typically placed where the gradient of the TFP equals zero, which is thought to describe the leading edge of a front, such as it occurs with the weather service labels. The used baseline method however is different in that regard, as it locates a front where the TFP equals zero, which corresponds to the center of the frontal area. This of course creates an inherent offset in the front's position. Following our evaluation as described above, we can estimate this offset is approximately $130\ \mathrm{km}$ ($80\ \mathrm{km}$) for warm (cold) fronts. Note that both distances are lower than the evaluation distances of $250\ \mathrm{km}$ and $500\ \mathrm{km}$ used for the computation of performance scores in the preceding section. This highlights that the difference in CSI should not be fully accounted for by methodological difference, but rather supports our statement that the network is better at the detection and placement of fronts than the baseline.

In Figures 6.4 and 6.5 we additionally show the temperature (b), relative humidity (c) and absolute wind speed (d) across the frontal zone. The temperature variation

across the frontal zone is quite similar for network and weather service detected fronts and is physically reasonable. For instance, the temperature difference for warm and cold fronts are clearly visible; also, the values agree quite well. For the relative humidity, there are some differences in the absolute values; the network detected fronts have usually enhanced relative humidity values. However, qualitatively, the variation of relative humidity across the frontal zone is well captured. For warm fronts, and also occlusions, there is a pronounced maximum in RH ahead of the front, which indicates the typical frontal cloudiness. A similar signature can be seen for cold fronts, where the maximum is only slightly shifted relative to the surface front position. For the absolute wind speed, we see similar values for the different fronts (detected by network and weather services), but no pronounced structure. Note here, that the mean absolute wind speed for stationary fronts is quite high ($|u| \sim 6 - 8\mathrm{m\,s^{-1}}$) compared to the threshold criterion used by the TFP method. However, the standard deviation is also quite high ($\sigma_u \sim 4\mathrm{m\,s^{-1}}$). A reason for this might be that the position of stationary fronts is not well captured by the network (also because they are only available in the NWS training dataset). Due to the uncertain position, the mean values are smeared out over a large range around the detected position. Nevertheless, the absolute wind speed at stationary fronts is much smaller than the wind speed at the others, which matches with the physical expectation that stationary fronts are moving quite slow - a feature still well captured by the network.

When comparing the frontal zone structure over North America according to NWS labels and our generated labels, generally also consistent structures are found (see Appendix) with deviations mirroring broadly those identified for the DWD data.

Overall, from the good agreement in physical structures across the identified frontal surfaces as detected by our algorithm and from the manual weather service analysis, we conclude that our algorithm detects physically meaningful positions. The positioning of the frontal surfaces is further consistent with physical intuition and interpretation prevalent in literature, and also with the physical constrains for the detection of fronts by an automatic method based on thermodynamic variables.

We can finally remark that even using the surface front as a proxy for the synoptic scale phenomena front (as transition of air masses), the related structures either for the fronts manually determined by the weather services or automatically determined by our network are physically meaningful. This analysis shows that indeed we can use surface fronts as a ground truth for the detection of fronts in reanalysis datasets.

7 Applications

7.1 Correlation to extreme precipitation events

In the previous section, we showed that our model detects fronts in accordance with physical expectations. We further showed that our method generally agrees with the theory of TFP methods, further demonstrating that our model predicts physically plausible fronts. In this chapter, we will further validate our results and at the same time provide an example of how our proposed method may be applied in a scientific context aside from pure front detection for operational weather forecasts. To do this, we evaluate how weather fronts as detected by our network are connected to extreme precipitation. We present the results (i) for the occurrence of extreme precipitation if there is already a front (Sect. 7.1.4), and (ii) for the presence of a front if an extreme event occurs at a grind point (7.1.6).

7.1.1 Data and terminology

Catto and Pfahl [15] previously investigated the co-occurrence of fronts and extreme precipitation using a front detection algorithm based on Thermal Front Parameters (TFP) and the ERA-Interim dataset. Due to the used front detection algorithm, they evaluated their results on a $2.5°$ spatial resolution, and they only use the 6 hourly accumulated precipitation variable of ERA-Interim.

Differently to Catto and Pfahl [15], our front detection can be applied on the $0.25°$ resolution of the ERA5 dataset to provide a more detailed evaluation. Additionally, ERA5 provides data at an hourly interval, allowing us to evaluate at a 6 times higher temporal resolution. Unlike Catto and Pfahl [15], we decided to use the 1 hourly accumulated total precipitation to match the temporal resolution of our data samples. As all evaluation data is taken directly from the ERA5 grid, we do not need to perform any resampling of data. We evaluate the data on a near global region spanning from $[-60°N, 60°N]$ and $[-175°E, 175°E]$. Grid points poleward of $60°$ are excluded as in Catto and Pfahl [15], while the restriction in the longitudinal direction is caused by our network's reduced output domain size. We further mask

regions with high topography ($> 2000\mathrm{m}$) from the evaluation. This filtering mostly removes stationary fronts associated with mountainous terrain.

Extreme precipitation is defined as any precipitation exceeding the $99th$ percentile of precipitation at a given grid point over the considered 9-year period (2010-2018). The correlation of fronts with extreme precipitation events and vice versa is investigated for the year 2016 only. We define that a grid point is considered to be associated with any event (e.g. a front or extreme precipitation) if such an event occurs within a predefined attribution radius. If not explicitly stated otherwise the attribution radius is chosen similar to Catto and Pfahl [15] to be $2.5°$, albeit our attribution radius is a bit more accurate, due to the higher resolution of the ERA5 grid. To decide whether a connection between extreme precipitation and fronts is significant, we conduct a statistical test using statsmodels v0.12.2 [69] for the quantile regression. For our investigations, we adopted the test procedure as described in the study by Pfahl and Wernli [55].

7.1.2 Definitions

For the determination of precipitation events, we use the surface precipitation as contained in the ERA5 dataset (hourly 2D field). Extreme precipitation is defined as any precipitation event that exceeds the $99th$ percentile of precipitation at each grid point. Due to a limitation of available data, we calculate this percentile using ERA5 data ranging from the years 2010 until 2018 (inclusive) using CDO.

We consider any grid point within an L2-distance of $2.5°$ (i.e. 10 grid points) to a front (extreme precipitation event) to be associated with a front (extreme precipitation event). We evaluate the connection between fronts and extreme precipitation using $N = 8784$ samples from the year 2016; we chose this year, as it was not used during the training of our network.

We need to define a few variables for the evaluation. For each grid point p, we define the number $N_{evt}(p)$ of different events evt as the count of an evt occurring at p during 2016. For a grid point p, the counted events are as follows:

- epr: An extreme precipitation event occurred at p
- $a(epr)$: p is associated with an extreme precipitation event. (E.g. p is within a certain range (here $2.5°$) of a grid point where an extreme precipitation event occurred.)
- fr: A front occurred at p

- $a(fr)$: p is associated with a front. (E.g. p is within a certain range (here $2.5°$) of a grid point where a front occurred.)
- $x + y$: Events x and y occur at the same time at p. (e.g. $epr + a(fr)$ describes the event that an extreme precipitation event occurs at p while p is associated with a front.)

.

We further define the proportions $P_{evt}(p) = N_{evt}(p)/N(p)$ for events evt as defined above. Finally, we also calculate the relations

- $R_1(p) = \frac{N_{a(fr)+epr}(p)}{N_{epr}(p)}$, describing the proportion of extreme precipitation events at grid point p that can be associated with a front
- $R_2(p) = \frac{N_{fr+a(epr)}(p)}{N_{fr}(p)}$, describing the proportion of fronts at grid point p that can be associated with an extreme precipitation event.

These definitions are slightly similar to the formulation of conditional probability.

We further define as high altitude regions any grid point within a 5 pixel distance from any grid point exceeding a height of $2000\mathrm{m}$. The height of a grid point is derived from the geopotential height variable of the ERA5 dataset

7.1.3 Statistical test

If we assume that both events, i.e. the occurrence of an extreme precipitation event and a front, are completely uncorrelated, we would expect R_1 to be similarly distributed as $P_{a(fr)}$, i.e. the frequency with which point p is associated with a front. For each grid point p poleward of $20°$, we calculate the frontal frequency $P_{a(fr)}(p)$. We then distribute all points p according to their respective frontal frequency, into bins of 1% width. For each bin with at least m entries, we randomly select m grid points (base points) and create 1000 event lists, each containing k successive extreme precipitation events sampled at 6 points. Each of those points is located at the respective opposite hemisphere from the corresponding base point. k is chosen as 50 such that we obtain at least 300 samples of extreme precipitation events for each base point. As result for each frequency bin we obtain m sampled distributions of the proportion of extreme precipitation events occurring while the base point is associated with a front. Taking the median of each of those samples, we get a sample of m points per bin. We then apply a percentile regression on this data to obtain linear functions describing the $1st$ and $99th$ percentile of our data with respect

to the frontal frequency. We then define that for each grid point p, where $R_1(p)$ is not within the limits described by these percentiles respecting the underlying frontal frequency, a significant connection between extreme precipitation and frontal frequency exists. We are then able to additionally mask all grid points where no significant connection is found.

For our test, $m = 12$ was chosen as the maximum observed frontal frequency was around 53%. Ignoring bins with less than m entries, a total of 576 base points were considered. This test will be used for the evaluation in different scenarios.

7.1.4 Extreme precipitation associated with fronts

In Tables 7.1 and 7.2 the proportion of extreme precipitation events at grid points that can be associated with a front (R_1) is presented for different regions. For comparison with the former work by Catto and Pfahl [15], we report values for the global evaluation, i.e. including the tropics, although the application of front detection methods in these regions remains questionable. In addition, we present a more detailed analysis for different parts of the midlatitudes (Tab. 7.2). We can clearly observe, that a high proportion of extreme precipitation events can be associated with fronts when considering sea covered regions. Filtering out mountainous regions, the correlation between extreme precipitation and fronts increases compared to the full midlatitude dataset. Over flat terrain, the frontal systems can develop in a quasi idealized fashion, thus warm, cold and occlusion fronts can develop quite undisturbed. Thus, extreme precipitation is mostly linked to the large scale features, whereas over (steep) terrain local effects can disturb the frontal development and / or generate extreme precipitation by other processes. This effect also explains why R_1 is higher for the southern midlatitudes or hemisphere compared to their northern counterparts. Further, we can see that R_1 is higher for the midlatitudes than for the tropics for all types except stationary fronts, where we observe the opposite effect. This is expected as it coincides with the frontal frequency at these locations and the presence of other processes generating extreme precipitation, e.g. organized deep convection. While stationary fronts are more often detected near high altitude regions, above land surface, and at the ITCZ, the other types of fronts tend to occur more often over the ocean, e.g. the storm tracks in the Atlantic and Pacific. This is in accordance with the correlations shown in Tabs. 7.1 and 7.2, where we can see the same connections for R_1.

Tab. 7.1.: Average proportion of extreme precipitation events associated with a front for different regions in 2016. Results are shown separately for the entire globe ($[-60°60°]$N), northern and southern hemisphere ($[0°, 60°]$N and S, respectively), and tropics ($[-30°, 30°]$N).

Region	all	warm	cold	occlusion	stationary
global	0.591762	0.207308	0.259069	0.137746	0.152227
northern hemisphere	0.523959	0.158889	0.205674	0.115030	0.176706
southern hemisphere	0.658888	0.255434	0.312175	0.160145	0.127472
tropics	0.419067	0.074942	0.144921	0.023774	0.225288
global land	0.426572	0.097443	0.168555	0.080147	0.186018
global sea	0.665551	0.256384	0.299502	0.163476	0.137133

Tab. 7.2.: Average proportion of extreme precipitation events associated with a front for different regions in 2016 for the midlatitudes ($[30°, 60°]$N and S, respectively).

Region	all	warm	cold	occlusion	stationary
midlatitudes	0.761661	0.337388	0.372848	0.248444	0.080310
northern midlatitudes	0.678892	0.270840	0.311021	0.212936	0.133108
southern midlatitudes	0.843307	0.402863	0.432948	0.284470	0.027839
midlatitudes no mountain	0.780816	0.354091	0.383997	0.260504	0.071064
midlatitudes sea	0.851108	0.415874	0.425085	0.295962	0.029676
midlatitudes land	0.565787	0.165520	0.258460	0.144388	0.191187
midlatitudes land, no mountain	0.596549	0.192130	0.276355	0.167556	0.179444

Figures 7.1 and 7.2 display R_1 for each frontal type at each grid cell. For this plot, all high altitude regions are gray shaded (light gray), while all regions where no significant connections between fronts and extreme precipitation could be found are white shaded. Further, we masked all regions where no extreme precipitation event was found using a dark gray overlay. This occurs since extreme precipitation is defined using all years from 2010 to 2018, while correlations to fronts are only investigated for the year 2016. In some storm track regions over the ocean, more than 90% of all extreme precipitation events can be associated to a front. Overall, extreme precipitation appears to be more often associated with cold fronts than warm fronts. In the northern midlatitudes we can see that extreme precipitation events associated with warm fronts occur farther north than those associated with cold fronts. For occlusions this is even clearer as the highest proportion of extreme precipitation associated with occlusions is found close to $60°N$. For the Southern Hemisphere, a similar tendency can be seen, even though the local maxima in the correlation are not as clearly visible. As previously mentioned, stationary fronts are less often found over midlatitude oceanic regions and therefore unsurprisingly almost no extreme precipitation events are associated with stationary fronts there. In contrast, extreme precipitation events in the tropics, especially at the ITCZ, are more likely to be associated with stationary fronts. Similarly, the eastern parts of North America and land surfaces near the north-eastern Pacific coast of Asia also have a relatively high percentage of extreme precipitation events associated with stationary fronts.

Note that for the tropics, the detection of fronts is quite questionable. However, for comparison with Catto and Pfahl [15] using a TFP front detection method, these regions are included, although front detection methods are generally designed for and therefore applicable in a meaningful way only to the extratropics. Overall, our results are in good agreement with those derived in Catto and Pfahl [15].

7.1.5 Extreme precipitation associated with fronts relative to frontal frequency

In Fig. 7.3 and 7.4 we display R_1 as a function of the frequency of a point being associated with a front ($P_{a(fr)}$) at all. Additionally, we plotted the $1st$ and $99th$ percentile derived from the statistical test as well as the identity as orientation.

We created the box plots by distributing all sample points into $k = 21$ bins. Each bin b_i with $0 \leq i < k, i \in \mathbb{N}$ contains all $R_1(p)$ for all grid points p within the midlatitudes, excluding high altitude regions, where $(i-1) \cdot 5\% < P_{a(fr)}(p) \leq (i) \cdot 5\%$.

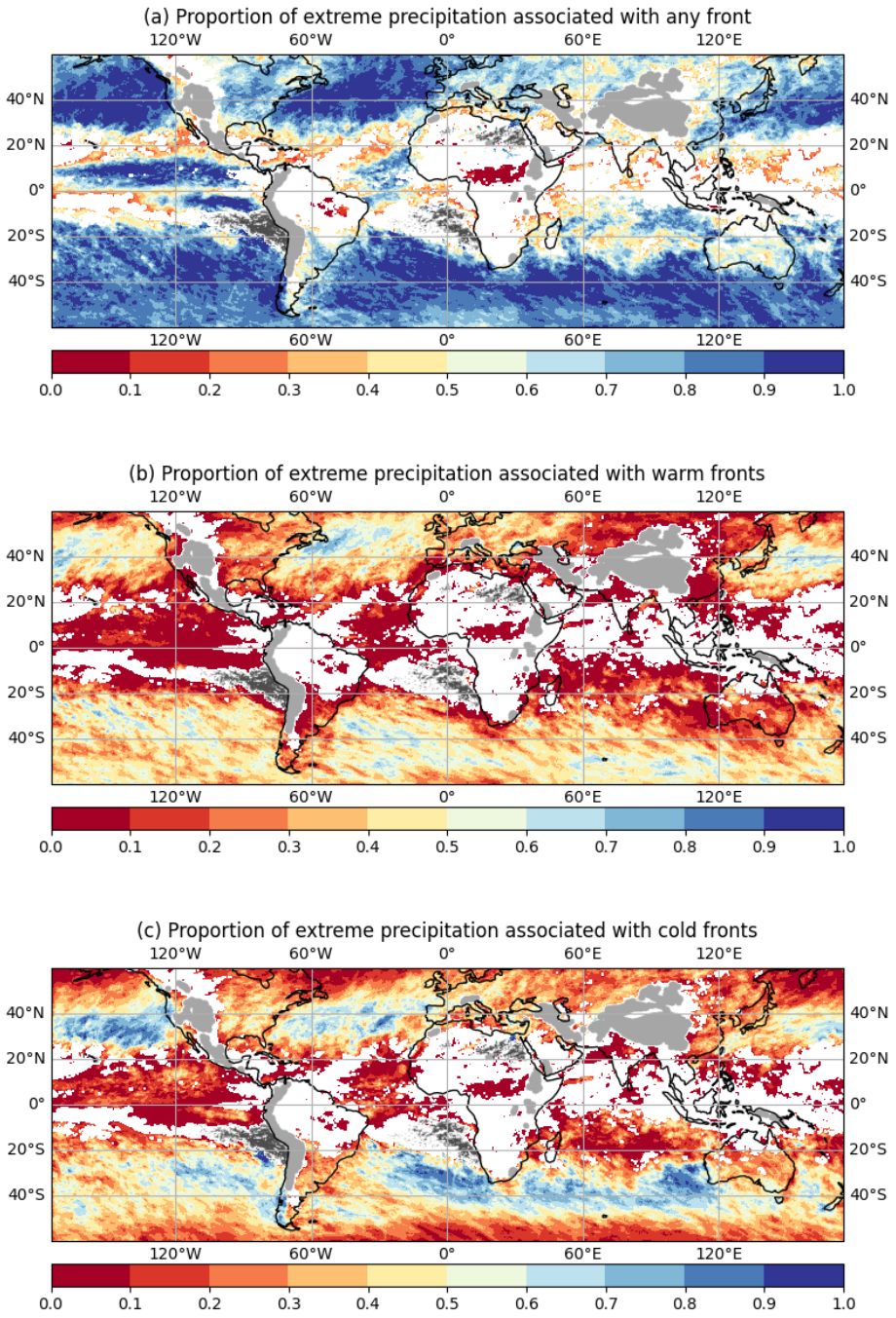


Fig. 7.1.: Proportion of extreme precipitation events, which are associated with a front. Regions with high topography are shaded in light gray, while areas where no extreme precipitation events occurred in 2016 are shaded in dark gray. Regions where no significant correlation between extreme precipitation and fronts was found are blanked. Results are shown for (a) any front, (b) warm fronts, (c) cold fronts

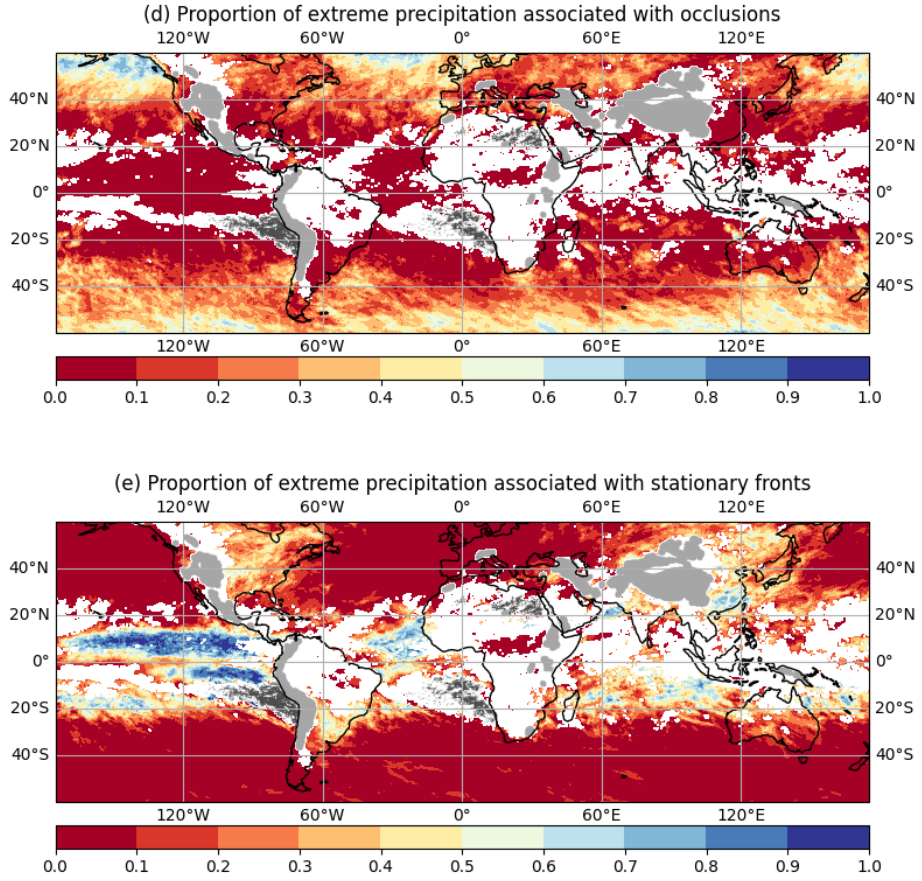


Fig. 7.2.: As Figure 7.1. Shown are (d) occlusions, and (e) stationary fronts.

The lines and box plots can be interpreted as follows: If the box plot is above the $99th$ percentile line, we can conclude that the correlation between extreme precipitation events and fronts is significant in terms of our statistical test.

For warm fronts, cold fronts and occlusions we find that both the median and the mean of each bin exceed the $99th$ percentile even for small front frequencies, i.e. a significant correlation between fronts and extreme precipitation exists. For stationary fronts this appears less clear: Up to 20% frontal frequency the curve connecting the medians indicates a significant correlation between extreme precipitation and stationary fronts, before flattening towards points with larger frontal frequencies. Considering for all types of fronts together (Fig. 7.3 a) the mean and median R_1 exceed the $99th$ percentile for all frontal frequency bins except the largest $P_{a(fr)}$ bin. This clearly indicates a strong connection between fronts and extreme precipitation.

7.1.6 Fronts associated with extreme precipitation

In the previous sections, we have shown that a high percentage of extreme precipitation events are associated with a front. We also found that outside the tropics, this connection is statistically significant according to the performed test. However, we are also interested in the proportion of fronts that are associated to extreme precipitation events (R_2). Similar to Fig. 7.1 and 7.2 we plotted R_2 per grid point in Fig. 7.5 and 7.6. Light gray and white shaded regions are masked as before, while regions where no front of the corresponding type occurred are shaded in dark gray. In general, over large swaths of the midlatitudes more than 40% of fronts are associated with extreme precipitation. Also, regions, where a front is less likely to occur, tend to have a higher percentage of fronts being associated with extreme precipitation. This is very clear for the occlusions: according to the climatology presented, earlier occlusions are predominantly found in the more poleward midlatitude region, but occlusions occurring close to $30°N/S$ are almost always associated with extreme precipitation. The decrease of R_2 for regions with a higher relative frontal frequency (P_{fr}) can at least partially be explained by the definition of extreme precipitation events, as it inherently limits the amount of such events. If P_{fr} exceeds that amount, it is likely that several fronts may not be associated with an extreme precipitation event, even though strong precipitation still occurs. This is somewhat dampened by the fact that for R_2 a grid point with a front only needs to be within the attribution radius to an extreme precipitation event, giving each front several grid points to be associated to. Compared to Catto and Pfahl [15] our results show the same

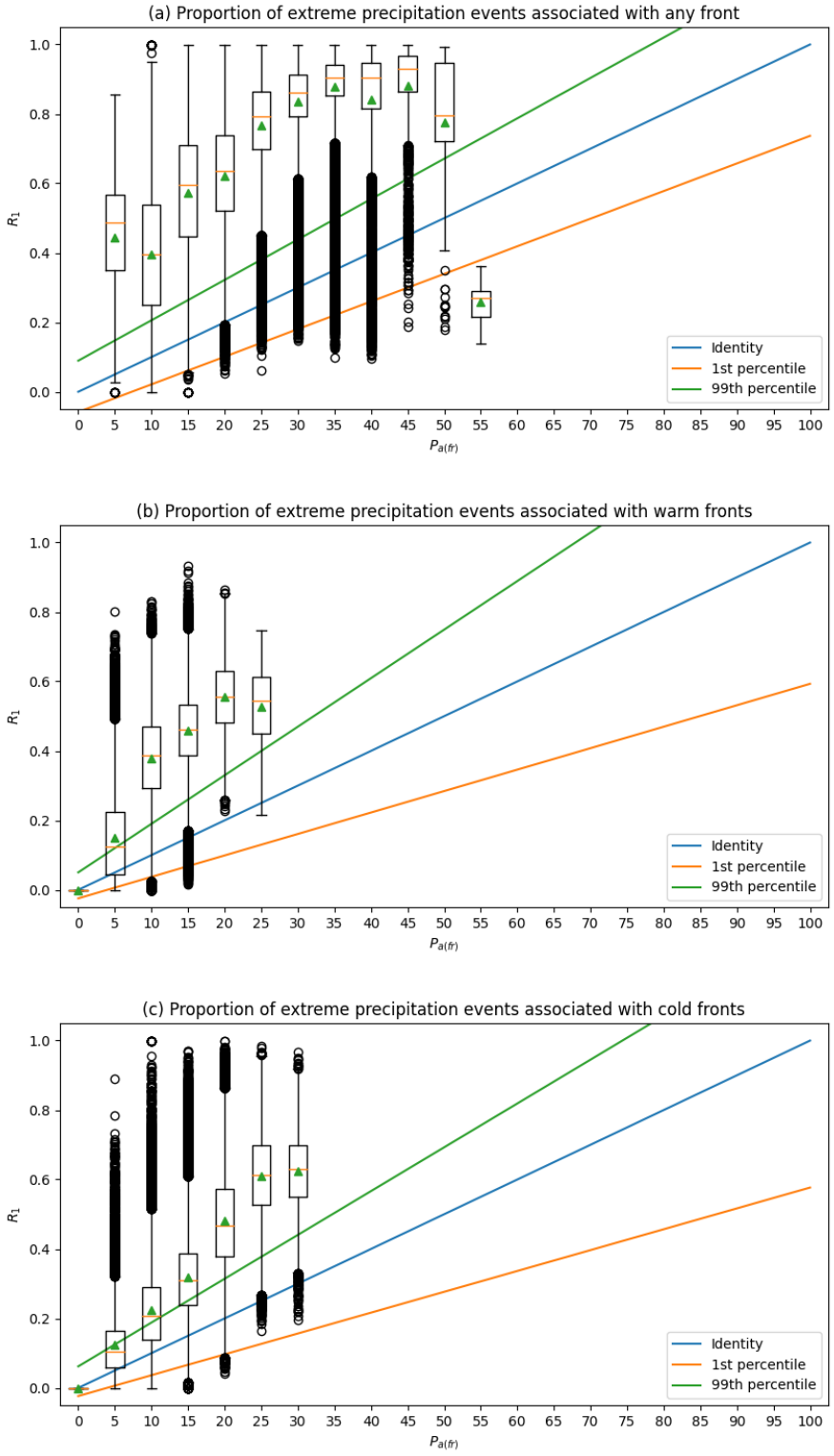


Fig. 7.3.: Fraction of extreme precipitation events grouped by frontal frequency as box plots. Including $1st$ and $99th$ percentile of the statistical test. Results are shown for (a) any front, (b) warm fronts, (c) cold fronts.

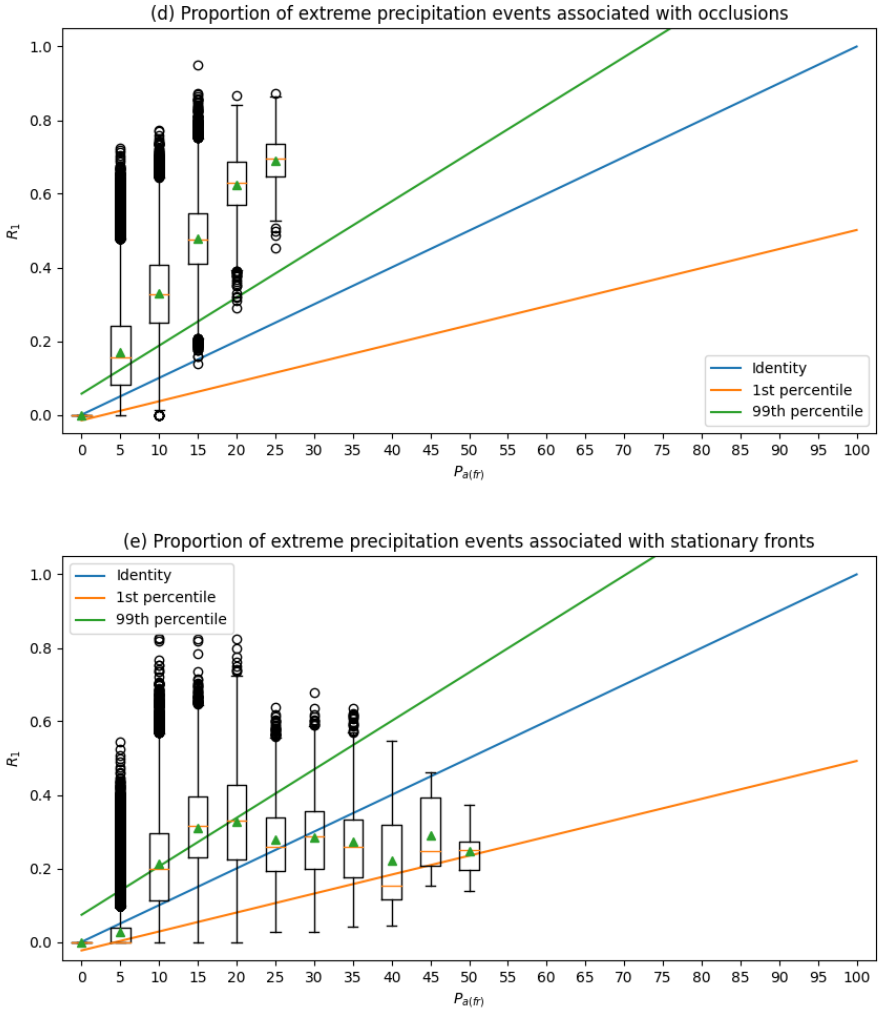


Fig. 7.4.: As Figure 7.3. Shown are (d) occlusions, and (e) stationary fronts.

tendencies, but in our analysis a larger fraction of fronts is associated with extreme precipitation events than in their work.

Overall, our results show a significant connection between extreme precipitation and fronts detected by our network. Our results generally agree with the results of the previous study by Catto and Pfahl [15]. We additionally investigated the correlation between fronts and extreme precipitation on a higher resolution, i.e. for two smaller attribution radii of $5px$ ($1.25°$) and $2px$ ($0.5°$). The qualitative features (i.e. the regions with high correlations) remain the same, but the correlation magnitude is reduced due to the smaller radius of influence. The respective figures can be found in Fig. 7.7. This once again highlights the potential of our network to be used in future scientific research. Such investigations cannot be carried out with classical TFP methods, since they are on a global scale (i.e. using fixed thresholds) most likely restricted to low resolution datasets. This underlines the benefit of our new method over existing ones.

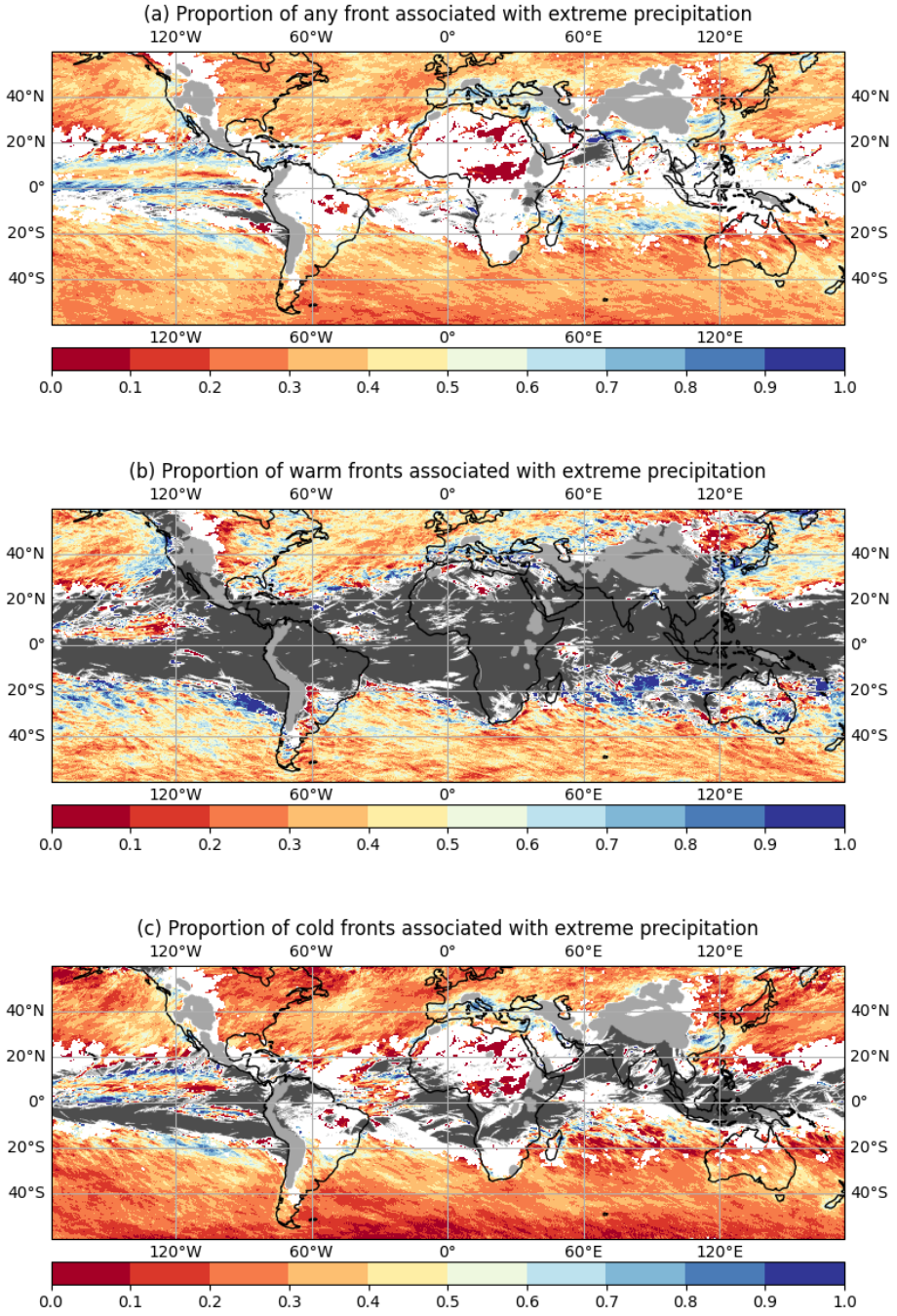


Fig. 7.5.: Proportion of fronts, which are associated with an extreme precipitation event. Regions with high topography are shaded in light gray, while areas where no fronts of the corresponding class were detected in 2016 are shaded in dark gray. Regions where no significant correlation between extreme precipitation and fronts was found are blanked. Results are shown for (a) any front, (b) warm fronts, (c) cold fronts, (d) occlusions, and (e) stationary fronts.

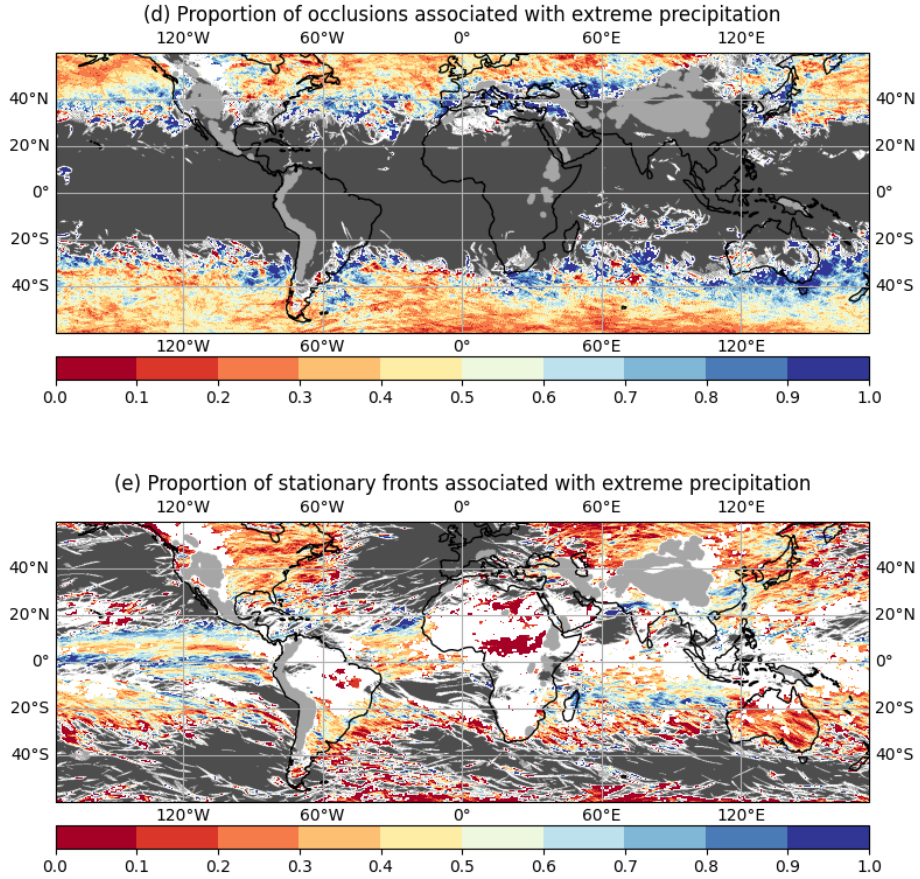


Fig. 7.6.: As Figure 7.5. Shown are (d) occlusions, and (e) stationary fronts.

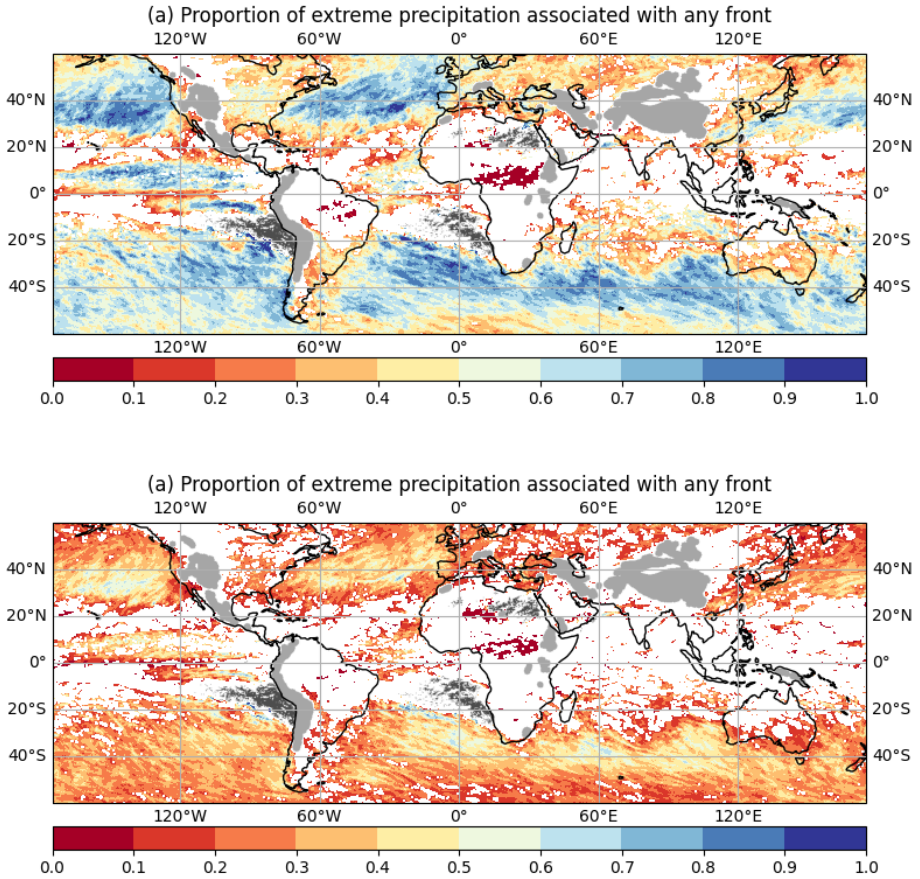


Fig. 7.7.: As Fig. 7.1(a). Proportion of extreme precipitation events, which are also associated with a front, where the association radius is $5px$ ($1.25°$) (a) and $2px$ ($0.5°$) (b), respectively. Regions with high topography are shaded in light gray, while areas where no extreme precipitation events occurred in 2016 are shaded in dark gray. Regions where no significant correlation between extreme precipitation and fronts was found are blanked.

8 Conclusion

Atmospheric fronts are important features, which are usually associated with synoptic scale weather systems. Since fronts are usually connected with significant weather, e.g. clouds and precipitation, and occasionally with extreme precipitation events, they are of high interest for weather forecasts but also in terms of scientific research of such events. While the term front refers to a sharp transition between air masses of different characteristics (e.g. in terms of temperature, humidity etc.), there is unfortunately not a generally accepted definition of a front. This is also reflected in many different approaches to detect fronts automatically, e.g. using (multiple) gradients of thermodynamic variables, or even recently using machine learning techniques.

In this study, we present a new method for automatic front detection based on a neural network, which uses ERA5 reanalysis data. As a ground truth for training and validation, we use surface front data from two different weather services (NWS and DWD) covering significant parts of the Northern Hemisphere; for validation a disjoint subset of this dataset is used. We train the network on a loss function, that allows to classify and predict fronts across the input regions. Our applied loss function results in the network predicting clearly localized fronts without the need of morphological post-processing thinning operations. The network is able to predict fronts with a Critical Success Rate higher than about 66.9%, and an Object Detection Rate higher than about 77%.

For a better evaluation of the quality of the method, we compare the network output with a baseline method, which uses a traditional approach operating on thermodynamic variables (TFP approach). For both methods, a climatology of fronts is derived. In this direct comparison, the new method outperforms the baseline method in the direct comparison with the data from the weather services. We can show that we cannot simply transfer a locally trained network onto any other region, but rather need to train on several datasets to obtain a reliable general front detection. The climatology results indicate that a transfer to oceanic regions may be feasible, however this has to be evaluated in future research. It is also desirable to further investigate up to which degree extrapolation onto different regions is

possible, and to investigate whether or not generalization onto global data is possible from just a few subregions.

The evaluation of physical properties relative to the network detected fronts shows that our detected fronts generally exhibit similar properties as those usually looked for in classical methods. As an example, gradients in the equivalent potential temperature are shown. In addition, a similar quantity as for classical TFP methods is determined from equivalent potential temperature. In the comparison of these quantities relative to fronts determined by the weather services and detected by the network, respectively, we find very good agreement; in addition, they exhibit the same features as would be detected by a TFP method. This also shows, that our ground truth data, surface fronts originating from two weather services, is a suitable choice; although surface fronts are detected, they show the correct structure in terms of thermodynamic variables. Thus, surface fronts can serve as a proxy for the detection of fronts, however, our analysis shows that the resulting fronts are meaningful.

In a final application, we investigate the connection of fronts with extreme precipitation events. This investigation is guided by the former investigation by Catto and Pfahl [15]; however, our network allows us to fully use the available resolution of ERA5 and to investigate characteristics of fronts at a high spatial and temporal resolution, leading to a more detailed investigation. For the midlatitudes the connection between extreme precipitation events and front occurrence is found to be most prominent, with the strongest correlation over flat terrain, especially over the ocean. This application shows that our new front detection method is not only just a tool for operational weather forecasting, but is also useful for scientific investigations. Since the method can be applied on high resolution data, this is a clear benefit of the new method over existing TFP methods, which are usually restricted to low resolution dataset or heavily rely on smoothing operators. The method is rather flexible, it is quite straightforward to include new training datasets, as e.g. surfaces fronts for the Southern Hemisphere. In addition, there is no principle obstacle for using meteorological datasets with higher resolution as input for the method.

In future work, separating the detection from the classification task may be beneficial, seeing the good detection rates of the presented network in the binary case. We would also like to further explore the application and effect of other methods to handle the label bias, such as the method described by Acuna et al. [1]. In terms of research in the field of meteorology, we want to apply this method for further research on the connection of frontal systems with other phenomena, e.g. for the

investigation of clouds at different heights around fronts or transport phenomena associated with frontal systems.

8.1 Code availability

The latest code is available at `https://github.com/stnie/FrontDetection` [46]. ERA5 Reanalysis data can be accessed via the ECMWF climate data center. Used NWS frontal label is available at http://doi.org/10.5281/zenodo.2642801 [41]. Our extracted polylines of the DWD data are available at http://doi.org/10.5281/zenodo.5785816 [45].

8.2 Video supplement

A video supplement showing predicted fronts for January 2016 is available at http://doi.org/10.5446/54716 [@8]

Part II

Creation of three-dimensional fronts

9 Overview

Chapter 10, parts of this chapter and parts of Chapters 13 and 14 are based on the following paper. Some text passages and images are directly taken from the mentioned paper. I designed, implemented and evaluated the software and wrote several parts of the manuscript.

Automated Identification and Location of Three Dimensional Atmospheric Frontal Systems
Stefan Niebler, Bertil Schmidt, Holger Tost, Peter Spichtinger
First published in Lecture Notes in Computer Science (ICCS 2023), Volume 14074, 2023, Pages 3-17 by Springer Nature
`https://doi.org/10.1007/978-3-031-36021-3_1`
Reproduced with permission from Springer Nature

Chapters 11 and 12 as well as parts of this chapter and parts of Chapters 13 and 14 are based on the following paper. Some text passages and images are directly taken from the mentioned paper. I designed, implemented and evaluated the software and wrote several parts of the manuscript.

Scalable GPU-Enabled Creation of Three Dimensional Weather Fronts
Stefan Niebler, Bertil Schmidt, Peter Spichtinger, Holger Tost
In: Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '24), 2024, Association for Computing Machinery
`https://doi.org/10.1145/3659914.3659916`

The common depiction of atmospheric fronts is limited to a two-dimensional representation, usually located at either $850\ \mathrm{hPa}$ or surface level. However, a weather front is in fact a three-dimensional structure, which should in return be depicted as such. In this part, we present our novel method, which expands frontal lines provided by the network presented in Part I into frontal surfaces, spanning multiple pressure levels. This part covers two implementations of the mentioned algorithm: A CPU based Python version as well as a GPU based CUDA implementation which notably speeds up the algorithm while also providing algorithmic improvements.

9.1 Introduction and related work

Weather fronts play a vital role in understanding atmospheric processes. Among others, they are connected to several important atmospheric features such as clouds, thunderstorms and wind gusts [16]. They can be linked to several more critical phenomena, including extreme precipitation [15, 44] and cyclones [65, 21]. With the increase of extreme weather phenomena due to climate change [27] it becomes more important to understand the dynamics and processes of these events. Thus, fronts play an important role in the overall scheme due to their connection to several of these phenomena. However, there is no strict physical definition of a front. They are often only described as the boundary of two (often not easily distinguishable) differing air masses.

Fronts are usually depicted on a 2D-spatial grid, showing the global position of a front at the surface level [44, 29]. While this depiction can be sufficient to satisfy the information about the current location of a front, it neglects the fact that a front is actually a three-dimensional, inclined structure instead of just a line in 2D-space.

Previous front detection tools are limited to the identification of two-dimensional fronts, located at a pressure level of e.g. $850\ \mathrm{hPa}$. Many numerical approaches rely on filtering the two-dimensional grid using derivations of thermodynamic variables such as the equivalent potential temperature (θ_e) to detect fronts [29, 64]. They are often based on properties of the thermal gradient and are extended by adding (handcrafted) filters and thresholds to improve the accuracy of the detected fronts [20]. They often suffer from the dependence on their thresholds, which is typically accompanied by the need for parameter tuning. Furthermore, this tuning is often specific for a certain resolution, whereas application on another resolution may need re-tuning. For high resolutions, the algorithm may even fail to correctly identify fronts. As a means to reduce this effect, some methods smooth their grid data. While such approaches allow the algorithm to run on higher resolution data, they also lead to a loss of information and spatial accuracy gained by the higher resolution.

Unlike for the three-dimensional case, various weather services provide $2D$ frontal information over their respective region. This data could be leveraged to create deep learning-based front detection methods. As shown in Part I machine learning (ML) and especially deep learning (DL) has already been applied to the task of front detection. Previous work to detect 2D fronts used random forests [10] and neural networks [38, 32, 7]. In contrast to traditional algorithms, these methods can be trained on high resolution grids and may therefore be directly applied to datasets at this resolution. In our recent work [44], we presented a deep neural

network for detecting and classifying frontal lines directly on ERA5 reanalysis data. The used data is provided as a latitude-longitude (lat-lon) grid at a resolution of $0.25°$ along both horizontal axes. Different to the other DL-based methods, this model directly outputs the identified fronts as thin lines, allowing for high spatial accuracy. Additionally, the model classifies the detected fronts into 4 types *warm*, *cold*, *stationary*, *occluded*. While the evaluation was restricted to areas covering parts of the Northern Hemisphere extending from North America over the Atlantic to Europe, we also showed qualitatively plausible results for other parts of the globe.

However, all of these methods only provide frontal information on a $2D$ grid, similarly to weather charts.

While the ECMWF provides atmospheric grids for several decades, there exists - to the best of our knowledge - no dataset regarding three-dimensional structure of weather fronts. As such, supervised training of a neural network to detect three-dimensional fronts is not directly possible.

To the best of our knowledge, there is only one previous method, which is capable of detecting three-dimensional frontal fields [30, 3]. This method detects fronts by using a numerical approach based on Hewson's front detection method. The method is utilized to identify fronts for certain case studies, which allows them to tune filter weights and thresholds for specific cases. Further, for application on highly resolved datasets, they apply a grid smoothing, similar to several front detection methods in two dimensions. This implementation is limited to the use within the Met.3D [56] framework, which currently makes analysis of the results outside the tool unfeasible.

In this thesis, we propose a new method based on our described DL model from the first Part of this work. We use the 2D fronts inferred by the network and add a new numerical post-processing pipeline, which extends them to frontal surfaces within the three-dimensional ERA5 grid. To avoid processing a complete 3D grid, we also use the information obtained from the network output such as the frontal location and classification for more efficient computation. Furthermore, our model does not rely on fixed thermal thresholds to filter out wrong results, such as the method proposed by Kern et al. [30]. Instead, we use an adaptive threshold to filter false positives per frontal object.

Our method is – to the best of our knowledge – the first to enable statistical analysis of three-dimensional fronts over multiple timestamps. It is able to provide qualitatively good results, which are in agreement with theories regarding the shape and characteristics of a front. As we are not restricted to singular cases, our method can

be a valuable tool in research of front-related weather phenomena. In comparison to previous methods, our algorithm can be directly applied onto the ERA5 grid, without any smoothing, making full use of high-resolution data.

As the amount of data in the atmospheric physics domain continues to grow, (e.g. [23]), efficient processing of the data is important, to make use of the improved spatial and temporal resolution. As such, we also present a GPU implementation of our algorithm for the creation of three-dimensional fronts. For this, we change the representation of $2D$ fronts used as input and provide several improvements of the overall pipeline. We further provide insights in the scaling and overall speedup of the GPU version compared to our CPU based method.

Chapters 10 and 11 will cover the methods of the Python and the GPU implementation, respectively. While there is some overlap in the general ideas, details may vary between both methods. In Chapter 12 we then provide implementation details regarding the parallelization of the algorithm on the GPU. Finally, results for both methods are presented and discussed in Chapter 13.

9.2 Atmospheric data

In this section, we shortly describe the different types of atmospheric data used to create three-dimensional fronts. We provide information for both the CPU and the GPU version.

9.2.1 Atmospheric data for the CPU version

Processing and evaluation is performed on ERA5 reanalysis data using pressure levels ranging from $1000\ \mathrm{hPa}$ (surface) up to $500\ \mathrm{hPa}$ (approx. $5\ \mathrm{km}$ above sea level). For our CPU algorithm, we use data covering the area $[-90, 50]^\circ\, E$ and $[90, 0]^\circ\, N$. As we want to extract wide cross-sections, we only take surface front pixels into account that are located within the region spanning $[-65, 25]^\circ\, E$ and $[65, 25]^\circ\, N$. This region mostly covers the North Atlantic, minimizing influences of orography such as mountains. The area also contains the northern midlatitudes, where frontal systems are commonly located [65]. The used ERA5 data does not mask invalid grid points which would be located within the terrain (e.g. data at $1000\ \mathrm{hPa}$ in Greenland), but rather uses interpolation to provide values for these areas. We do not alter this data except for the calculation of the equivalent potential temperature θ_e, which is a conserved quantity of reversible phase transitions, and

thus well suited for investigations of fronts. It can be calculated from the atmospheric variables (pressure p, temperature T, water vapor mixing ratio q_v). For the CPU implementation, we used the approximation provided by MetPy v1.3.1 [39].

9.2.2 Atmospheric data for the GPU version

As in the Section 9.2.1, we use atmospheric data from the ERA5 reanalysis dataset, represented as a spatial $3D$ grid (latitude $\times$ longitude$\times$ pressure). This grid is also used as the output grid, on which the $3D$ fronts are located. For the GPU implementation, we still use the output created by the network, presented in Part I of this thesis. However, we use a different, more compact representation of the fronts (see Section 9.3.2). Unless otherwise stated, we restrict ourselves to data within the rectangular grid of extent $[-90, 50]^\circ E$ and $[90, 0]^\circ N$. The pressure levels are the same as for the CPU version.

Differently to the Python implementation, we calculate θ_e from temperature (T), specific humidity (q), pressure (p) and relative humidity (r) using the approximation formula of Bolton [12] (see Eq. 9.1), with the saturation vapor pressure $e_s(T)$ (Eq. 9.4). The dew point temperature T_d was calculated using the Magnus Formula with constants $b = 17.67$ and $c = 243.5$ [34] (Eq. 9.5).

In the case of $r \leq 0$, which may rarely occur within ERA5, we set T_d to $\lim_{r\to 0^+} T_d = -c$

$$\theta_e = \theta_L \exp\left(\frac{3036\ \mathrm{K}}{T_L} - 1.78\right) q(1 + 0.448q) \tag{9.1}$$

$$\theta_L = T\left(\frac{1000\ \mathrm{hPa}}{p - e_s(T)}\right)^{0.2854}\left(\frac{T}{T_L}\right)^{0.28q} \tag{9.2}$$

$$T_L = \frac{1}{\frac{1}{T_d - 56\ \mathrm{K}} + \frac{\log T/T_d}{800\ \mathrm{K}}} + 56\ \mathrm{K} \tag{9.3}$$

$$e_s = 0.61078 \cdot \exp\left(17.27\frac{T_d}{T_d + 237.3}\right) \cdot 10 \tag{9.4}$$

$$T_d = \frac{c\gamma(T,r)}{b - \gamma(T,r)},\ \text{with}\ \gamma(T,r) = \ln\frac{r}{100} + \frac{bT}{c+T} \tag{9.5}$$

9.3 Frontal data

The CPU and GPU version of our algorithm differ in the processed types of fronts as well as in how these fronts are represented. These differences are highlighted in the remainder of this chapter.

9.3.1 Frontal types

Fronts are distributed into the 4 categories: **warm**, **cold**, **stationary**, **occluded**. Warm and cold fronts have a clear structure consisting of a warm and a cold side. Stationary fronts are mainly defined by their travelling speed and not by their gradient direction. Occluded fronts on the other hand are the result of a cold and a warm front overlapping. Thus, gradient strength is reduced at lower levels, while they are split into two parts at higher altitudes, where the two fronts have not overlapped yet. For the CPU version, we only focused on the three-dimensional structure of cold and warm fronts, even though the network is able to provide surface fronts for all 4 categories [44]. For the GPU version, we additionally evaluated occluded and stationary fronts.

9.3.2 Surface front data

To create the three-dimensional fronts we rely on surface fronts, as a prior to locate the start of $3D$ calculations. This provides us several benefits. We can assume the presence of a front at the location with high confidence. In addition, we also have a first estimate of the frontal shape. We use two different representations of frontal data between our implementations. A dense grid representation used for the CPU version and a sparser polyline (or polygonal chain) representation used for the GPU version. We will now briefly highlight the differences between these two representations.

Grid representation Storing the frontal information on a $2D$ grid, as before, is convenient, as the frontal point locations can be directly positioned onto the atmospheric grid. In return, no further preprocessing of the frontal data is necessary and the sampling of each front is sufficiently dense near the level, at which the fronts were detected. Nevertheless, it suffers from several drawbacks. First, calculating the normal direction is error-prone, especially near the edges of a front. Second, as our gridded frontal data does not contain any information, regarding the frontal

orientation, the normal has to be oriented based solely on the underlying atmospheric field. In our case, we used the wind field. As a result, the normal orientation could be inconsistent across a single frontal object, if this approximation is insufficient. This may occur if the wind is (near) perpendicular to the normal of the front. Furthermore, fronts are relatively sparse with respect to the ERA5 grid. Thus, the provided data grid is unnecessarily large, as several zero pixel need to be saved. Accordingly, this complete grid needs to be read during processing, which induces unnecessary IO overhead.

Polyline representation To resolve the problems of the gridded representation, we have converted the $2D$ frontal grid into polylines, which we save in a human-readable text file similar to the CSB data provided by the NWS [41]. Our trained neural network provides the front as one-dimensional lines rather than two-dimensional surfaces and contains no further symbols such as the images of the DWD. Frontal objects are hereby determined as connected components. Similar to the algorithm in Section 5.1.3, vertices are sampled along the frontal objects and sorted to form a line. Each object is then stored using an identifier, describing the type of front, as well as a list of global latitude and longitude coordinate pairs representing each vertex. As the data is obtained from the ERA5 grid, coordinates are only stored with two decimals.

This representation solves the previously mentioned problems. As each frontal object is provided as a line, we can easily calculate a consistently oriented normal for the complete object. Further, the normal can be algebraically determined for each line segment. Regarding the file size, storing global frontal data for 2016 at an 1 hour resolution takes only $605\ \mathrm{MB}$. As a comparison to the gridded data, storing frontal information as $8\ \mathrm{bit}$ integers needs $42\ \mathrm{GB}$ instead. Additionally, the polyline representation is scale invariant, in a sense that we can draw the frontal line without the need of up- or downsampling steps. At the same time, the format preserves frontal object information. A drawback of this representation is, that the data needs to be converted from its textual form into a suitable representation, which can be used together with the gridded atmospheric data.

10 Three-dimensional fronts on the CPU

Abstract We present a novel method to identify and locate weather fronts at various pressure levels to create a three-dimensional structure using weather data located in the North Atlantic. We provide statistical evaluations regarding the slope and weather phenomena correlated to the identified three-dimensional structure. Our approach is based on a deep neural network to locate 2D surface fronts first, which are then used as an initialization to extend them to various height levels. We show that our method is able to detect frontal locations between $500\ \mathrm{hPa}$ and $1000\ \mathrm{hPa}$.

10.1 General

Our pipeline consists of 4 stages processing the detected 2D surface fronts provided by the deep neural network. For each pixel in the surface front, we calculate the normal direction and extract a cross-section of the underlying thermal field along this vector. We evaluate each point of a cross-section and perform local optimization to find the positions of the front at each processed pressure level within each cross-section. The workflow is illustrated in Figure 10.1 and explained in more detail in the following sections.

10.2 Surface front detection

Prior to our new pipeline, we locate and classify surface fronts within the ERA5 data grid. For this task, we use our neural network described in [44]. The network was trained using the frontal labels provided by two different weather services (DWD, NWS), both of which covering parts of the Northern Atlantic region in their analysis. Our evaluation also shows that the network performs well over the sea covered surface - such as the Northern Atlantic, which we use in this work. The network

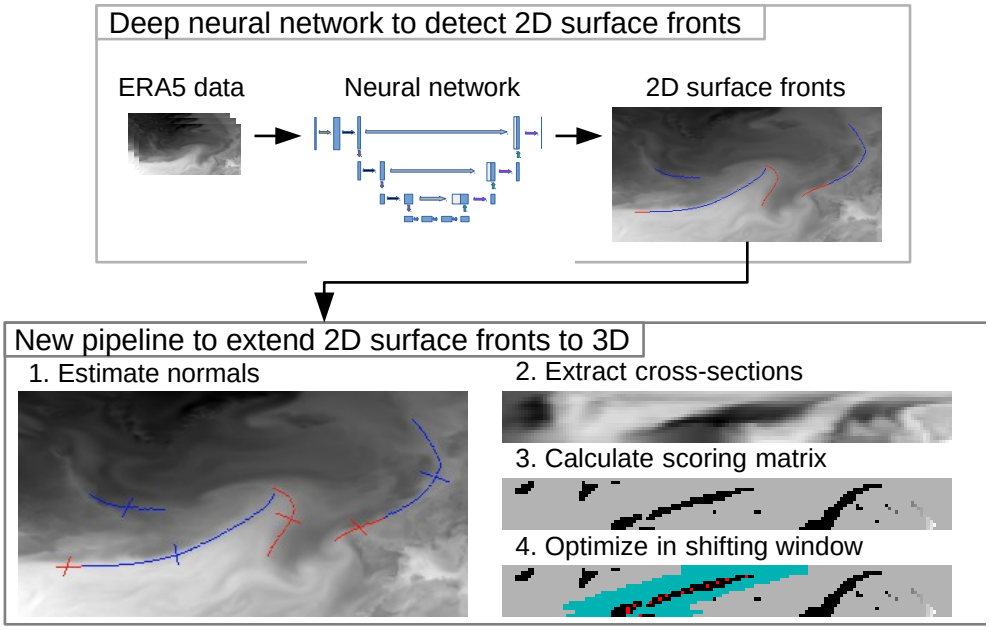


Fig. 10.1.: Top Box: 2D surface fronts are detected from ERA5 data using a deep neural network. Bottom Box: Steps performed in our pipeline for one normal per front: (1) Normal directions of surface fronts are estimated; (2) cross-sections along these normal directions are extracted; (3) each pixel is evaluated, creating a scoring matrix; (4) the minimum within a shifting window (turquoise area) is calculated and returned as front (red dots).

output consists of a multichannel 2D grid where each channel denotes the locations of the different frontal classes (*warm*, *cold*, *stationary*, *occluded*) on the ERA5 grid.

10.3 Cross-sections

Given the locations of the surface fronts for each class, we identify frontal objects by their connectivity and ignore small objects in the following steps. For each front remaining, we extract a cross-section along an estimated normal for each pixel in each determined surface front. Each cross-section contains values for the equivalent potential temperature θ_e, at each pressure level in our dataset, for 100 sample locations before as well as 100 sample locations behind the location of the surface fronts. Sample locations are determined by estimating a normal and then sampling the temperature field at 20 km intervals along the normal direction. As the earth can be approximated by a sphere, we estimate the normal on the tangential plane of each

pixel on a sphere rather than on the flat latitude-longitude grid. Variable values for sampling locations not located directly on a grid point are interpolated. Points where the cross-section exceeds the limits of our input region, as well as cross-sections containing invalid data (e.g. due to an error in the files or interpolation with values outside the data region) are removed. The orientation of each cross-section is chosen such that the sign of the dot product with the wind is positive.

10.3.1 Oversampling

The determined surface fronts may be smaller - in terms of pixel count - than the same front at higher altitudes. For example, a front may extend in a radial fashion where the radius increases with altitude, leading to a larger circumference of the circular sector. This may result in comparatively sparse samples at higher altitudes, even though the lower levels may be densely sampled. To compensate for this, we create multiple samples from each surface front pixel by additionally creating cross-sections with slightly deviated positions and/or angles. For angular deviations of each point, we additionally sample cross-sections where the normal is rotated by $\pm 30°$, resulting in 3 different rotations including the non-rotated cross-sections. Additionally, we sample each of those 3 normal directions at shifted positions. We evaluate a total of 9 shifted positions. These shifted sample points are calculated as the points at a distance of $[-4, -3, ..., 3, 4]$ from the surface front pixel in the direction perpendicular to the non-rotated normal direction. This results in a total sampling of 27 cross-sections per point.

10.3.2 Orientation correction

It is important that cross-sections are correctly oriented, because the algorithm may not correctly process wrongly oriented samples, as they exhibit contrary features such as an inverted sign of the thermal gradient. However, in some cases, the wind may be oriented (near) perpendicular to the front-normal. In these cases, the orientation based on the wind direction may be unreliable, leading to wrongly oriented cross-sections. Oversampling - mainly caused by the rotation - may further add some wrongly oriented samples. As such, we adjust the orientation of samples by taking batches of each frontal object's pixel and reorient each sample within a batch based on their sign of the dot product with the mean direction of the batch.

10.4 Extension to other pressure levels

Once we obtain the 2D cross-section slices for each surface front pixel, we can calculate the front location for each height level within that slice.

In a first step, we flip the orientation of our cold fronts such that the cold region is located right to the center, as it should already be the case for warm fronts. This way we can calculate the positions regardless of the frontal type and potential differences in the sign of the gradients. A front typically bends over its cold region, meaning that the purely warm side has little to no information. We can therefore cut large parts of the warm region located left of the center of each cross-section to reduce computation, without a loss of information. We cut the first 60 pixels of each cross-section, keeping a buffer of 40 pixels left of the center.

To find the frontal location in each cross-section, we first score each pixel of the cross-section to create a score matrix. This score matrix is created evaluating the scoring functions as described in Section 10.5. Once the scoring matrix is calculated, we can determine the frontal structure. Starting from the surface front which we initially locate at the center of the bottom of our cross-section, we crop a smaller region of 40 pixels width centered at the x position of our surface front position and a height (extent in y-coordinate) of $h = 1$ pixel. We call this the current window. The optimal front location is then determined by minimizing a scoring function within the current window. Once we determined h separation points (1 point per pressure level) we shift our window such that it is now centered at the mean x position determined as our front location and covering the next set of h pressure levels. With this windowing approach, we restrict the front detection to a smaller region where the simple optimization is applicable, as we assume that only one front is located within the window. The shifting of the window with increasing altitude allows us to potentially follow the front's inclination across the whole width of each cross-section, without the need to increase the window size. This method may fail if the difference between two succeeding height levels is larger than the distance covered by the window. Overall, our approach essentially reduces the problem of locating a front in a three-dimensional space to a set of 2D local optimization problems. As there is no dependency between cross-section - even of the same object - the method can be easily adapted for parallel processing using multiple threads on GPUs or CPUs.

10.5 Scoring function

The presented algorithm finds the optimal position as the position that optimizes a scoring function at each pressure level. Our scoring function consists of three parts, each being connected to a typical characteristic of weather fronts.

$$L_g = \nabla_x \theta_e \tag{10.1}$$

$$L_e = \left|\left|\left(\mathrm{var}[\theta_e^{i,j+1}, ..., \theta_e^{i,j+k}], \mathrm{var}[\theta_e^{i,j-k}, ..., \theta_e^{i,j-1}]\right)\right|\right|_2, i, j \in \mathbf{N} \tag{10.2}$$

$$L_d = \mathrm{mean}[\theta_e^{i,j+1}, ..., \theta_e^{i,j+k}] - \mathrm{mean}[\theta_e^{i,j-k}, ..., \theta_e^{i,j-1}], i, j \in \mathbf{N} \tag{10.3}$$

where index i indicate the vertical level, j denotes the horizontal grid position, and we set $k = 20$ as half the running window size of 40 respectively. Equation 10.1 describes the horizontal gradient of the temperature field. As we orient each sample such that the cold side is located at higher x indices, the location of the strongest negative gradient is considered to be the ideal position for our front. This is consistent with the classic definition where the front is located at the zero pass of the thermal front parameter, which in our case simply becomes the 2^{nd} derivative of θ_e. We can use an optimization approach for this equation instead of a commonly seen threshold based decision, as

a) we only look at a locally restricted area, where we can assume that the frontal gradient is the most dominant effect

b) due to our initialization, we already know that a front is located here, so we have a lower risk of false positives.

However, we need to respect that fronts may not extend all the way up to $500\ \mathrm{hPa}$. We therefore added a variable threshold that only points where equation 10.1 is among the lowest 10% of all gradients within a front at each pressure level are considered valid locations for a front. As a result, points with very weak gradients are omitted and not seen as fronts. The latter threshold is adaptable to the general surroundings, meaning that it works for both very strong fronts with a strong gradient and weaker fronts where the gradient may be not as dominant.

Equation 10.2 describes our assumption that the air masses before and after a front should be locally consistent, i.e., we do not expect high variation in the temperature within one air mass. As a result, regions with very high temperature variance in any air mass are considered less ideal locations for a front.

Finally, Equation 10.3 describes the fact that the cold air mass is considered to have a lower mean equivalent potential temperature than the warm air mass. Our final scoring function therefore can be described as

$$L = w_g L_g + w_e L_e + w_d L_d \tag{10.4}$$

with positive weights $w_g, w_e, w_d \in \mathbf{R}$. For our evaluations, we set all weights to 1.

10.6 Final processing

Once we calculated the separation point for each height level, we obtained a 1D array consisting of the optimal frontal position for each level. If we combine all these result vectors for all cross-section of a frontal object, we can create a point cloud sampling of the 3D frontal object. If desired, one could use post-processing techniques to create a 2D-surface instead. However, for our purposes point clouds are sufficient as we are interested in statistical evaluation of the resulting fronts. As our initial surface fronts are already classified, we also have a classification for the three-dimensional fronts by simply copying the initial frontal type for all levels.

11 Three-dimensional fronts on the GPU

Abstract Weather fronts play an important role in atmospheric science. Their correlation to severe natural hazards such as extreme precipitation, cyclones or thunderstorms makes localization and understanding of frontal systems an important factor in weather forecasting. Despite their importance, weather fronts are mostly studied on horizontal slices, ignoring their three-dimensional characteristics. In this paper, we present an efficient GPU-based parallelization for the detection of three-dimensional weather fronts. We achieve comparable results to our previous CPU-based method, on which we based our algorithm, while being more than two orders-of-magnitude faster. Furthermore, we extend our previous method by providing additional information for *warm*, *cold*, *occluded*, and *stationary* fronts. Thus, our approach drastically increases the ability to provide statistical evaluations of three-dimensional fronts for different setups. Even faster runtimes can be achieved by using multiple GPUs with linear scaling.

11.1 General

In this section, we will describe the concepts for the calculation of $3D$ fronts in detail and emphasize the conceptual improvements and changes compared to Niebler et al. [48]. The overall pipeline is depicted in Figure 11.1. Our method supports output of the frontal points as a binary file as well as in netCDF format.

11.2 Definitions

We call our atmospheric data an *atmospheric grid*. It is represented as a three-dimensional matrix of shape $h \times w \times b$, where $h \in \mathbb{N}$ corresponds to the number of atmospheric pressure levels. $w \in \mathbb{N}$ and $b \in \mathbb{N}$ describe the spatial extent in terms of latitude and longitude pixels, respectively. A *front* consists of a vector of latitude-longitude coordinates. These coordinates describe a polygonal chain and

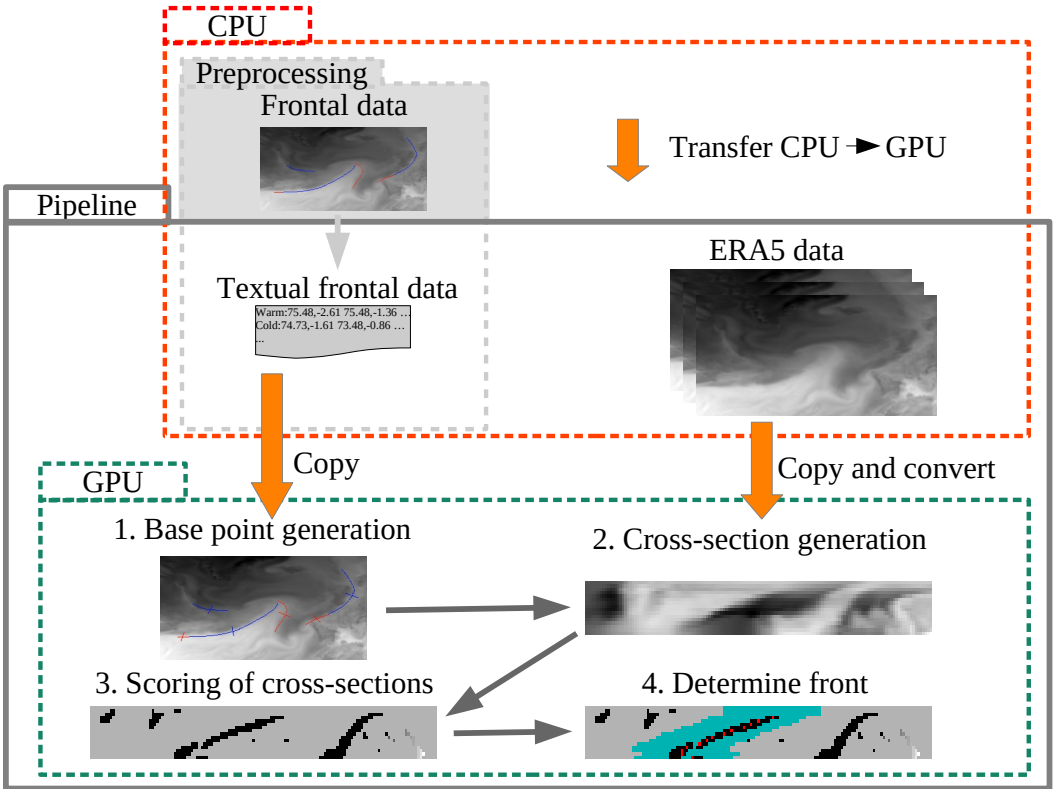


Fig. 11.1.: Our processing pipeline. Preprocessing consists of turning frontal images or grid data to a textual representation. Data is transferred to the GPU where it is processed in four stages. (1) Base points for each front and the corresponding normal of each front are calculated. (2) Cross-sections at each base point are sampled from the atmospheric grid and then scored (3). (4) The frontal surface is determined by a local optimization approach.

will be referred to as *frontal vertices*. A latitude-longitude coordinate, that describes a spatial position, at which we read from the atmospheric grid at any given pressure level, will be called a *sampling coordinate*. We define a *cross-section* as a $2D$ matrix of shape $h \times c$. Each of the h rows is created, reading from the atmospheric grid at $c \in \mathbb{N}$ sampling coordinates for each front. The sampling coordinates of a cross-section are consistent across all height levels. We create additional latitude-longitude coordinates for each front by interpolating between the frontal vertices. We refer to the resulting coordinates as *base points*.

Note that there is a direct mapping from each base point to a corresponding cross-section and vice versa.

11.3 Atmospheric data conversion

For our algorithm we calculate the equivalent potential temperature (θ_e, in Kelvin K), which is also often used for numerical $2D$ front detection, as it provides better features for the determination of fronts [64]. We calculate θ_e as described in Section 9.2 using the approximation formula of Bolton [12] (see Eq. 9.1).

11.4 Base point generation

In our previous work, each pixel of a front was used as a base point, from which a cross-section was calculated. In this work, a front is represented as a polygonal chain, consisting of its frontal vertices v_i. Relying only on these vertices as base points would result in a too sparse representation of the frontal surface. We therefore need to create additional base points. We define the segment connecting vertices v_i and v_{i+1} of the same front as s_i. To create the additional base points, we define an interpolation factor $l \in \mathbb{N}$, which describes the number of base points we create by interpolating each s_i. The q^{th} base point of s_i is calculated as $bc_{i,q} = v_i + \frac{q}{l} \cdot (v_{i+1} - v_i)$. To better handle curvature of a front, we create additional l base points along the quadratic Bézier curve $B_i(t)$ defined by the three vertices (v_i, v_{i+1}, v_{i+2}) of two consecutive segments (s_i, s_{i+1}) belonging to the same front. Further, the parameter t ranges from 0 to 1. We generate these l points equidistantly sampled along the parametrization of the Bézier curve. Therefore, the $(q+l)^{th}$ base point in this case is calculated as $bc_{i,q+l} = B_i\left(\frac{q}{l}\right)$. In the case that no three vertices are available, we use the linear Bézier curve, which is equivalent to linear interpolation.

As each base point can be fully described by either a line segment or a Bézier curve, we can define a normal direction $n_{i,q}$ for each $bc_{i,q}$ as the normal of the corresponding interpolation curve located at $bc_{i,q}$. For both interpolation methods, this normal can be algebraically determined. Additionally, with this approach, we get a consistent normal direction across all segments of a front. For the following sections, we will address to base points and normals with a one-dimensional index bc_j and n_j.

11.5 Cross-section generation

Once all base points and their normal directions are determined, we extract the atmospheric cross-sections for any base point bc_j and its corresponding normal direction n_j, with $j < N$ and N being the number of base points extracted. For a single pressure level p, we create the cross-section $cs_j(p)$ by reading from our atmospheric grid at c sampling points $sp_c(j,i) = bc_j + n_j(i - \frac{c}{2}) \cdot d$, where $d = 20\ \mathrm{km}$ is a parameter defining the step width between sample locations. With this definition, the cross-section is centered around bc_j containing information before and after the front. Values for positions not located at a grid point are bilinearly interpolated, which is a good approximation for the rather smooth θ_e variable. The formula for calculating a sample point is independent of the pressure level and can be applied for each pressure level to obtain the complete cross-section cs_j.

11.6 Orientation correction

For the determination of the optimal frontal surface point, we want each cross-section to face in the direction, such that the thermal gradient across the frontal surface is negative while keeping the orientation consistent across the front. Even though all cross-sections of a front are oriented in the same way; this common orientation may be wrong, as it only depends on the sequence of frontal vertices, not taking into account the underlying atmospheric conditions. To correct the orientation, we calculate a smoothed θ_e gradient at $825\ \mathrm{hPa}$ across the obtained cross-sections. For any cross-section represented by a matrix A, this gradient is calculated as the difference of the mean temperature m pixel or $m \cdot d$ km before and after the front (Eq. 11.1). In this case, p and i correspond to the index of the $825\ \mathrm{hPa}$ level and the central horizontal position of the cross-section, respectively. m was set to 16 for this calculation. If the majority of a front's cross-sections needs to be

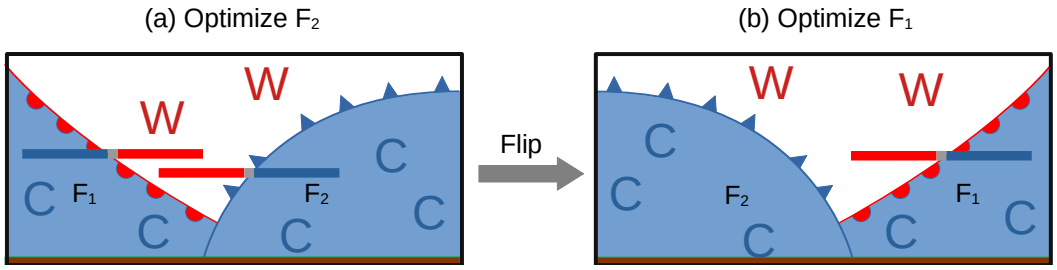


Fig. 11.2.: Sketch of how occluded fronts are extracted. (a) The cold front F_2 is extracted, (b) after flipping the cross-section the warm front F_1 is extracted. Red (warm) and blue (cold) bars indicate the expected temperature.

corrected, this is done for all cross-sections within the front to preserve consistency, else no cross-section is corrected.

$$L(A_{p,i}) = mean(A_{p,i+m}, ..., A_{p,i+1}) - mean(A_{p,i-1}, ..., A_{p,i-m}) \tag{11.1}$$

11.7 Occluded and stationary fronts

In the work of Niebler et al. [48] only warm and cold fronts were extracted. In this study, we additionally work with occluded and (quasi-)stationary fronts. An occlusion or occluded front is defined as an overlapping cold front and warm front [e.g. 67], as sketched in Figure 11.2. We see that the occlusion consists of two fronts F_1 and F_2; thus, we need to create two corresponding frontal surfaces. We now assume that F_1 is the warm front and F_2 the cold front (the opposite case works analogous). If we were to simply apply our $3D$ algorithm to the given sketch panel (a), we would only be able to extract the cold front, as we search the minimal horizontal gradient. To obtain the warm front part as well, we have two options. We could reuse the same cross-section and optimize for the maximum gradient instead. This works, because the horizontal thermal gradient across the frontal surface of F_2 has the opposite sign compared to the gradient across the frontal surface of F_1. Our algorithm assumes that a front exhibits the steepest thermal gradient across its surface. As F_2 already has the steepest loss of temperature across its surface, we can implicitly follow that F_1 should exhibit the steepest rise of temperature across its surface within the current cross-section. While searching for the maximum works for the given scoring function, it is harder to apply this approach to other scoring schemes. Instead, we chose to horizontally flip the cross-section after F_2 is extracted. In this case, the gradient across the surface of F_1 is now negative, which allows us

to apply our optimization code without any alterations to the optimization criterion. With this approach applying other scoring schemes is also straightforward as the problem of finding the surfaces of an occluded front in 3D simply turned into finding the surface of a warm and a cold front, respectively, and can be handled in the same way. The thermal gradient of both fronts may be less dominant at lower levels (Figure 11.2), where both the cold sectors of F_1 and F_2 have already overlapped. Instead of a strong gradient between a warm and a cold air mass as for warm or cold fronts, we are now searching a weak gradient of a cold and a cool air mass instead. As our method does not rely on a thermal threshold to define the existence of a front but rather searches for the minimal gradient, we can potentially still obtain the frontal surface of the front with the colder air mass. In our example, it would be the cold front. For the other frontal part, we cannot reliably predict if the correct position is found, as it depends on whether or not the expected structure of the front is still present or not. Nevertheless, we still provide results for both parts of an occlusion at all levels. In addition to obtaining the location of the frontal surfaces of occlusions, we also get a separation of each occlusion front object into its warm and cold front part, respectively. As for warm or cold fronts, the initial estimation of the normal orientation is dependent on the temperature gradient at $825\ \mathrm{hPa}$. The orientation of this gradient, however, depends on whether the cold sector of the warm or the cold front is more dominant. While for a single occluded front the orientation is chosen consistently such that all warm parts and all cold parts are separated, it is not guaranteed that the normal orientation is consistently chosen across different occlusions.

For completeness, we also provide results for (quasi-)stationary fronts. These fronts are mainly defined by the fact that they move at a speed lower than $5\ \mathrm{kn}$. These fronts are processed in the same fashion as warm or cold fronts.

11.8 Scoring of cross-sections

A common method for determining the position of a weather front is based on thermal front parameter (TFP) [20, 64, 63]. As the direction within our cross-sections is already determined, the TFP simply becomes the second derivative of the thermal field [63]. While the TFP is usually used to determine the warm side edge of a front [75], we are going to determine the center of a frontal zone, which can be located at the zero pass of the TFP [64], which coincides with an extremum of the thermal gradient, i.e. in our case the minimum as we adjusted our cross-section's orientation accordingly. Therefore, the frontal surface at each position

can be obtained by searching the (local) minimum of the thermal gradient within a cross-section. Hence, we score each pixel within a cross-section by its smoothed gradient using Eq. 11.1 with $m = 5$, which corresponds to an area $100\ \mathrm{km}$ ahead and past the given position.

11.9 Optimal path approximation

It may occur that points not belonging to a front contain the minimal score. In this case, the resulting surface may not be consistent. To prevent this we adjust our scores with a dynamic programming approach, where we score each pixel based on the optimal score obtainable, when following a path from the lowest pressure (highest altitude) level to the given pixel. The idea is, that a pixel belonging to a frontal surface has other pixels at lower pressure levels belonging to the same surface in close vicinity, while a pixel not belonging to a front does not. This method is outlined in Algorithm 1. Starting from the lowest pressure level and going towards the surface, let the current pressure level correspond to the p^{th} row of the cross-section cs. We determine the new score of the cross-section at row p and column i as $cs(p, i) + min_\ell(\lambda\ell^2 + cs(p-1, i+\ell))$, where $-k \leq \ell \leq k$ for a given value $k \in \mathbb{N}$. The term $\lambda\ell^2$ is a gap penalty that penalizes large jumps within the frontal surface for neighboring pressure levels. The value $\lambda \in \mathbb{R}$ is a weight, controlling the penalty. Note that the gap penalty is only applied during scoring, not during optimization. We therefore allow jumps, if the new path is scored better.

Data: x and y are GPU thread positions, h is the number of pressure levels, $score$ are the scored cross-sections. λ is a customizable weight.
for $p = 1$ *to* h **do**
  $optScore \leftarrow MAX\_FLOAT$ // set an upper limit ;
  $current \leftarrow score[x, p, y]$;
  for $\ell \leftarrow -k$ *to* $k + 1$ **do**
    $penalty \leftarrow \ell \cdot \ell$;
    $predecessor \leftarrow score[x, p-1, y+\ell]$;
    $optScore \leftarrow min(optScore, current + \lambda \cdot penalty + predecessor)$;
  end
  $scores[x, p, y] \leftarrow optScore$;
  $syncThreads()$;
end

Algorithm 1: Calculating optimal path score

11.10 NetCDF file generation

Our method is able to provide output as a netCDF file containing both the frontal positions and the normal direction extracted during sample generation. The grid is the same as the spatial grid of the atmospheric background data used during processing. As the fronts are sparsely located on that grid, writing the data to disk has a notable effect on the pipeline's performance. A common way to reduce netCDF file size is to pack the data into a smaller datatype. To minimize information loss, the data usually has to be transformed prior to packing. This transformation is described by two parameters add_offset and $scale\_factor$ which are used in the formula $\frac{data - add\_offset}{scale\_factor}$ to pack and unpack data. We chose to pack the resulting single precision floating point data into an 8 bit unsigned integer. As the frontal position is already only represented as a binary flag, whether or not a grid point contains a front, no data loss occurs. For the normal direction a loss is expected as the packed datatype limits the direction to only 256 possible angles. However, as the sampling of normal direction was already relatively coarse, we believe that possible loss of accuracy is acceptable. The values add_offset and $scale\_factor$ need to be chosen for each variable individually, as these are also necessary to extract the original data from the compacted representation. We chose $add\_offset = 0$ for all variables and $scale\_factor = 1$ for the frontal position. For the normal direction, we set $scale\_factor = \frac{2\pi}{255}$. The same values are chosen independent of the type of front.

12 Parallelization on the GPU

This chapter contains details regarding the implementation of the individual stages that were outlined in the previous section. We provide information about memory access patterns as well as parallelization strategies used for the algorithm.

12.1 Overview

Tab. 12.1.: Frontal statistics per type for 09/2016

	#Fronts	Segments per front		
type	mean	mean	min	max
warm	7.8	6.4	1.0	40.0
cold	9.4	8.9	1.0	47.0
occluded	5.6	6.7	1.0	33.0
stationary	10.6	7.0	1.0	53.0

The potential for parallelization of the presented method comes from the fact that each pixel, within each cross-section, can be read and scored independently. Both creation and initial scoring of a cross-section are therefore parallelizable by mapping each GPU thread to a single cross-section pixel. Dynamic programming optimal path approximation as well as the final optimization step exhibit data dependency over pressure levels, although cross-sections belonging to separate base points are still completely independent. Horizontal dependencies for the optimization can also be efficiently parallelized by means of parallel reductions.

In the following section, we will give a short overview about how data on the GPU is formatted before we provide details regarding the individual parts executed on GPU as outlined in Figure 11.1. In a final section, we will briefly discuss our host pipeline parallelization.

12.2 Data representation on GPU

Each front consists of a list of frontal vertices. As we usually have multiple fronts of each type within a given timestamp, we create a list of fronts per type. As the number of vertices per front differs, this is represented as a nested list rather than a two-dimensional matrix. To allow for efficient memory access to all vertices, we represent this nested list as a sparse matrix in a compressed sparse row (CSR) format. As displayed in Figure 12.1, each row of the sparse matrix starts at column 0 and consists of a single dense row segment. Thus, we can omit the column index array of the CSR format, as we only need to know the indices of each row's start to correctly access the data. Access to the atmospheric grid within a timestamp is read-only. Further, most sample positions are not located directly on the grid points, thus interpolation between grid points is necessary. Finally, our memory access pattern is not suited for coalesced access, as it depends on the horizontal orientation of the cross-sections. However, threads of a warp will access data spatially correlated. For these reasons, we store the atmospheric grid using a layered CUDA texture object, where the layers correspond to the pressure levels of the grid. As we have read only access using texture memory is suitable. This provides hardware-accelerated bilinear interpolation and a caching scheme optimized for $2D$ spatial locality. Even though we are working on three-dimensional data, we can utilize the latter, as each pressure level is read independently. Finally, cross-sections are provided as three-dimensional matrices stored in global memory. Memory accesses to cross-sections are mainly coalesced within a warp, except for the optimization kernel, where memory access is only coalesced within a half warp. Output buffers are stored as two-dimensional matrices in global memory as well, containing the results of each base point at each height level.

12.3 Base point creation

This kernel processes all frontal vertices to create all base points and their corresponding normal direction. The overall idea of this kernel is sketched in Figure 12.1.

As stated in the previous section, frontal data consists of a compressed representation of a sparse matrix, where each row contains all vertices belonging to a front. Similarly, the output can be reinterpreted as a sparse $3D$ matrix, where the additional third dimension is of size $2l$, the interpolation factor, while the other dimensions are the same as for the input.

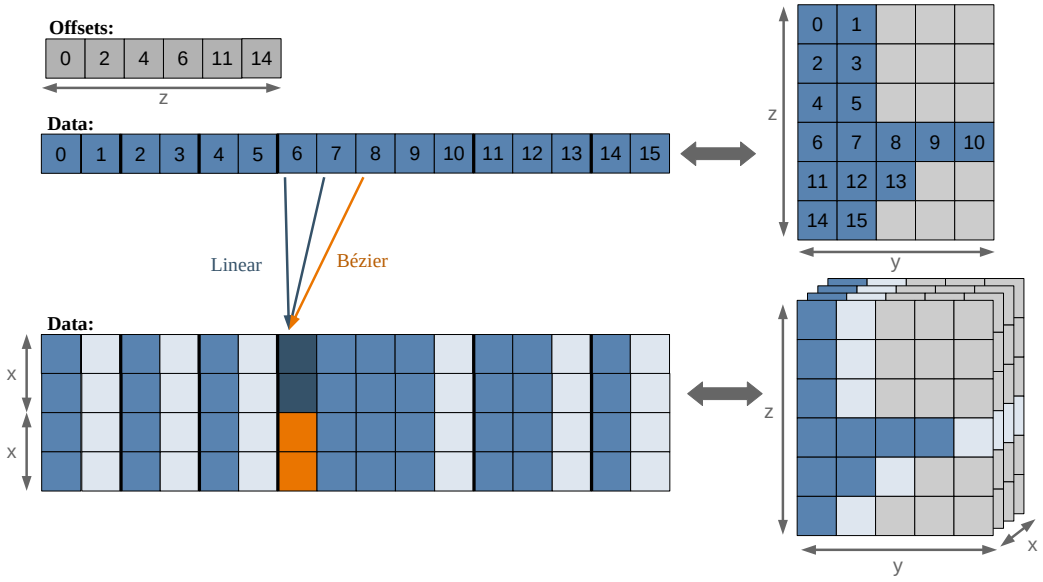


Fig. 12.1.: Sample points are extracted from the frontal objects by linear interpolation between two points (blue) or on a Bézier curve using three vertices (blue and orange). The data can be interpreted as a sparse matrix (top right). The interpolated results can be seen as a sparse $3D$ matrix, where the new dimension equals the number of interpolated points per segment (bottom right). Interpolation is only performed for vertices of the same front. Therefore, the bright boxes are in fact not created. They are displayed for visual purposes only, to clarify the mapping from vertices to interpolated sample points.

As described in Section 11.4 base points are calculated using two interpolation methods. Each thread performs these steps sequentially. To calculate the Bézier curve, we have chosen to implement the algorithm of De Casteljau [11], which gives us a closed form formula for calculating the position as well as the corresponding normal direction for each base point.

The provided output is then written similar to the input as a condensed sparse matrix, where we only keep the values. We do not need to additionally save the row indices, as we can directly calculate these from the row indices of the input by multiplying with $2l$. This kernel provides a total of three output arrays. The first array provides the coordinates for each base point, the second array contains the normal direction in radians for each base point and the final array contains a key, denoting to which front a base point belongs.

During this, kernel threads are arranged in a three-dimensional grid. Within this grid, a thread's z-index denotes the frontal object or the row of the sparse input matrix that the thread processes. The y-index is used to determine the segment of the current frontal object for which base points are interpolated. As the number of vertices per front can greatly vary, launching a thread grid to contain the largest

front can lead us to creating several idle threads. We instead perform a grid stride loop to cover the largest front. For each segment, we create l base points via linear interpolation and l base points by interpolating along the Bézier curve. Each thread's x-index tx is therefore used to calculate the interpolation position within a segment as $\frac{tx}{l}$. We define the block size as $(bx, by, bz) = \left(\min(l, 32), \max\left(1, \frac{32}{\min(l,32)}\right), 16\right)$. The block grid for a given is correspondingly chosen as $\frac{l}{bx}, \frac{mseg}{by}, \frac{num\_vertices}{bz}$, where $mseg$ describes the mean number of segments for all fronts currently processed. The y-dimension of the grid is chosen as a compromise between spawning idle threads and performing many iterations within the grid stride loop.

Memory stores within a warp are sequential, for $l \geq 8$ mostly even coalesced. Loading from global memory also occurs at either the same or sequential memory positions within each warp. While this is not coalesced, memory accesses may be cached. Across warps, memory access is likely to be strided, depending on the sizes of the fronts. As input and outputs are different, no data races occur.

12.4 Cross-section sampling and orientation correction

Once all base points and their normal directions are calculated, we create cross-sections for each base point and adjust their orientation. These steps are split into three consecutive kernels.

Cross-section generation In the first kernel, we calculate the sampling points for each base point to read from the atmospheric grid to create the cross-sections. The array containing all cross-sections has the dimension $(N \times c \times h)$ where c is the number of sampling points per cross-section, h is the number of pressure levels and N corresponds to the number of base points extracted in the previous step. To calculate the cross-section, we spawn a three-dimensional thread grid of (c, h, N) threads, where each thread writes a single pixel in the cross-section. The value of the thread at position (x, y, z) is obtained by reading the atmospheric grid texture at layer y and sampling position $sp_c(z, x)$ as described in Section 11.5. In the same kernel we perform an initial orientation correction, where for each cross-section, we evaluate the smoothed thermal gradient as described in Section 11.6. While the gradient could be calculated from the cross-section, we chose to implement a more general method, where the data used for orientation correction is read from an additional $2D$ texture evaluated at $sp_{32}(z, li)$ for the first half warp or $sp_{32}(z, li + 1)$ for the second half warp within each warp. The value li corresponds to the lane ID of each thread within its warp. The values of the first half warp are multiplied by -1

before using a sum reduction across the values of the warp. If the resulting difference is positive, the cross-section is horizontally flipped by setting the x-position of the corresponding threads to $x \mapsto c - x$. For occlusions, we calculate two versions, where the second version should be oriented in the opposite direction than the first. In this case, the orientation of the cross-section is flipped one additional time. We store the information whether or not a cross-section was flipped to global memory as an array $needsFlip$. As we determine the orientation of the cross-section using warp reduction, we chose the block size to be $(32, 4, 8)$, to ensure that a full warp fits into each block's x-dimension, which allows for coalesced global stores.

Direction adjustment If the normal direction of a front should be provided as output, we need to flip the direction by adding π. As the normal direction is used by several threads during cross-section creation, we perform this step in a separate kernel to ensure that no data race occurs. Within this kernel, we use the information whether or not a cross-section was flipped to determine whether or not the direction needs to be flipped as well. This kernel is trivially parallelizable using a $1D$ grid.

Orientation consistency adjustment In a third kernel we want to ensure that our cross-sections are consistently oriented relative to their initially extracted normal, i.e. either all cross-sections of a front are flipped or none are. We use the array from the Section 12.3 denoting the index of the front which each cross-section belongs to and refer to it as $frontIDs$. We use Thrust's `reduce_by_key` function with $frontIDs$ as the key array and $needsFlip$ as the values to reduce using the sum operator. With the result we can determine for each front if the majority of cross-sections did flip or not and then readjust the other cross-sections of this front accordingly. This is done in a separate kernel using a three-dimensional grid, with the same layout as in the cross-section generation kernel.

12.5 Scoring

Using the same thread arrangement as in Section 12.4 each pixel is individually scored as described in Section 11.8. The corresponding kernel maps each thread to a single pixel. As the output array containing the scores is different from the input cross-sections, no data races occur. Further memory loads are consecutive, while stores are coalesced as the x-dimension of a block is set to 32. As mentioned in Section 11.9 we adjust these scores with a dynamic programming approach. As each row of a cross-section depends on its predecessor and multiple threads access the same locations, data races may occur. To prevent this, we need to synchronize

all threads processing a row once the row is completely calculated. We provide two corresponding implementations. In the first implementation, each row is calculated sequentially within its own kernel launch, which ensures the necessary thread synchronization. As multiple threads access the same read position, we use shared memory to improve repeated memory accesses to the same location. As we cannot process the pressure levels in parallel, only a two-dimensional grid is necessary. Thus, we use a fixed block size of $(32, 16)$, which ensures coalesced memory accesses to global memory. The second implementation uses a single block to process each row. This allows us to loop through all levels within the kernel and use `__syncthreads()` for synchronization within the kernel as well. However, we did not see a notable difference in performance.

12.6 Optimization

Using the optimal path score instead of just the local score for each pixel, does not change the optimization step, where starting from the lowest level, the pixel with the lowest score is searched within a window of size $w = 16$. The initial window-offset wo is placed at $\frac{c}{2} - \frac{w}{2}$ for a given cross-section width of c. As we set our optimization window size to 16, each cross-section can be optimized by a half warp group. All threads of a group read the scores at the current row and at column $wo + li$, where li is the thread's lane ID within the half warp, and calculate the minimal score using a group-wide reduction. We identify all threads whose score equals the minimal score and determine the first thread to be located at the optimal position within the window. From the thread's position and the window offset, we can read the optimal coordinates from a corresponding array. For the next iteration, we may need to adjust wo to ensure that the optimal position can be found within the optimization window. To calculate the new wo, one thread per group writes the optimal offset within its window to shared memory. Subsequently, all groups read those values and calculate the mean offset mo of the thread-block. We then calculate the new offset as $wo = wo + mo - \frac{w}{2}$. During the next iteration, the window will be positioned at this offset. This kernel is executed with a $2D$ thread grid, where the block size is set to $(16, 16)$, as each half warp should process a single window. As the y-blockdim is only 16, the reduction to identify the mean offset is possible.

12.7 IO and calculation pipeline

Processing a single file consists of three stages; reading, processing and writing. We employ 3 CPU threads, one for each of these stages. As we generally process several hundreds of files, it is important to appropriately run these stages in parallel. We implement a pipelining approach to provide such parallelization, where IO and computation can run concurrently. The read thread fills host side buffers with data read from disk. Once these buffers are filled, the processing thread is signalled, such that it can start to process the data. Once these buffers are cleared by the processing thread (e.g. by copying the data to GPU) it signals the reading thread that it can start reading the next. Similarly, the same is applied between the processing and writing thread. This communication is realized by using condition variables and mutexes. Pipelining is also employed within the processing thread, where we make use of asynchronous GPU computing using CUDA streams. Copying data from host to device and vice versa can run asynchronous to the computational kernels. We use CUDA events to synchronize and signal between streams, whenever a buffer is safe to be overwritten or read from. Accordingly, these streams need to wait if the corresponding event has not been recorded.

12.8 Multi-GPU support

In the algorithm, each timestamp is processed independently. As such, splitting the workload across multiple processes is embarrassingly parallel. We added an MPI-based implementation, which automatically splits the workload across all participating processes using a simple block schedule. This parallelization assumes that each participating rank owns its own GPU.

13 Results

In this chapter, we will highlight how both implementations fare in the detection of three-dimensional fronts, and provide a comparison between both methods. Although, it is to note that due to a lack of a ground truth, the comparisons are more focused on how certain expected features are represented within the results of the two methods.

13.1 Setup and evaluation details

We present several applications of our methods and show results for the evaluated dataset consisting of hourly reanalysis data for 2016. Due to a lack of a ground truth dataset of annotated 3D fronts, we can only provide qualitative evaluations. These evaluations show that the characteristics of our three-dimensional fronts are in accordance with classical theoretical knowledge surrounding fronts and their expected behavior. Further, we can see that both the aggregated results and the visual presentation are plausible in terms of the expressed and expected features.

We will provide a statistical evaluation, showing how the temperature across our calculated fronts differs from randomly sampled temperature differences. We further present the mean inclination of each type of our calculated fronts. For the GPU version, we only calculated these statistics based on 1464 timestamps of the year 2016 evenly spaced at a 6 hour interval. Additionally, the GPU version is evaluated on baroclinity within September 2016 and we furthermore provide a case study based on our three-dimensional fronts in vicinity to a cyclone. Finally, we compare the runtime of both our algorithms and show results of a strong scalability test for up to 6 processes of the GPU version.

13.1.1 Setup

CPU version:
For all our tests of the CPU version, we first calculated surface front positions using the neural network from Part I of this thesis and saved them to disk. This way, the

rest of the pipeline could be executed without the need for a GPU. To create the results shown in this work, we run the method on a compute node with an Intel 2630v4 CPU with 10 cores and 128 GB RAM. For the runtime comparison against the GPU version, we used a dual 20-core Intel Xeon E5-2698 v4 at 2.2 GHz using Python 3.8.2 compiled with GCC 9.3.0 using a single process. Parameter were chosen as provided in the repository.

GPU version:
For the scalability evaluation of our algorithm, we used up to 6 nodes connected via Infiniband HDR. Each node is equipped with an AMD EPYC 7713 dual socket CPU with 64 cores per socket and an NVIDIA A40 GPU.

For the results presented in Section 13.6.3 we further used a single node equipped with an Intel Xeon Gold 6237 CPU and an NVIDIA GV100 GPU.

We used CUDA 12.3, GCC 12.3.0, and OpenMPI 4.0.1. For reading netCDF we use the current master branch of netcdf-cxx4, netcdf-c 4.8.1 and HDF5 1.10.7.

We used a single GPU and up to 4 CPU threads per process to evaluate the CUDA version, one thread for reading, processing and writing respectively as well as the main thread. As the used library for reading and writing netCDF files is not thread-safe, we serialized IO, which is why we omitted using multiple read or write threads. We set our parameter as $d = 20$ km (in accordance to [48]), $k = 8$ and $m = 5$ (see Section 11.5). $k = 8$ is a restriction chosen for computational purposes, as offsets larger than 8 are extremely unlikely to be the optimal choice. With the same argument, $k = 8$ does not really restrict the optimization algorithm. We run our algorithm for $m = 1, 2, 5, 10$, and evaluated the scoring function (eq. 11.1, chosen in accordance to [48]) with $m = 10$ for 720 samples from $09/2016$. We evaluate the results by the mean score obtained over all samples. While running the algorithm with $m = 10$ expectedly provides the best results, we found that $m = 5$ provides only a slightly worse score while needing less compute. All other evaluated values of m generally scored worse, and were not further considered. In the end, we chose to run our evaluations with $m = 5$, as a compromise between score and compute.

13.1.2 Evaluations

Removed samples in the CPU version:
In the evaluations of the CPU version, we removed results where the optimal separation point was found to be at the 0 location, within the reduced cross-section grid. In addition, all locations where no valid separation location was found were

also set to 0 before the filter was applied, removing those as well from the evaluation. This results in approximately 7% to 20% of cross-sections being filtered per level. Here, we can observe that the lower and upper levels contain more filtered samples compared to the levels around 850 hPa, which indicates that frontal characteristics are more pronounced at this level instead of high altitudes or near surface. Potential causes why no valid separation could be found include falsely identified surface fronts, weakly expressed frontal characteristics at the edges of a front, bad samples caused by the additional sampling applied in the CPU version or that the front is not present at this level at all.

In the GPU version, invalid fronts were marked during the algorithm. This can occur if a filter is added and not fulfilled or if parts of the input data yield invalid values (e.g. negative humidity values).

Evaluated types of fronts:
As described in the paper and this work, the CPU method was only designed for warm and cold fronts in the first place. We therefore only evaluate these types of fronts for this method. In Section 11.7 we mentioned, that we consider occluded fronts to consist of a warm front and a cold front overlapping. As a result, our GPU version obtains two results for the occluded fronts, which we evaluate separately. These are annotated as *occluded fronts* and *flipped occluded fronts*, respectively. As the orientation between different occlusions may be inconsistent, depending on the thermal gradient, we found that both, *occluded fronts* and *flipped occluded fronts*, contain a mixture of warm and cold parts of different occlusions. Nevertheless, we should still be able to see temperature gradients across the frontal surfaces.

For our GPU version, we therefore evaluate the following 5 types of fronts. *warm fronts*, *cold fronts*, *occluded fronts*, *flipped occluded fronts*, *(quasi-)stationary fronts*.

13.1.3 Sensitivity of the GPU implementation against input noise

We measured how random shifts of the provided 2D fronts affect our results, i.e., the placements of 3D fronts. We only investigate shifts of the complete front, preserving the frontal shape. Shifting every vertex individually, would introduce unphysical curvature and waves. Frontal shifts are normally distributed with $\mathcal{N}(0, \sigma)$, whereas $\sigma = 2, 4$. The shift is obtained in units of $0.25°$, which equals the pixel size of our atmospheric grid. We evaluate the quality of the results against the case without noise. We score each front individually, by evaluating the median of the minimal distance between any pixel of the “noisy" front input to the nearest pixel of the same

front obtained from the regular input. We refer to this measurement as distance D. As shifts preserve overall frontal structure, we should not encounter edge cases where the frontal points collapse to a single location. Short fronts consisting of 2 or fewer segments were filtered before this evaluation. One potential issue is wrong orientation of cross-sections, as the thermal gradient at the new position has flipped its sign. This can, e.g., occur near an occlusion with warm and cold front in close vicinity. This effect is also visible in our results, where, for $\sigma = 4$, we obtain an averaged distance value of $D = 0.61°$ at $1000\ \mathrm{hPa}$ for warm fronts. At $700\ \mathrm{hPa}$ this value increases to $D = 0.67°$. If we, however, enforce the orientation of the shifted front in accordance with the regular case, these values decrease to $0.30°$ and $0.12°$ respectively, showing that the algorithm actually decreases the distance with increasing altitude. For cold fronts, a similar observation can be made with $D = 0.40°$ $(0.25°)$ at $1000\ \mathrm{hPa}$ and $0.32°$ $(0.11°)$ at $700\ \mathrm{hPa}$, respectively. For $\sigma = 2$, the distance for warm fronts decreases to $0.33°$ $(0.16°)$ and $0.38°$ $(0.08°)$, respectively, which is already less than 2 (1) pixels. For cold fronts, all 4 values fall below $0.18°$. Occlusions behave very much like warm fronts, which enhances our point about flipped cross-sections in the vicinity of an occlusion. The flipped occlusions, however, perform worse for $\sigma = 4$ with distances of $0.71°$ $(0.55°)$ and $0.82°$ $(0.34°)$, respectively. At $\sigma = 2$ distances at high altitudes are once again similar to the warm fronts, although the near surface deviation is worse with $0.37°$ $(0.24°)$; this may be due to insufficiently gradient information within an occlusion. Overall, we find that while errors in the initial placement of a front affect the results, these effects are mostly restricted to a distance of a few pixels. Additionally, we can also see that these shifts can affect the orientation of the extracted cross-sections, which introduces larger errors, as different fronts are identified. Nevertheless, most frontal algorithms should be skillful enough to keep warm and cold fronts separated.

13.2 Evaluation of thermal gradient

In this section, we will provide information regarding the thermal difference across the fronts identified by our CPU version, based on which we define a randomly sampled baseline, which we then use to provide a statistical evaluation of the thermal difference across all evaluated pressure levels for both our methods.

13.2.1 Probability density distribution of temperature difference

A typical warm or cold front is classified by a strong temperature gradient between two air masses. During the passing of a warm front, the temperature rises, while for a cold front it falls. For our cross-sections, this means that a warm front should have a cold side to the right side of the center (i.e. the side the front has not yet passed) and a warm side at the left of the center (i.e. the side where the front already passed through). For a cold front, the location of the warm and cold side are swapped.

To highlight the effect of a passing front, we plotted the distribution of the difference of the mean temperature in a region $200\ \mathrm{km}$ behind the identified front and the mean temperature in the region $200\ \mathrm{km}$ ahead of the front at $500\ \mathrm{hPa}$, $850\ \mathrm{hPa}$ and $1000\ \mathrm{hPa}$ for the results of our algorithm. We further plot the results of a baseline at $850\ \mathrm{hPa}$, where the locations at which we evaluate the temperature difference within a cross-section were randomly sampled instead.

Algorithm results

Figure 13.1 shows the histogram of temperature differences for fronts identified by our algorithm. As expected, we can see for all presented levels that the temperature does indeed rise (fall) with the passing of warm (cold) fronts. We can further see that the strongest temperature difference can be found at $850\ \mathrm{hPa}$, a level commonly used for detecting fronts. Further, the plot shows that the temperature difference at $1000\ \mathrm{hPa}$ is higher than at $500\ \mathrm{hPa}$, indicating that the temperature gradient at higher levels is less expressed than at (near) surface level. For both cold and warm fronts, we can see that a small percentage of identified front locations express a wrong sign in the temperature difference, which is most likely caused by false positive frontal positions or wrongly oriented samples.

Randomly sampled baseline

As a comparison, we plotted the temperature differences at $850\ \mathrm{hPa}$ if the frontal separation was uniformly randomly sampled instead of using our algorithm in Figure 13.2. The data for the random samples consists of the same cross-sections used for the cold (warm) fronts. To create these plots, we aggregated the results every 168 hours (1 week) within 2016. In addition to the histogram, we also fitted both a normal and a Laplace distribution to the data. As can be seen, the Laplace distribution provides a good fit for the temperature difference of the random

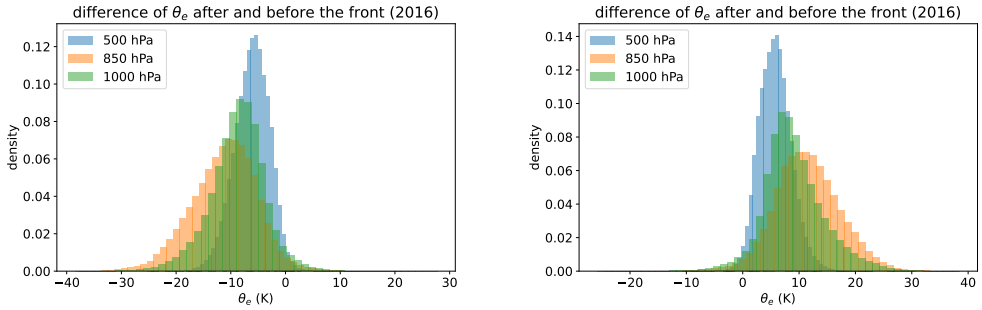


Fig. 13.1.: CPU version: Histograms showing the distribution of temperature differences across (a) cold and (b) warm fronts as calculated by our method at different height levels.

samples. Further, the choice of warm or cold front cross-sections barely changes the distribution of the random samples. In both cases, the mean μ and the peak are located near 0. We realize that in this case the position of the standard deviation σ of both distributions almost coincide, thus it makes sense to use σ as a measure for testing the quality of the derived probability densities.

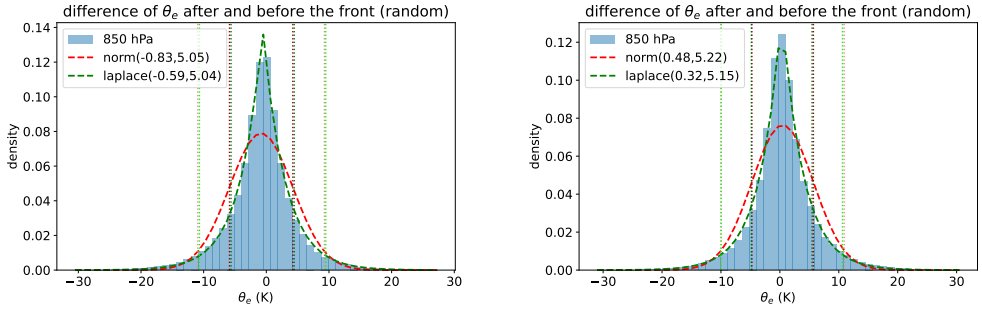


Fig. 13.2.: CPU version: Histograms of temperature differences for randomly separated cross-sections. Additionally, a fitted Laplace and normal probability density function, as well as mean ± 1 or 2 standard deviations are displayed.

13.2.2 Temperature difference at various levels

In this section, we present results for all height levels, showing how the algorithm fares against the randomly sampled case and showing that the algorithm does provide meaningful results. In Figure 13.3, we plotted the mean temperature difference of the random separation case, as well as $\mu \pm \sigma$ and $\mu \pm 2\sigma$ for each height level. Additionally, box plots of the results of our algorithm at each evaluated

pressure level were added to highlight the differences to the randomly sampled case.

As seen in the previous section, the most dominant temperature difference, even for the random case, lies around $850\ \mathrm{hPa}$. This is in agreement with various non-DL-based front detection algorithms, that rely on the thermal field at $850\ \mathrm{hPa}$ to determine the location of fronts. Our results confirm that this commonly used but empirically chosen pressure level is indeed a good choice.

In comparison to the random case, the plots show that for most cases our results are at least at a σ distance from μ, with the more extreme samples even exceeding the 2σ distance. Mean (green triangle) and median (orange line) also tend to be located closely to the 2σ line, sometimes even exceeding it.

As mentioned in Subsection 13.2.1 and Figure 13.2, we can see that the random fronts temperature difference distribution can be approximated using a Laplace distribution $L(\mu, b)$ with mean $\mu \in \mathbf{R}$ and scale $b \in \mathbf{R}$. We know that the standard deviation can be calculated as $\sigma = \sqrt{2}b$, and from this we can calculate that a point randomly sampled from a Laplace distribution has a probability of approx. 75% (resp. 94%) to lie within the interval $[\mu - \sigma, \mu + \sigma]$ (resp. $[\mu - 2\sigma, \mu + 2\sigma]$). This further enhances our point, that the extracted 3D fronts are meaningful.

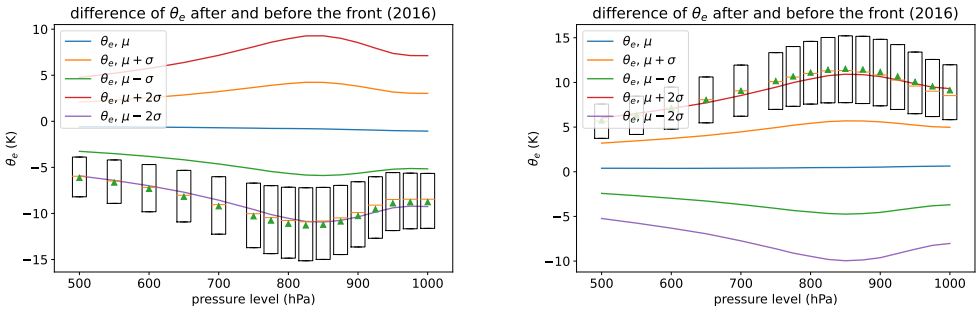


Fig. 13.3.: CPU version: Temperature difference across the detected fronts for different pressure levels for (a) cold and (b) warm fronts as box plots including their mean (green triangle) and median (orange line). Additionally, the mean (μ) as well as the one and two σ intervals for the randomly separated fronts are inserted as orientation.

We plot the same distribution for our GPU version against another randomly sampled baseline created with the GPU method. We again calculate mean (μ) and standard deviation (σ) from these randomly generated samples. We then plot the distribution of temperature differences calculated by our algorithm as box plots and compare these results against the mean and σ or 2σ difference of the random samples. The

mean and median of the algorithm results are depicted as a dashed or full line in the box plots, respectively. The temperature differences $\Delta\theta_e$ are calculated according to Equation 11.1 with m set to 10. With a given cross-section step width of $20\ \mathrm{km}$, this evaluates up to $200\ \mathrm{km}$ in front as well as behind the calculated frontal surface. We chose this distance for evaluation in accordance to the previous implementation, to facilitate a qualitative comparison between the two methods, if desired.

As can be seen in Figure 13.4 (first two rows) we obtain qualitatively similar results for the GPU implementation for both cold and warm fronts, where the median of our distribution is approximately at the 2σ distance of the random distribution, sometimes even exceeding it. We can also observe that the most dominant temperature differences are still located around the $850\ \mathrm{hPa}$ pressure level. As we updated some core components of the pipeline, such as the cross-section generation, we cannot expect to obtain the same results, which renders a direct quantitative comparison pointless. Nevertheless, seeing as the relative distribution compared to the random cases is similarly expressed as in the previous implementation, we can be confident that the results are of a similar quality. More importantly, the GPU method allows us to evaluate the same score for occluded fronts as well as stationary fronts, also displayed in Figure 13.4. For the occluded fronts, the calculated distribution is less distant from the mean temperature of the random samples. There are several explanations for this. First, the temperature difference at the lower levels is expected to be smaller, as the overlap of the cold and warm front part leads to a temperature gradient between a cold and a cool air mass rather than a cold and a warm air mass as it would be the case for warm or cold fronts. Second, even past the occlusion point, where warm and cold front intersect, the mean temperature difference is still lowered, if the enclosed warm sector is not wide enough. Further, several occlusions tend to coincide with cyclones or other strongly curved phenomena. As a result, we do not obtain such clear temperature gradients, especially not, when comparing over larger areas. The curved structure may mix several cold and warm sectors, thus smoothing the mean temperature at either side of the front; this leads again to a reduced temperature difference. For stationary fronts, the results are also weaker than for the warm and cold fronts, however, we can still see the same tendencies. Also, the median still exceeds the σ distance for almost all pressure levels, which still shows a relatively strong signal.

While the statistical evaluation shows that our presented method performs well compared to a completely random method, it is questionable, how much the random sampled baseline is a meaningful comparison. A better approach, might be to provide a naive implementation for the creation of three-dimensional fronts as a prior. The most simple approach would be to extend the $2D$ front to all height levels,

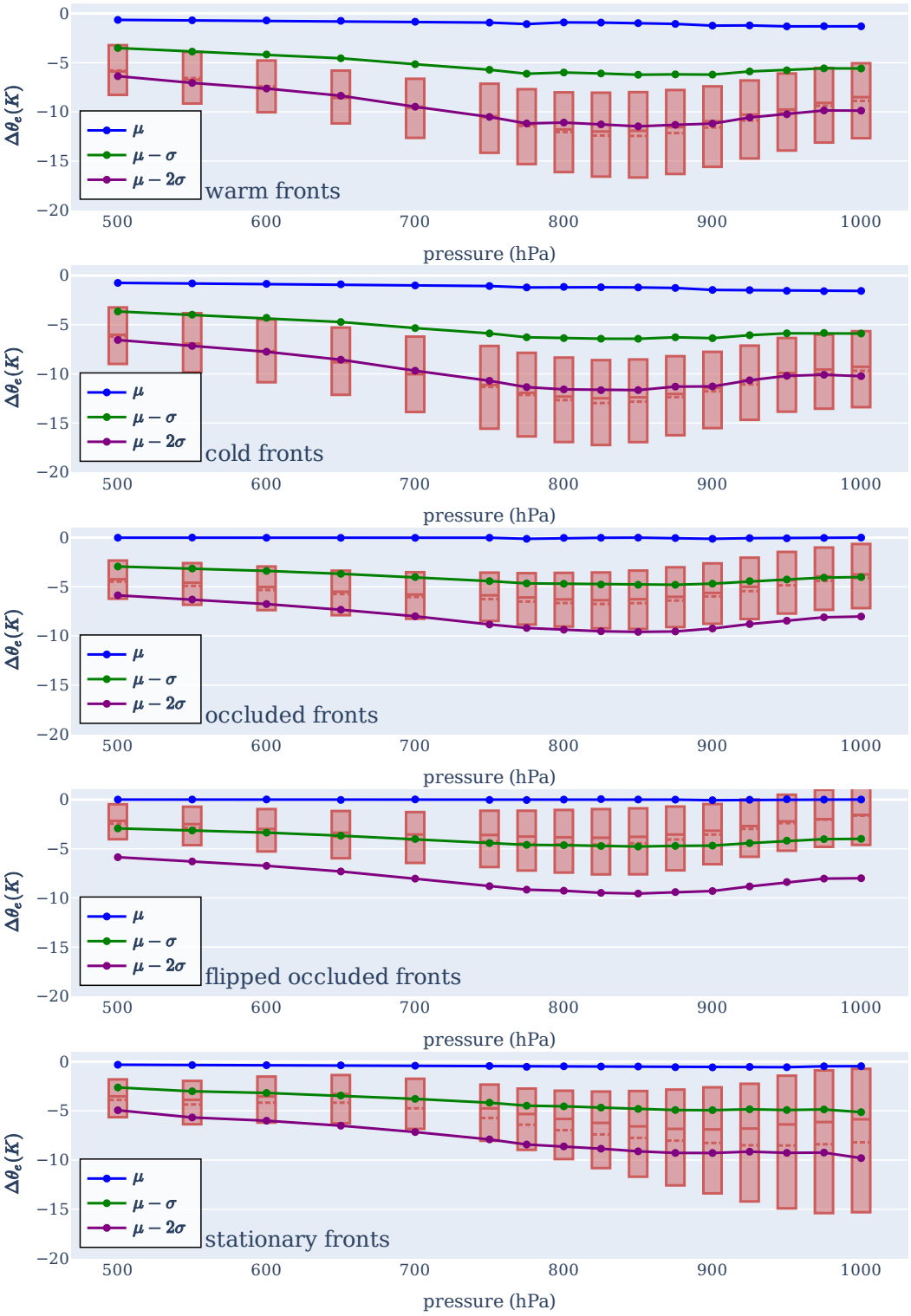


Fig. 13.4.: GPU version: Box plots of the distribution of temperature differences across the calculated fronts for 16 pressure levels. In addition, the mean (μ) as well as the 1 and 2 standard deviations σ distance for the randomly sampled points are shown.

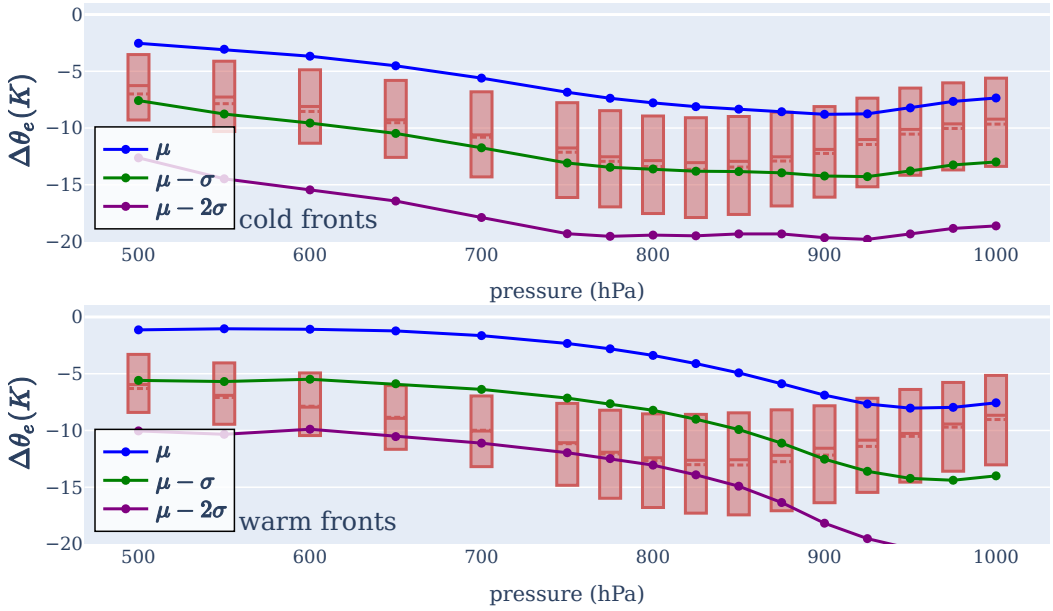


Fig. 13.5.: GPU version: Comparison of algorithm results against a static baseline

ignoring any possible inclinations. At least for cold fronts, such an approach could already provide a good approximation of the front, as these fronts are very steeply inclined. As such, the deviation from the optimal front point would most likely be very low. Therefore, the difference in the score should be low in a statistical evaluation as in Section 13.2.2. Warm fronts, on the other hand, have a flatter inclination, exhibiting a higher disparity between the surface front location and the higher level front locations. Results of such an evaluation for September 2016 are depicted in Figure 13.5. We can see the expected behavior. For the baseline, we can see that the mean temperature difference increases towards 0 at higher altitude. This effect is more pronounced for the warm fronts. This behavior is in line with the expectation that the warm front quickly deviates from its surface position as the pressure level decreases. Thus, the static baseline quickly deviates from the frontal surface, becoming more similar to the randomly sampled case. Albeit, we can still see that the static case scores slightly better than the random case, due to the still present correlation to the warm front. For the cold fronts, this is less dominant. As these fronts have a steep inclination at surface level, the deviation from the frontal surface at the lower levels is small. Thus, we can observe, that near the surface level, both the algorithm and the static evaluation score similar. With increasing height, however, the algorithm exhibits stronger differences in temperature, approaching the σ or 2σ distance for the cold or warm fronts, respectively, at around 800 hPa. These findings are in line with our expectation, that the algorithm is capable of following the frontal surface as the altitude increases.

13.3 Inclination

In a second evaluation, we determine the inclination of weather fronts and provide a histogram of how the inclination is distributed across fronts of each type. For the CPU version, we determine the inclination in degrees using the following formula:

$$\text{inclination} = \arctan\left(\frac{|\text{pressure\_difference}\ (\mathrm{hPa})|}{\text{horizontal\_distance}\ (\mathrm{km})}\right) \tag{13.1}$$

While the formula uses different units, it has the advantage of better showing the inclination, as opposed to describing the vertical difference in terms of km, since the atmosphere's length scales are very different and thus the aspect ratio (vertical vs. horizontal scale) is very small. For instance, the horizontal extent of a pixel is $20\ \mathrm{km}$ while the vertical distance from $1000\ \mathrm{hPa}$ to $500\ \mathrm{hPa}$ is only approximately $5\ \mathrm{km}$. Determining the inclination on this scale would make it hard to see differences in the angles, and results in mostly near zero inclinations. In Figure 13.6 we show the mean cold (a) and warm (b) front as well as the average inclination of the frontal objects from $1000\ \mathrm{hPa}$ to $600\ \mathrm{hPa}$ (left) and from $1000\ \mathrm{hPa}$ to $850\ \mathrm{hPa}$ (right). To calculate the inclination of a frontal object, we first determine the median position of the frontal separation points for each pressure level. The inclination between two pressure levels is then described by the line connecting the resulting positions for the corresponding pressure levels (e.g. $1000\ \mathrm{hPa}$ and $850\ \mathrm{hPa}$ or $600\ \mathrm{hPa}$). The median position is chosen as it is more robust towards outliers, where the algorithm may have failed to determine one of the separation points correctly. The mean inclination of frontal objects as depicted in Figure 13.6 is then obtained by connecting the median separation positions averaged over all fronts. In case of the warm fronts (Fig. 13.6b) we can clearly see the expected flat inclination, while for the cold fronts (Fig. 13.6a) we can see a rather steep inclination of near $90°$ and a slight backward tilt for the inclination. Both of these observations are in accordance with the typical theory surrounding the shape and inclination of warm and cold fronts.

For the GPU version, we conducted a similar evaluation (see Figure 13.7). Differently to the previous inclination plots, we plot the calculated front position on the corresponding cross-section for each pressure level relative to the frontal position at the lowest level (i.e. the surface position is always located at 0). Calculating the average plot over all sample points, we then obtain an image showing the average cross-section of a frontal surface. As all of our cross-sections were extracted such that the thermal gradient is negative, all samples should have the cold section of the front at the right of the frontal surface (more positive values on the x-axis) while the warm sector should be on the left (more negative values on the x-axis).

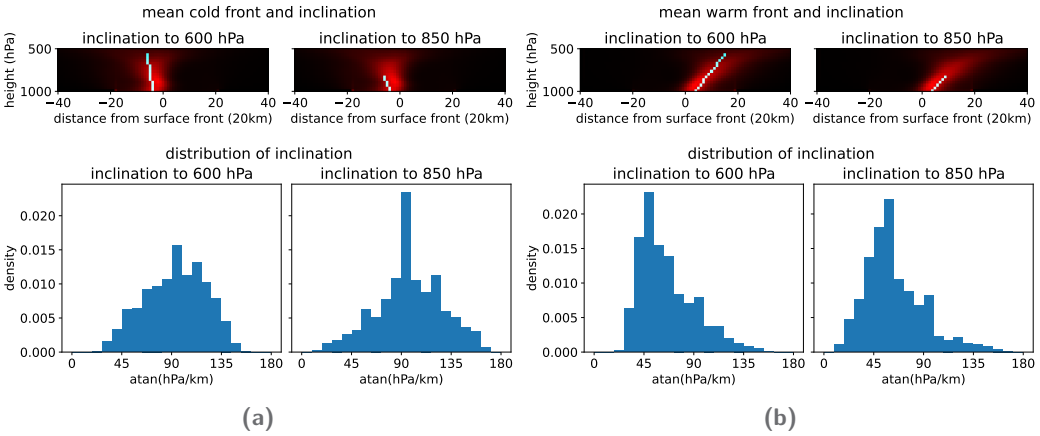


Fig. 13.6.: CPU version: (a): Average frontal separation for cold fronts in red. The light blue line indicates the average inclination from surface (1000 hPa) to 600 hPa (left) or 850 hPa (right). Histograms show the distribution of inclinations accordingly. (b): as (a), but for warm fronts.

As in the previous implementation, we can see that the calculated warm fronts concur very well with the theory, which expects a flat inclination over a long distance. For the cold front the results are a bit more mixed, where the fronts tend to be rather steep, with a near vertical inclination. As the front goes up, we can see inclination in both direction, however mainly staying near the surface front's position. Using data at levels higher than 500 hPa might show more curvature, however, it is to expect that the quality of the detected fronts decreases as well.

For the GPU version, we can also present average inclination information for both, occluded and (quasi-)stationary fronts. For the latter, we can see that inclination is mainly vertical, with deviations in both directions existing. This result seems to be plausible, as stationary fronts are characterized by their very low travel speed. The effects which lead to the inclination of warm or cold fronts are therefore less dominant, thus we might expect these fronts to show almost no inclination. Finally, for the occluded fronts, we see two dominant signals. As we would expect, one signal tilts over the colder zone, similar to how a warm or cold front would be tilted. This is expected, as an occluded front arises from the overlap of a warm and cold front. Especially above the occlusion point, where both the warm front part and the cold front part might still be separated, such a signal should be observable. On the other hand, we can see a similarly strong signal, which is inclined in the opposite direction. Differently to the cold front, that signal is already present at lower levels, which indicates, that an occlusion may differ from the expected behavior of a warm or cold front as seen before.

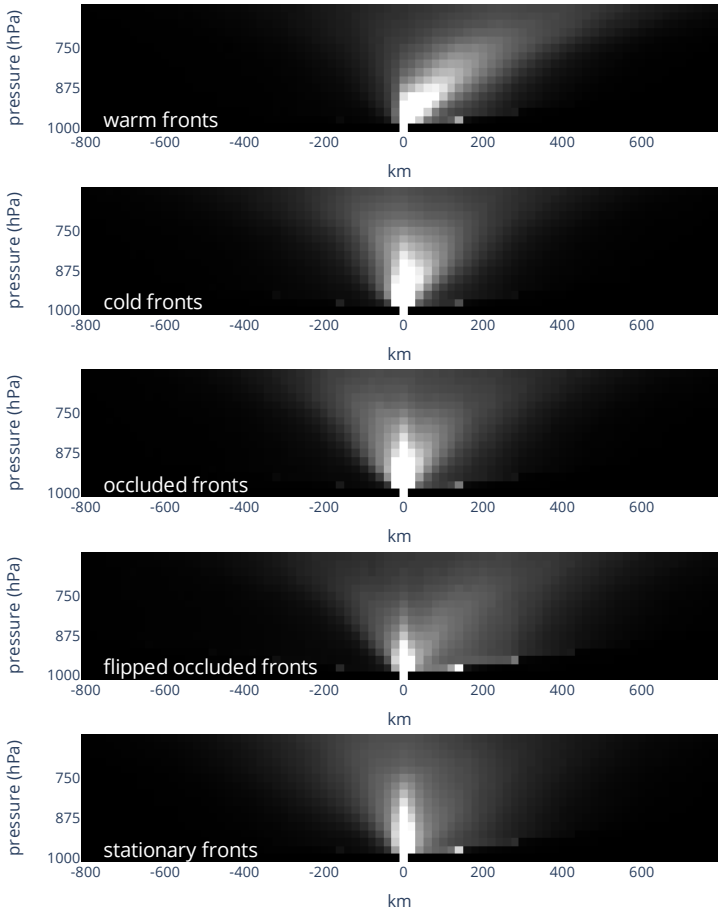


Fig. 13.7.: GPU version: Averaged position of the optimal frontal point, relative to the surface front.

13.4 Baroclinity

As temperature across the frontal surface, coincides with our optimization criterion, which might introduce a bias in the evaluation, we conducted a second evaluation, which uses the baroclinity in vicinity to our fronts, instead. This evaluation was only conducted for the GPU implementation, however we expect similar results on the CPU based method, albeit inconsistencies in the orientation may impact the quality. Typically, a front is drawn onto the ridge of the isobars, following the steepest loss in pressure. At the same time, the thermal gradient should be oriented across the front, effectively (near-)perpendicular to the pressure gradient. As a result, it is to be expected that the baroclinity $|\nabla p \times \nabla T|$ reaches a maximum at the front, as these two vectors should be near orthogonal as well as have a high absolute value. As our fronts are detected on pressure levels, we cannot use p. Instead, we substitute the formula to $|\nabla \phi \times \nabla \theta_e|$, where we use the geopotential (ϕ) instead of the pressure (p). Further, we replace the temperature (T) with the equivalent potential temperature (θ_e), which is the thermal quantity our algorithm uses. As we are only interested in the position of the maximum of this cross product, we can ignore the change in overall scale. An interesting property of this feature is, that it should mainly be present around fronts and not other points, which exhibit a strong thermal gradient without being a front. It further serves as a good indicator of our algorithm's skill, as we do not directly optimize against this feature.

We calculate the baroclinity on the atmospheric grid, and sample the values the same way as we created cross-sections (see Section 11.5). We then extract the baroclinity $200\ \mathrm{km}$ ahead and past the front by sampling 21 points from the cross-sections centered around the identified position of the front. We averaged the results over all samples of September 2016. The resulting values are then shifted and scaled to $[0, 1]$ for each type of front individually for better visualization. In Figure 13.8 these results are shown. We further provided a second evaluation, where we normalized each sample before calculating the average (see Figure 13.9). With this, we can reduce the bias induced by extreme samples, where the baroclinity may be much larger than usual. A possible case for such an extreme value may be due to errors in the calculation of the θ_e caused by faulty input values present in the ERA5 datasets (e.g. humidity below 0). While these cases are rare, their high values can have a notable impact on the score. Interestingly, we find that the detected fronts are very much located at the maximum baroclinity. For warm fronts we can even see a slight displacement, which however is very sensible, as the typical placement of a front in weather charts coincides with the leading edge of the warm air mass which we would expect to be slightly offset from the largest thermal gradient, which we detect.

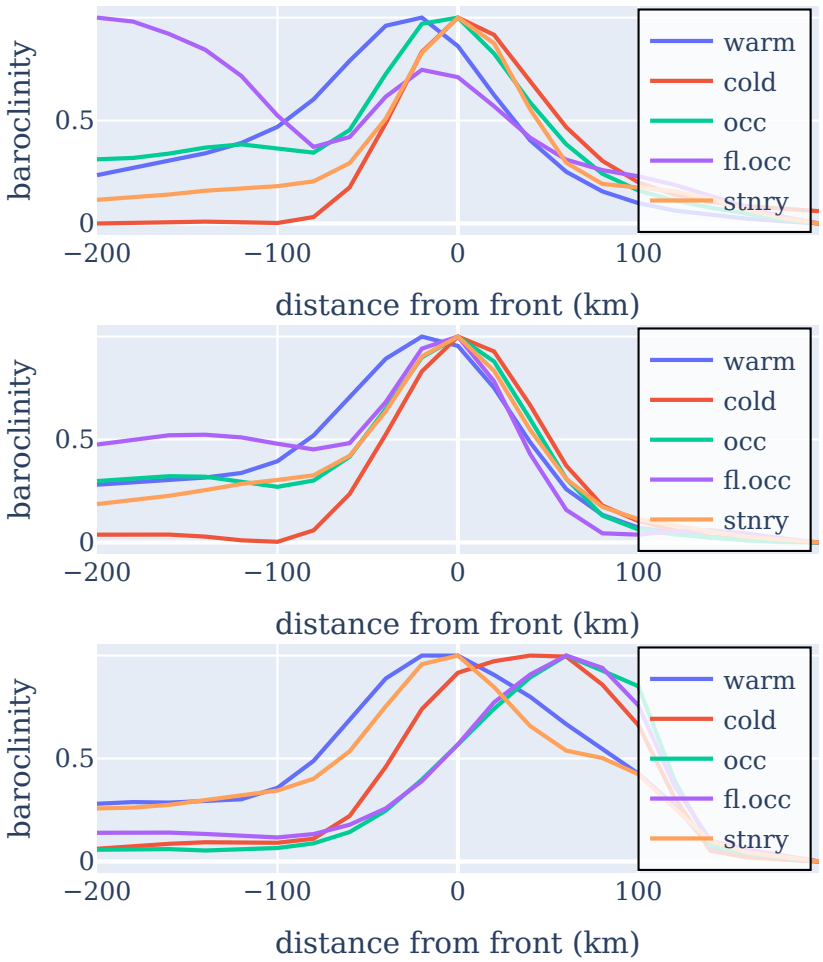


Fig. 13.8.: Normalized baroclinity for 1000 hPa (top), 850 hPa and 700 hPa (bottom) across the identified frontal surfaces for each type.

Overall, we can see that the results of our algorithm are sensible with respect to the expected behavior across fronts. At the highest level of 700 hPa, we can observe a shift in the position of maximum baroclinity for the occlusions. This may be due to the presence of clouds near the front, which exhibit a stronger gradient in the used θ_e field than the front itself does. Although, in Figure 13.9 this effect vanishes, which would fit to our assumption that the offsets are caused by the influence of larger values in the mean calculation.

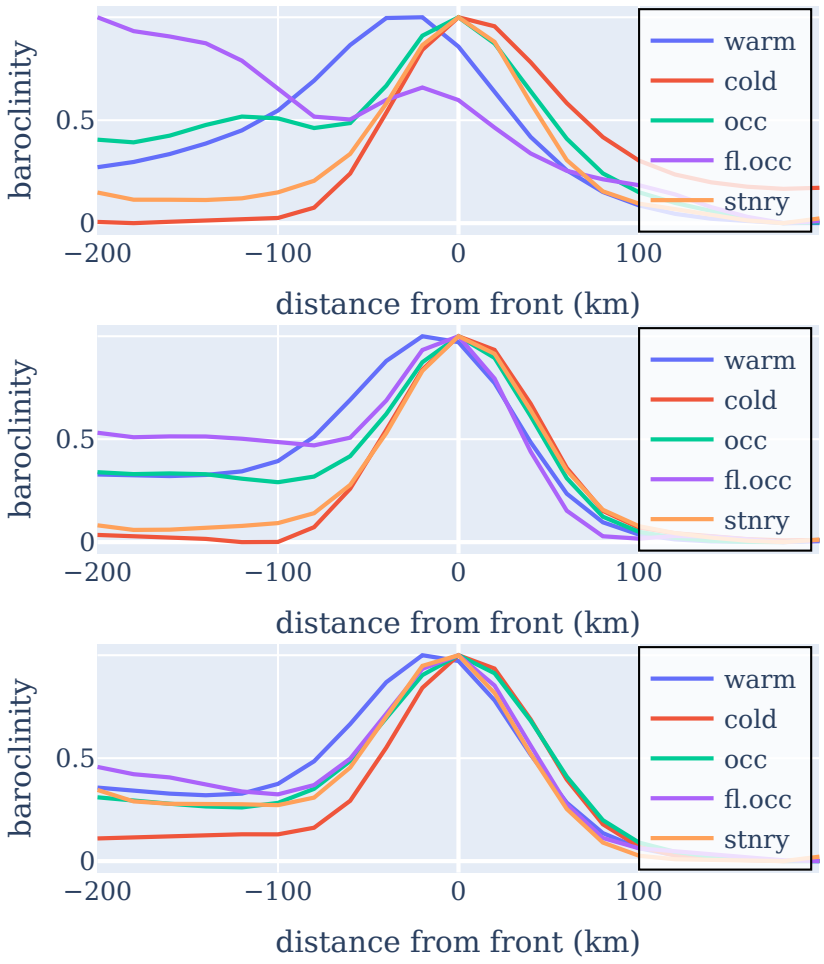


Fig. 13.9.: Same as Figure 13.8, however, each sample has been normalized before aggregation.

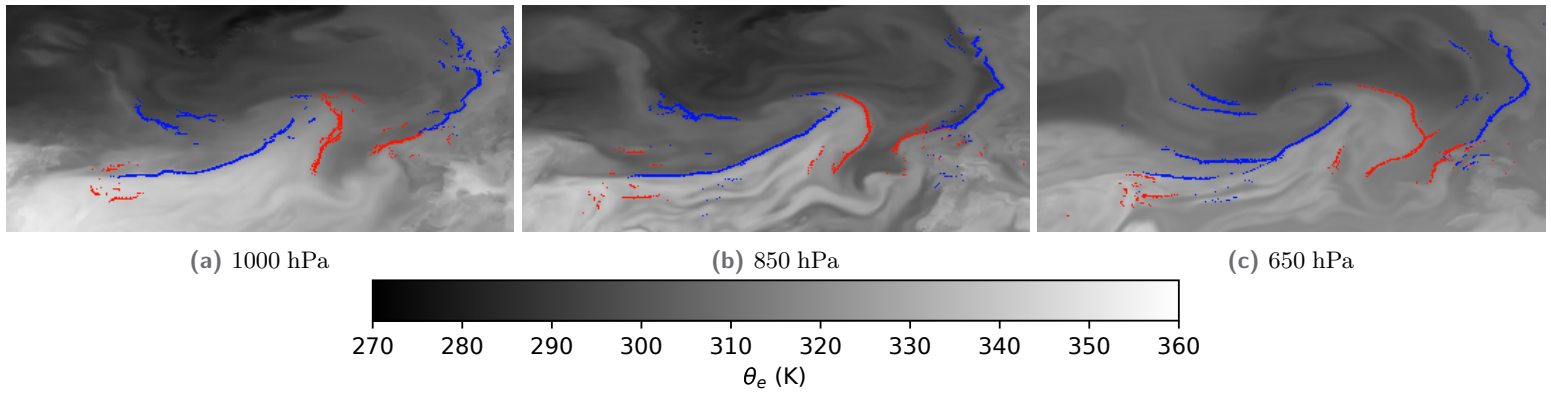
13.5 Case study

In this section, we will compare the three-dimensional fronts as provided by both presented implementations, based on an exemplary case. We will then provide more details of the case study, based on the results of the GPU implementation.

13.5.1 Comparison of three-dimensional fronts at 2016-09-23::00 UTC

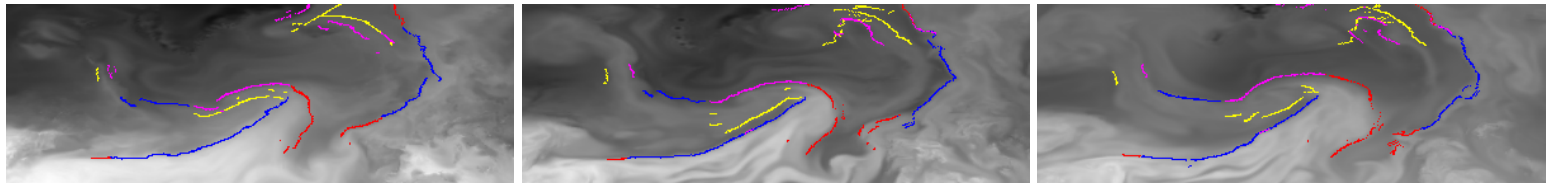
Figure 13.10 shows warm and cold fronts as identified by our proposed method at $1000\ \mathrm{hPa}$, $850\ \mathrm{hPa}$ and $650\ \mathrm{hPa}$ for 2016-09-23 00:00 UTC over a background of θ_e (interval: $[270, 360]\ \mathrm{K}$) created with the CPU based Python implementation. We can clearly see an inclination of the central meridional warm front, as the frontal line "shifts" eastward with increasing altitude. The central cold front on the other hand stays roughly at the same location, indicating a much steeper inclination instead. Note that occluded and stationary fronts are not shown.

For the GPU version, we can see similar features in Figure 13.11. The meridional warm front at the center expands similar to the CPU version. At the higher altitude, we can however see that the sparser sampling creates a less dense representation of the frontal line. The cold fronts behave similar in both versions, although the GPU version is less likely to create doubled frontal features, as could be seen on the south-western cold front. Further, we can see that the occlusions at the center of the image are connected well onto the nearest warm front, continuing the circular or spiral movement of the cyclone. On the other hand, we can see that the inner spiral of the occlusion at the cyclone is not hit very well, but rather the southern part of the outer spiral is detected instead. This can be explained by the fact that in the lower levels the cold-front part of the occlusion is not well expressed. As such, during optimization, the point, where the cold side of the southern cold fronts meets warm air coming in from the west is detected instead. For the upper levels, this still exhibits a good gradient, which is why the inner spiral is omitted. This is further enforced by using the optimal path approximation, where the inner side of the cyclone scores comparatively bad, due to only being well expressed in the higher levels.



(a) 1000 hPa (b) 850 hPa (c) 650 hPa

Fig. 13.10.: CPU version: Locations of cold (blue) and warm (red) fronts at $1000\ \mathrm{hPa}$ (a), $850\ \mathrm{hPa}$ (b) and $650\ \mathrm{hPa}$ (c) for 2016-09-23 00:00 UTC. Shown snippet coordinates: $[70, 30]^\circ$N, $[-80, 10]^\circ$E. Background: θ_e (interval: $[270, 360]$ K)



(a) 1000 hPa (b) 850 hPa (c) 700 hPa

Fig. 13.11.: GPU version: Fronts at 1000, 850 and $700\ \mathrm{hPa}$ for 2016-09-23 00:00 UTC. Background is θ_e ranging from 270 K to 350 K. Fronts are color coded as warm (red), cold (blue), occlusion (pink and yellow.)

13.5.2 Case study: Cyclone Vladiana

In Figure 13.11, we present a representation of the three-dimensional fronts as three horizontal slices at $1000\ \mathrm{hPa}$, $850\ \mathrm{hPa}$ and $700\ \mathrm{hPa}$ respectively, representing the 3D structure of the scenario. The samples show the cyclone Vladiana, which formed on September 22^{nd} 00:00 UTC near Newfoundland and moved across the North Atlantic Ocean toward Iceland. This cyclone is a good example of a Shapiro-Keyser cyclone [71]. We can use this case to identify how well cold and warm fronts merge into the occlusion accompanying the cyclone. For further details of this case study, we refer to previous studies [3, 49]. The cyclone located at the center is visible across all pressure levels. We can clearly see the identified leading warm front at the eastern side of the cyclone as well as the cold front located at the western or southern side, both rotating anti-clockwise as expected for a cyclone. For a Shapiro-Keyser cyclone, we would not expect an occlusion, but rather a cold front fragmentation. Nonetheless, our algorithm only expands the labeling of the input data, thus the northern part is labeled as an occlusion rather than a warm front. Within the sample we can clearly see the warm seclusion located just north of the dominant cold front. Our algorithm is able to identify the continuation of the warm front within the labeled occluded front without fragmentation. On the other hand, we can see that the algorithm has problems to locate the flipped occluded front and places it at the lower end of the warm seclusion; this is not optimal, although sensible given the optimization paradigm. Moving on from the occlusion, the leading warm front expands rather rapidly with increasing altitude, which clearly shows the flat inclination, as commonly attributed to warm fronts. We can also see how the algorithm follows this inclination, although due to the radial expansion, the sampling becomes relatively sparse. The cold front on the other hand barely changes its position with altitude, which also corresponds to common theoretical expectations. This is also captured well by our algorithm. At the $700\ \mathrm{hPa}$ level, we see a fragmentation of the cold front due to warm air moving in from the west (see also Fig. 13.10(c)). Parts of the cold front remain at the southern position, where still a gradient exists, while the cold front part closer to the cyclone center tends to move more northwards, aligning with the northern edge of the intruding warm air. In this case, the algorithm is unable to detect the front north of the warm air mass, as only a single front per cross-section is identified. At the eastern side of the case study, we can see similar results for the meridional warm and cold fronts. The result adapts well to the small changes in structure and shape of this cold front.

In the Section 13.3 we showed results indicating that the inclination of occluded fronts behaves counter-intuitive, expressing two dominant modes of inclination.

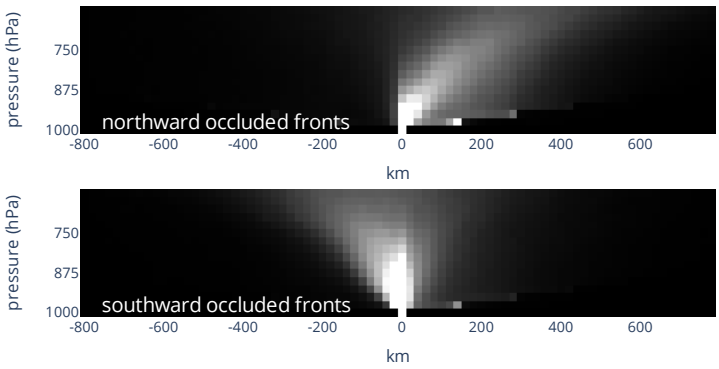


Fig. 13.12.: GPU version: As Figure 13.7 but only occlusions with northward (top) or southward (bottom) oriented normal

As in our case study, there is a high correlation between extra-tropical cyclones in the storm track region and occlusions [65]. This connection is also reflected in common cyclone models such as the Norwegian [66] or Shapiro-Keyser model [71]; in the latter case the occlusion is not central in the cyclone development, but may still occur in the cyclone dissipating stage. Furthermore, we have to assume that the underlying front detection method has labeled parts of the leading warm front as an occlusion. In both of these cyclone models, extratropical cyclones over the North Atlantic consist of a leading warm front (or the corresponding part of an occlusion) rotating anticlockwise, while the trailing cold front rotates in the same direction; however it may rotate at a smaller radius. From this motion, we can deduce that the part of an occlusion, with a normal direction pointing northwards is very likely corresponding to the warm front part, while a southward normal direction corresponds to the cold front part instead. For eastward and westward oriented normals, this distinction is not as clear, due to the strong curvature of occlusions associated with a cyclone. If we apply such a filter (northwards vs. southwards) during our statistical evaluation, this allows us to separate both modes of inclination into two parts, which in turn legitimates the initial assumption (see Figure 13.12, top row). Furthermore, this filter also moves the maximum baroclinity of the occlusions closer toward the identified frontal point (Figure 13.13), indicating that the warm front part within cyclones is generally well identified. A reason for this may be the fact that fronts, falsely labeled as an occlusion in the vicinity of a Shapiro-Keyser cyclone, do not exhibit a cold front part as we would expect, which in return leads to falsely identified fronts.

If we apply the opposite filter, we should gain the cold front parts of cyclonic occlusions. As showing in Figure 13.12 (bottom row), we can see that this filter also provides a clear separation, leaving us only with the "wrongly" oriented inclination mode of our occlusions. Looking at sample cases, we can see that this inclination

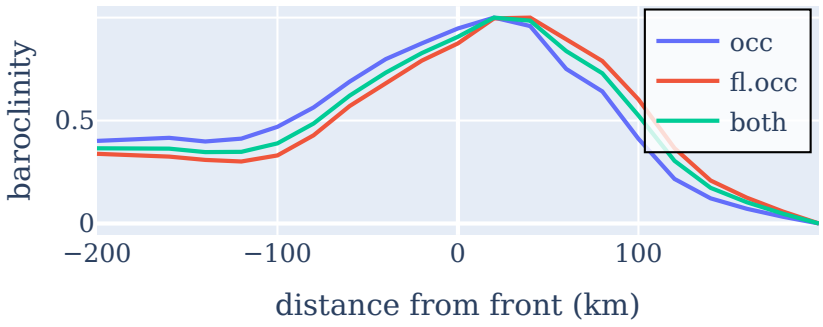


Fig. 13.13.: As Figure 13.8 (bottom) but only for occlusions with a northward oriented normal, corresponding to Figure 13.12, top row

behavior is correct with regard to the scoring function (Equation 11.1). However, this contradicts the expected shape of a regular cold front. To determine the cause of this, further investigation may be necessary.

13.6 Runtime evaluation

We will now compare the runtime of both implementations. Afterwards, we provide scaling information of our GPU version and give more details about the runtime of the individual stages of this method.

13.6.1 Runtime comparison

When comparing the runtime of the CUDA and CPU-based method, we need to take into account that the Python implementation directly works on frontal data as provided by the network, extracting the front and its orientation from the $2D$ grid. In comparison, the CUDA version uses an input data format, where the fronts are already extracted as textual polylines. The CUDA version on the other hand provides output for 4 types of fronts, with two different outputs for occluded fronts, while the Python implementation only provides cold and warm fronts.

For all our tests, we found that copying data from the distributed file system to the local file system and vice versa dominates the time necessary for the actual computation. Depending on the hardware setup, this may not be necessary if the compute machine has direct fast access to the file system, which is why we did not include these transfer times in our evaluation. Further, the transfer may influence

IO due to caching effects. However, as this would be the case for both measured methods, we decided to ignore this during evaluation.

For the CPU implementation, we measure the runtime for the conversion of 720 files, which corresponds to the 30 days of September 2016. This conversion took the algorithm 263 minutes and 26 seconds. Converting the same timestamps using the CUDA implementation, this conversion only took 40 seconds, which corresponds to a speedup of more than 350. This shows the potential of the GPU implementation, allowing for faster experimentation with various datasets and experimental setups, while providing additional information by showing positions of occluded and stationary fronts as well as providing information about the frontal orientation. Although the speedup is high, a comparison against a multi-threaded C++ implementation would be more adapt, especially as the CPU implementation was not tuned for performance. Therefore, the CUDA version has been additionally evaluated in terms of scalability, and we provide a breakdown of the individual stages, showcasing that the method is currently limited by file IO.

13.6.2 Scalability of GPU version

To evaluate the algorithm's scalability, we processed 8784 timestamps, one for each hour in 2016 using $1, 2, 4$ or 6 MPI processes, evenly splitting the workload among the processes. Each process has its own disk and GPU. The runtime was measured, including only the netCDF output. We present the absolute runtime, as well as the speedup and efficiency relative to the single process case in Table 13.1. The presented runtimes are obtained as the average runtime from three runs each. From the results we can see that the algorithm exhibits linear scaling with an efficiency of 95% up to 6 processes.

Tab. 13.1.: Strong scalability analysis for 1 to 6 processes, calculating the three-dimensional fronts for 8784 timestamps from 2016. Only the netCDF data was written to disk. Results are averaged over 3 runs.

#Processes	runtime (s)	speedup	efficiency
1	498	1	1
2	241	2.07	1.03
4	117	4.25	1.06
6	88	5.71	0.95

13.6.3 Breakdown of runtime on GPUs

For the GPU version, we additionally evaluated averaged runtimes per file for each of the three stages measured for 720 files of $09/2016$. Due to the slow runtime, we did not perform such an evaluation for the Python version. In Table 13.2 we show how much time each stage spends at certain tasks, when executed on the GV100. When providing netCDF output, the processing thread spends roughly 29% of its runtime processing the data, while more than 68% is spent waiting for any of the IO threads. As the netCDF reading and writing is not thread-safe, it has to be performed serially. IO of netCDF accounts for 97.8% of the time spent on reading from disk and 89.2% of writing to disk. Adding these values, our algorithm spends 95% of the overall time per file on netCDF IO operations, showing that our pipeline is able to almost completely hide any computational time. Further, we can see that reading data takes almost twice as long as overall processing of data. Running the code without netCDF output, we see that overall time per file reduces from $59.1\,ms$ to $20.4\,ms$. Surprisingly, other seemingly unrelated tasks execute faster, such as reading netCDF files or overall processing of kernels. Without the netCDF output, we can identify inefficiencies where the reading thread has to wait for the processing thread to process input data on the host before freeing the read buffer, which could eventually be solved by using multiple buffers. However, we did not further optimize IO in this work. The calculation process itself, can be split into the four stages as mentioned in Section 12. We further evaluated the relative runtimes of these stages within the processing steps, ignoring any wait times. The largest part is spent on cross-section creation (36%), scoring (27.8%) and optimization (19.3%). Base point creation only takes less than 1%. The remaining time is spent on processing the input data on the host (9%), conversion to θ_e (3.8%) and memory operations such as resizing or allocation (3%). There is potential for a better overlap of memory transfers, although this was currently not a main focus.

As we can see, the runtimes for the A40 GPU (see Table 13.3) differ from those observed on the GV100. While on the one hand the underlying setup appears to be better equipped to perform the IO tasks, we can see that calculation time grew, while memory transfers were faster. Inspecting the individual kernel, we see that the memory heavy kernels, such as cross-section creation and conversion to θ_e took almost $3ms$ longer than for the GV100. Shifting the relative runtimes more towards these kernels, with the cross-section creation now taking almost half of the overall calculation time. The processing kernel on the other hand became faster, which counteracted the mentioned slowdown. Some of this could be explained by the fact that the GV100's memory bandwidth ($870GB/s$) is larger than the A40's ($676GB/s$).

The higher processing power of the A40 does not seem to be able to fully compensate this difference, which leads to slightly slower overall calculation times. Nonetheless, the main difference comes from the disk IO. Both these observations further enhance our assumption that this algorithm is very much restricted by memory accesses, both from disk and on the GPU.

Tab. 13.2.: Relative runtimes per file (ms) using a GV100 GPU

stage	wait	calc/copy	IO	other	total
read	30.3	-	28.0	0.6	58.8
process	40.1	12.9/4.1	-	1.5	58.6
write	6.0	19.7	32.9	0.4	59.0
read (no netCDF)	4.4	-	15.8	0.1	20.2
process (no netCDF)	4.1	11.2/3.9	-	0.9	20.1
write (no netCDF)	13.6	-	5.2	1.4	20.2

Tab. 13.3.: Relative runtimes per file (ms) using a A40 GPU with 10GB RAM

stage	wait	calc/copy	IO	other	total
read	31.7	-	15.2	4.0	50.9
process	29.3	16.4/2.1	-	4.0	51.8
write	2.0	32.0	17.7	0.3	52.0
read (no netCDF)	1.8	-	18.6	0.8	21.3
process (no netCDF)	2.4	16.1/1.9	-	0.8	21.2
write (no netCDF)	16.9	-	4.2	1.0	21.3

14 Conclusion and future work

14.1 Conclusion

We presented a novel pipeline for finding three-dimensional structures of frontal lines based on a surface front detection neural network. Our algorithm uses the prior knowledge of frontal characteristics and surface front positions to expand frontal lines from a two-dimensional grid to three-dimensional structures. This model is the first that enables statistical analysis of three-dimensional fronts. Therefore, we can provide a statistical evaluation of characteristics of automatically detected three-dimensional fronts, which previous methods were not suited to do. Our method may thus provide important insights, deepening the understanding of fronts and their three-dimensional structure and how these are connected to other weather phenomena.

We further presented an extension of the method, which utilizes GPU hardware. This implementation vastly outspeeds the previous implementation by more than two orders of magnitude, which in turn allows for faster creation of three-dimensional fronts, speeding up possible applications of the algorithm. At the same time, the presented method provides linear scaling, which allows even faster computation using multiple GPUs. We updated several stages of the pipeline to create more robust calculation and a scalable parallelization of the algorithms. Furthermore, the GPU version additionally allows us to create three-dimensional fronts for stationary and occluded fronts. Thus, we are able to expand all types of fronts, commonly depicted by weather services or our network.

We were able to show that our identified $3D$ fronts exhibit a steep temperature gradient across their surface, which is an integral frontal feature. Furthermore, we highlighted differences in the inclination of the different types of fronts. Especially for warm fronts, we could reproduce the commonly expected flat inclination as prevalent in theoretical descriptions of fronts. This effect could also be seen when comparing the thermal gradients across the frontal surface against a static baseline, where we showed that our warm fronts further deviate from the static baseline compared to cold fronts. Additionally, for the CUDA implementation, we investigate the robustness with respect to noise in the 2D surface front data. Further, we

compare the results with the baroclinity as an independent measure for fronts and found good agreement. In a short case study of an extra-tropical cyclone, we showed the ability of our method to determine the 3D structure of fronts; Driven by the case study, we could also discriminate different types of occluded fronts statistically. Finally, the runtime and scalability of the new method was evaluated.

14.2 Future work

While the application region of the algorithm was currently restricted to the North Atlantic, the high processing speed makes global application feasible, as long as appropriate surface frontal data is available, such as the results generated by the method of Niebler et al. [44]. This would enable further investigation on how the structure of a front influences phenomena often correlated to frontal activity, such as extreme precipitation.

Further work may include other improvements in the pipeline to reduce false positives, such as adaption of other frontal features. Currently, the algorithm is limited to expanding surface fronts, which is limiting when the frontal shape varies with increasing height. Additionally, the GPU version, does not contain a filter for weak or vanishing fronts. As such, the result will always provide a potential front location, even though the front may not be present at high altitudes. This effect is likely not dominant, as our datasets are limited to $1000\ \mathrm{hPa}$ to $500\ \mathrm{hPa}$. While the CPU version employed a filter based on the gradient within the cross-sections, statistical results barely change between both methods. Nevertheless, future iterations should include better, more sophisticated methods to filter non-front pixel from the results. Especially, if higher altitudes are to be explored. For example, a filter based on statistical properties of the detected fronts could be used.

The provided frontal data could also be used to train a neural network, allowing a direct detection of fronts without the need of the intermediate surface front data. We do believe that creation of such a network would allow us to leverage the deep neural networks' ability to model more complex shapes and phenomena, which our numerical algorithm may not correctly reconstruct due to its inherent limitations. This could, as proven in the $2\mathrm{D}$ case, create clearer results and reduce computation time. In addition, it may be able to better generalize, i.e. frontal activity over challenging terrain.

Ideally, however, human annotated datasets should be created. Such datasets could not only be used to train deep learning networks, but also serve as a baseline to validate different methods for the detection of three-dimensional fronts.

Part III

Future work and Conclusion

15 Future work

The presented methods are all evaluated on ERA5 atmospheric data provided as a latitude longitude grid. In the future an expansion of these methods onto different datasets such as the DWD's ICON-EU dataset, which has a higher resolution, may be of interest. A possible challenge of these high resolution datasets may occur in the representation of a front, as the representation as a line may no longer be a sufficient approximation of the frontal zone. Especially as these zones may be more than $100\ \mathrm{km}$ wide. However, the human annotated datasets used in this work do not provide information about the complete frontal zone, which provides a challenge when trying to use machine learning approaches.

For the three-dimensional fronts, it might be of interest to skip the need for the intermediate two-dimensional fronts and instead directly output the $3D$ data using machine learning. However, this faces similar challenges as increasing the resolution, as there currently are no valid ground truth datasets, which would allow direct end-to-end learning. On the algorithmic side, we could extend and adjust the algorithm for the creation of three-dimensional fronts to utilize other or additional features to provide more robust results. We could research how the dynamical state index would fare as a score for the given task. Furthermore, we showed that baroclinity is maximal near or at the location of our identified fronts, which lets us believe that maximizing the baroclinity might be another good approach to identify the three-dimensional shape of fronts.

Up until now, all of our algorithms are limited to a single time step. As can be seen in the video supplement [@8], frontal movement over time is rather continuous, which makes tracking of fronts over time very feasible. While this information could be leveraged to research the evolution of frontal properties over time, we could also utilize it to improve the accuracy of our methods. Using multiple time steps as input could possibly reduce errors, as another criterion based on temporal consistency could be used. Further, we could use the continuous motion as a prior to better localize fronts within challenging terrain, such as near mountainous regions or over landmass in general.

Our current methods only detect fronts within a reanalysis dataset. Although, predicting the frontal position might be of interest, as this information could be

integrated into forecasts of either weather or potentially extreme weather or natural hazard prediction. To integrate this information into forecasts a network would need to be able to predict the weather fronts several hours in advance, which is more difficult than detection, as the network needs to implicitly learn how the atmospheric conditions change over time.

In recent years, neural networks capable of providing stable multi-hour forecasts have been developed and attracted public attention [33, 31, 6]. These networks showed impressive capabilities of being able to provide weather forecasts near or sometimes even better than the ECMWF's numerical forecast system IFS. Even though these models are highly capable of processing atmospheric data, they were designed to provide global weather forecasts. Nonetheless, the capability of these newer models might be leveraged for front detection as well. While training from scratch might be costly, it might suffice to simply use a pretrained model and adapt it to front detection rather than weather forecasting. As these models have shown to actually be able to forecast weather, these models might be used to provide forecasts of weather fronts, instead.

A problem regarding frontal detection is, however, the limited availability of human annotated and labelled data on a global scale. While our network can produce fronts globally, we cannot quantify the quality of these detections outside our weather service analysis regions. Providing globally annotated datasets could help to improve the evaluation of front detection algorithms. For example, novel algorithms, such as our presented method for three-dimensional fronts, could benefit from valid ground truth datasets to quantify their overall skill. This would aid in the identification of potential shortcomings of such algorithms, which could in return be leveraged to further improve or develop such methods. With regard to deep learning, we showed that training on multiple regions improved the detection quality, which is another reason why increasing the amount of available datasets would be beneficial.

16 Conclusion

Weather fronts are a ubiquitous structure found in the atmosphere. Their correlations to other important phenomena makes identifying and researching these structures an interesting topic. Being able to identify and predict frontal positions could lead to improvements in several areas, such as flight planning or natural hazard prediction.

In this work, we presented three novel algorithms for the processing and creation of frontal structures within atmospheric data grids. We restricted ourselves to the ERA5 reanalysis dataset, which is available globally at a one-hour time interval. Our deep learning model outperforms a common numerical method for the detection of $2D$ atmospheric fronts, highlighting its skill in detecting surface fronts, learned from human annotated datasets. The network is applicable globally, where we could show that the detected results are at least qualitatively appropriate. An important observation during the training of the network was the fact that using only labels from a single weather service was insufficient to provide a decent generalization to global data. This could be observed by the worse performance of the neural networks which were only trained on a single weather service dataset. Although it is still to be researched, whether this is caused by atmospheric processes, which may differ between various regions, e.g. due to orography, or simply because different weather services use different guidelines for the creation of surface fronts. In our implementation we provided a custom loss function, which allows our network to directly output the detected fronts as lines, similar to how weather services annotated their frontal data instead of providing areas covering the bias in the ground truth data. Due to its general nature, this loss may also be applied in similar cases, where lines are desired, such as contour detection in images.

We presented an application of this method, within which we showed how our detected fronts and extreme precipitation are connected using the full $0.25°$ resolution of the ERA5 dataset for this analysis. We could showcase a significant correlation of fronts and extreme precipitation within the midlatitudes and provide detailed information for different global regions.

In the second part, we presented our novel algorithm for the creation of three-dimensional frontal structures. To the best of our knowledge, this was the first

method to allow for a statistical evaluation of frontal structure based on their three-dimensional properties. We showed that our method is capable of providing three-dimensional fronts over the North Atlantic and that those fronts exhibit features commonly associated with atmospheric fronts, such as a thermal gradient or maximum baroclinity. We provided a first statistical evaluation regarding the average inclination for the different classes of fronts and were able to show that these inclinations match with theoretical expectations. This holds especially true for warm fronts, where we could reproduce the flat inclination commonly depicted for warm fronts. With a case study, we highlighted a possible filter to separate warm and cold front parts of occluded fronts near cyclonic activity. Applying this filter before evaluation improved the positioning of the occluded fronts in relation to the position of the maximum baroclinity. It further improved the expression of the average inclination, due to the better separation of warm and cold front parts. Furthermore, we provided an improved version of this algorithm, which runs on a GPU. This parallel implementation drastically reduces the overall runtime to less than 1 minute per month. We showcased that our implementation is bound by IO from the disk. Our pipeline approach efficiently overlaps computation and IO operations, leading to shorter runtimes. Omitting netCDF output, the algorithm tends to be compute-bound, effectively masking the IO.

The codes for both front detection and the creation of three-dimensional fronts are publicly available at `https://github.com/stnie/`.

Bibliography

[1] David Acuna, Amlan Kar, and Sanja Fidler. “Devil is in the Edges: Learning Semantic Boundaries from Noisy Annotations”. In: (2019) (cit. on p. 86).

[2] Gianpaolo Balsamo, Florence Rabier, Magdalena Balmaseda, et al. “Recent progress and outlook for the ECMWF Integrated Forecasting System”. In: *EGU23* EGU23-13110 (2023) (cit. on p. 1).

[3] A. A. Beckert, L. Eisenstein, A. Oertel, et al. “The three-dimensional structure of fronts in mid-latitude weather systems in numerical weather prediction models”. In: *Geoscientific Model Development* 16.15 (2023), pp. 4427–4450 (cit. on pp. 5, 7–9, 93, 139).

[4] Gareth Berry, Michael J. Reeder, and Christian Jakob. “A global climatology of atmospheric fronts”. In: *Geophysical Research Letters* 38.4 (2011) (cit. on pp. 5, 6, 25).

[5] James Betker, Gabriel Goh, Li Jing, et al. “Improving image generation with better captions”. In: *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf* (2023) (cit. on p. 14).

[6] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, et al. “Accurate medium-range global weather forecasting with 3D neural networks”. In: *Nature* 619.7970 (July 2023), pp. 533–538 (cit. on pp. 14, 152).

[7] James Biard and Kenneth Kunkel. “Automated detection of weather fronts using a deep learning neural network”. In: *Advances in Statistical Climatology, Meteorology and Oceanography* 5 (Nov. 2019), pp. 147–160 (cit. on pp. 7, 9, 26, 27, 31, 43, 92).

[8] E. Bitsa, H. Flocas, J. Kouroutzoglou, et al. “Development of a Front Identification Scheme for Compiling a Cold Front Climatology of the Mediterranean”. In: *Climate* 7.11 (2019) (cit. on p. 25).

[9] J. BJERKNES. “ON THE STRUCTURE OF MOVING CYCLONES”. In: *Monthly Weather Review* 47.2 (1919), pp. 95–99 (cit. on p. 1).

[10] Bogdan Bochenek, Zbigniew Ustrnul, Agnieszka Wypych, and Danuta Kubacka. “Machine Learning-Based Front Detection in Central Europe”. In: *Atmosphere* 12.10 (2021) (cit. on pp. 8, 9, 26, 92).

[11] Wolfgang Boehm and Andreas Müller. “On de Casteljau’s algorithm”. In: *Computer Aided Geometric Design* 16.7 (1999), pp. 587–605 (cit. on p. 115).

[12] David Bolton. “The Computation of Equivalent Potential Temperature”. In: *Monthly Weather Review* 108.7 (1980), pp. 1046–1053 (cit. on pp. 95, 107).

[13]Harold E. Brooks. “TORNADO-WARNING PERFORMANCE IN THE PAST AND FUTURE: A Perspective from Signal Detection Theory”. In: *Bulletin of the American Meteorological Society* 85.6 (2004), pp. 837–844 (cit. on p. 49).

[14]J. Catto, E. Madonna, H. Joos, I. Rudeva, and I. Simmonds. “Global Relationship between Fronts and Warm Conveyor Belts and the Impact on Extreme Precipitation”. In: *Journal of Climate* (2015) (cit. on pp. 11, 24).

[15]J. L. Catto and S. Pfahl. “The importance of fronts for extreme precipitation”. In: *Journal of Geophysical Research: Atmospheres* 118.19 (2013), pp. 10, 791–10, 801 (cit. on pp. 69, 70, 72, 74, 77, 80, 86, 92).

[16]Jennifer L. Catto and Andrew Dowdy. “Understanding compound hazards from a weather system perspective”. In: *Weather and Climate Extremes* 32 (2021), p. 100313 (cit. on pp. 11, 24, 92).

[17]Marilei Foss, Sin Chan Chou, and Marcelo Enrique Seluchi. “Interaction of cold fronts with the Brazilian Plateau: a climatological analysis”. In: *International Journal of Climatology* 37.9 (2017), pp. 3644–3659 (cit. on p. 26).

[18]ALFRED J. HENRY. “J. BJERKNES AND H. SOLBERG ON THE LIFE CYCLE OF CYCLONES AND THE POLAR FRONT THEORY OF ATMOSPHERIC CIRCULATION”. In: *Monthly Weather Review* 50.9 (1922), pp. 468–473 (cit. on p. 1).

[19]Hans Hersbach, Bill Bell, Paul Berrisford, et al. “The ERA5 global reanalysis”. In: *Quarterly Journal of the Royal Meteorological Society* 146.730 (2020), pp. 1999–2049 (cit. on pp. 27, 29).

[20]T D Hewson. “Objective fronts”. In: *Meteorological Applications* 5.1 (1998), pp. 37–65 (cit. on pp. 5, 6, 9, 25, 92, 110).

[21]T D Hewson. “Objective identification of frontal wave cyclones”. In: *Meteorological Applications* 4.4 (1997), pp. 311–315 (cit. on p. 92).

[22]Tim D. Hewson and Helen A. Titley. “Objective identification, typing and tracking of the complete life-cycles of cyclonic features at high spatial resolution”. In: *Meteorological Applications* 17.3 (Sept. 2010), pp. 355–381 (cit. on p. 24).

[23]L. Hoffmann, G. Günther, D. Li, et al. “From ERA-Interim to ERA5: the considerable impact of ECMWF’s next-generation reanalysis on Lagrangian transport simulations”. In: *Atmospheric Chemistry and Physics* 19.5 (2019), pp. 3097–3124 (cit. on pp. 1, 94).

[24]Pandora Hope, Kevin Keay, Michael Pook, et al. “A Comparison of Automated Methods of Front Recognition for Climate Studies: A Case Study in Southwest Western Australia”. In: *Monthly Weather Review* 142.1 (2014), pp. 343–363 (cit. on p. 25).

[25]B. J. Hoskins and F. P. Bretherton. “Atmospheric Frontogenesis Models: Mathematical Formulation and Solution”. In: *Journal of Atmospheric Sciences* 29.1 (1972), pp. 11–37 (cit. on p. 11).

[26]Yang Hu, Yi Deng, Yanluan Lin, et al. “Dynamics of the spatiotemporal morphology of Mei-yu fronts: an initial survey”. In: *Climate Dynamics* 56.9-10 (May 2021), pp. 2715–2728 (cit. on p. 24).

[27]IPCC. *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Vol. In Press. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press, 2021 (cit. on pp. 1, 92).

[28]Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. *pybind11 – Seamless operability between C++11 and Python*. https://github.com/pybind/pybind11. 2017 (cit. on p. 44).

[29]J. Jenkner, M. Sprenger, I. Schwenk, et al. "Detection and climatology of fronts in a high-resolution model reanalysis over the Alps". In: *Meteorological Applications* 17.1 (2010), pp. 1–18 (cit. on pp. 5, 6, 9, 25, 46, 92).

[30]Michael Kern, Tim Hewson, Andreas Schätler, Rüdiger Westermann, and Marc Rautenhaus. "Interactive 3D Visual Analysis of Atmospheric Fronts". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 1080–1090 (cit. on pp. 8, 9, 93).

[31]Thorsten Kurth, Shashank Subramanian, Peter Harrington, et al. "FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '23. Davos, Switzerland: Association for Computing Machinery, 2023 (cit. on p. 152).

[32]Ryan Lagerquist, Amy McGovern, and David John Gagne II. "Deep Learning for Spatially Explicit Prediction of Synoptic-Scale Fronts". In: *Weather and Forecasting* 34.4 (2019), pp. 1137–1160 (cit. on pp. 7–9, 27, 31, 41, 43, 49, 92).

[33]Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, et al. *GraphCast: Learning skillful medium-range global weather forecasting*. 2022 (cit. on pp. 14, 152).

[34]Mark G. Lawrence. "The Relationship between Relative Humidity and the Dewpoint Temperature in Moist Air: A Simple Conversion and Applications". In: *Bulletin of the American Meteorological Society* 86.2 (2005), pp. 225–234 (cit. on p. 95).

[35]Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV] (cit. on p. 15).

[36]Laura Mack, Annette Rudolph, and Peter Névir. *Identifying atmospheric fronts based on diabatic processes using the dynamic state index (DSI)*. 2022. arXiv: 2208.11438 [physics.ao-ph] (cit. on pp. 7, 9).

[37]O. Martius, S. Pfahl, and C. Chavalier. "A global quantification of compound precipitation and wind extremes". In: *Geophysical Research Letters* 43.14 (2016), pp. 7709–7717 (cit. on pp. 11, 24).

[38]Daisuke Matsuoka, Shiori Sugimoto, Yujin Nakagawa, et al. "Automatic Detection of Stationary Fronts around Japan Using a Deep Convolutional Neural Network". In: *SOLA* 15 (2019), pp. 154–159 (cit. on pp. 8, 9, 27, 43, 45, 53, 92).

[39]Ryan M. May, Kevin H. Goebbert, Jonathan E. Thielen, et al. "MetPy: A Meteorological Python Library for Data Analysis and Visualization". In: *Bulletin of the American Meteorological Society* 103.10 (2022), E2273–E2284 (cit. on pp. 62, 95).

[40] Fedor Mesinger, Geoff DiMego, Eugenia Kalnay, et al. “North American Regional Reanalysis”. In: *Bulletin of the American Meteorological Society* 87.3 (2006), pp. 343–360 (cit. on p. 27).

[41] National Weather Service. *National Weather Service Coded Surface Bulletins, 2003-*. Apr. 2019 (cit. on pp. 31, 87, 97).

[42] Catherine M. Naud, Anthony D. Del Genio, Mike Bauer, and William Kovari. “Cloud Vertical Distribution across Warm and Cold Fronts in CloudSat–CALIPSO Data and a General Circulation Model”. In: *Journal of Climate* 23.12 (2010), pp. 3397–3415 (cit. on p. 11).

[43] Peter Névir. “Ertel’s vorticity theorems, the particle relabelling symmetry and the energy?vorticity theory of fluid mechanics”. In: *Meteorologische Zeitschrift* 13.6 (Dec. 2004), pp. 485–498 (cit. on p. 7).

[44] S. Niebler, A. Miltenberger, B. Schmidt, and P. Spichtinger. “Automated detection and classification of synoptic-scale fronts from atmospheric data grids”. In: *Weather and Climate Dynamics* 3.1 (2022), pp. 113–137 (cit. on pp. 3, 92, 96, 99, 146).

[45] Stefan Niebler. *Front polylines extracted from DWD Maps*. Dec. 2021 (cit. on p. 87).

[46] Stefan Niebler. *FrontDetection*. Version 0.0.2. 2021 (cit. on pp. 29, 87).

[47] Stefan Niebler, Bertil Schmidt, Peter Spichtinger, and Holger Tost. “Scalable GPU-Enabled Creation of Three Dimensional Weather Fronts”. In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC ’24. , Zurich, Switzerland, Association for Computing Machinery, 2024 (cit. on pp. 3, 24).

[48] Stefan Niebler, Bertil Schmidt, Holger Tost, and Peter Spichtinger. “Automated Identification and Location of Three Dimensional Atmospheric Frontal Systems”. In: *Computational Science – ICCS 2023*. Ed. by Jiří Mikyška, Clélia de Mulatier, Maciej Paszynski, et al. Cham: Springer Nature Switzerland, 2023, pp. 3–17 (cit. on pp. 3, 24, 105, 109, 122).

[49] Annika Oertel, Maxi Boettcher, Hanna Joos, et al. “Convective activity in an extratropical cyclone and its warm conveyor belt – a case-study combining observations and a convection-permitting model simulation”. In: *Quarterly Journal of the Royal Meteorological Society* 145.721 (2019), pp. 1406–1426. eprint: `https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3500` (cit. on p. 139).

[50] OpenAI, : Josh Achiam, et al. *GPT-4 Technical Report*. 2023. arXiv: `2303.08774 [cs.CL]` (cit. on p. 14).

[51] Rhys Parfitt, Arnaud Czaja, and Young-Oh Kwon. “The impact of SST resolution change in the ERA-Interim reanalysis on wintertime Gulf Stream frontal air-sea interaction”. In: *Geophysical Research Letters* 44.7 (2017), pp. 3246–3254. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2017GL073028` (cit. on p. 11).

[52]Rhys Parfitt, Arnaud Czaja, Shoshiro Minobe, and Akira Kuwano-Yoshida. “The atmospheric frontal response to SST perturbations in the Gulf Stream region”. In: *Geophysical Research Letters* 43.5 (2016), pp. 2299–2306. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2016GL067723` (cit. on p. 11).

[53]Rhys Parfitt, Arnaud Czaja, and Hyodae Seo. “A simple diagnostic for the detection of atmospheric fronts”. In: *Geophysical Research Letters* 44.9 (2017), pp. 4351–4358 (cit. on p. 26).

[54]Adam Paszke, Sam Gross, Francisco Massa, et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Curran Associates, Inc., 2019, pp. 8024–8035 (cit. on p. 29).

[55]Stephan Pfahl and Heini Wernli. “Quantifying the Relevance of Cyclones for Precipitation Extremes”. In: *Journal of Climate* 25.19 (Oct. 2012), pp. 6770–6780 (cit. on p. 70).

[56]M. Rautenhaus, M. Kern, A. Schäfler, and R. Westermann. “Three-dimensional visualization of ensemble weather forecasts – Part 1: The visualization tool Met.3D (version 1.0)”. In: *Geoscientific Model Development* 8.7 (2015), pp. 2329–2353 (cit. on p. 93).

[57]Michael J. Reeder, Thomas Spengler, and Clemens Spensberger. “The Effect of Sea Surface Temperature Fronts on Atmospheric Frontogenesis”. In: *Journal of the Atmospheric Sciences* 78.6 (2021), pp. 1753–1771 (cit. on p. 11).

[58]R. J. Renard and L. C. Clarke. “Experiments In Numerical Objective Frontal Analysis”. In: *Monthly Weather Review* 93 (1965), pp. 541–556 (cit. on pp. 5, 6, 25).

[59]Bruno Zanetti Ribeiro, Marcelo Enrique Seluchi, and Sin Chan Chou. “Synoptic climatology of warm fronts in Southeastern South America”. In: *International Journal of Climatology* 36.2 (2016), pp. 644–655 (cit. on p. 26).

[60]Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: `2112.10752 [cs.CV]` (cit. on p. 14).

[61]Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: `1505.04597 [cs.CV]` (cit. on pp. 16, 27, 37).

[62]F Sanders. “A proposed method of surface map analysis”. In: *Monthly Weather Review* 127.6, 1 (June 1999), pp. 945–955 (cit. on p. 24).

[63]P. G. Sansom and J. L. Catto. “Improved objective identification of meteorological fronts: a case study with ERA-Interim”. In: *Geoscientific Model Development Discussions* 2022 (2022), pp. 1–19 (cit. on pp. 5–7, 9, 110).

[64]Sebastian Schemm, Irina Rudeva, and Ian Simmonds. “Extratropical fronts in the lower troposphere–global perspectives obtained from two automated methods”. In: *Quarterly Journal of the Royal Meteorological Society* 141.690 (2015), pp. 1686–1698 (cit. on pp. 5, 6, 9, 11, 24, 25, 27, 46, 92, 107, 110).

[65] Sebastian Schemm, Michael Sprenger, and Heini Wernli. “When during Their Life Cycle Are Extratropical Cyclones Attended by Fronts?” In: *Bulletin of the American Meteorological Society* 99.1 (2018), pp. 149–165 (cit. on pp. 24, 92, 94, 140).

[66] David M. Schultz, Daniel Keyser, and Lance F. Bosart. “The Effect of Large-Scale Flow on Low-Level Frontal Structure and Evolution in Midlatitude Cyclones”. In: *Monthly Weather Review* 126.7 (1998), pp. 1767–1791 (cit. on pp. 1, 140).

[67] David M. Schultz and Geraint Vaughan. “Occluded Fronts and the Occlusion Process: A Fresh Look at Conventional Wisdom”. In: *Bulletin of the American Meteorological Society* 92.4 (2011), pp. 443–466 (cit. on p. 109).

[68] Uwe Schulzweida. *CDO User Guide*. Oct. 2019 (cit. on p. 30).

[69] Skipper Seabold and Josef Perktold. “statsmodels: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*. 2010 (cit. on p. 70).

[70] N. P. Shakina. “Identification of zones of atmospheric fronts as a problem of postprocessing the results of numerical prediction”. In: *Russian Meteorology and Hydrology* 39.1 (2014), pp. 1–10 (cit. on p. 25).

[71] M. A. Shapiro and Daniel Keyser. “Fronts, Jet Streams and the Tropopause”. In: *Extratropical Cyclones: The Erik Palmén Memorial Volume*. Ed. by Chester W. Newton and Eero O. Holopainen. Boston, MA: American Meteorological Society, 1990, pp. 167–191 (cit. on pp. 1, 139, 140).

[72] Evan Shelhamer, Jonathan Long, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651 (cit. on p. 27).

[73] Ian Simmonds, Kevin Keay, and John Arthur Tristram Bye. “Identification and Climatology of Southern Hemisphere Mobile Fronts in a Modern Reanalysis”. In: *Journal of Climate* 25.6 (2012), pp. 1945–1962 (cit. on pp. 6, 9, 11, 25).

[74] Carl M. Thomas and David M. Schultz. “Global Climatologies of Fronts, Airmass Boundaries, and Airstream Boundaries: Why the Definition of “Front” Matters”. In: *Monthly Weather Review* 147.2 (Feb. 2019), pp. 691–717 (cit. on pp. 11, 24).

[75] Carl M. Thomas and David M. Schultz. “What are the Best Thermodynamic Quantity and Function to Define a Front in Gridded Model Output?” In: *Bulletin of the American Meteorological Society* 100.5 (May 2019), pp. 873–896 (cit. on pp. 6, 11, 24, 110).

[76] LW Uccellini, SF Corfidi, NW Junker, PJ Kocin, and DA Olson. “Report On The Surface-Analysis Workshop Held At The National-Meteorological-Center - 25-28 March 1991”. In: *Bulletin of the American Meteorological Society* 73.4 (Apr. 1992), pp. 459–472 (cit. on p. 24).

[77] R. T. Williams and Jon Plotkin. “Quasi-Geostrophic Frontogenesis”. In: *Journal of Atmospheric Sciences* 25.2 (1968), pp. 201–206 (cit. on p. 11).

Webpages

[@1]Christian Kühnlein (ECMWF), Till Ehrengruber (CSCS), Stefano Ubbiali, et al. *ECMWF collaborates with Swiss partners on GPU porting of FVM dynamical core*. URL: https://www.ecmwf.int/en/newsletter/175/news/ecmwf-collaborates-swiss-partners-gpu-porting-fvm-dynamical-core (visited on Mar. 28, 2024) (cit. on p. 2).

[@2]Mihai Alexe, Zied Ben Bouallegue, Matthew Chantry, et al. *ECMWF unveils alpha version of new ML model*. URL: https://www.ecmwf.int/en/about/media-centre/aifs-blog/2023/ECMWF-unveils-alpha-version-of-new-ML-model (visited on Mar. 28, 2024) (cit. on p. 2).

[@3]DWD. *Deutscher Wetterdienst*. 2021. URL: https://www.dwd.de/ (visited on Dec. 15, 2021) (cit. on pp. 34, 36).

[@4]DWD. *Deutscher Wetterdienst, Wetter- und Klimalexikon*. 2021. URL: https://www.dwd.de/DE/service/lexikon/Functions/glossar.html?nn=103346&lv2=101518&lv3=101618 (visited on June 10, 2024) (cit. on p. 11).

[@5]ECMWF. *L137 model level definitions*. 2021. URL: https://www.ecmwf.int/en/forecasts/documentation-and-support/137-model-levels (visited on May 18, 2021) (cit. on p. 29).

[@6]ECMWF. *Supercomputer facility*. URL: https://www.ecmwf.int/en/computing/our-facilities/supercomputer-facility (visited on Mar. 28, 2024) (cit. on p. 2).

[@7]Hans Hersbach, Bill Bell, Paul Berrisford, et al. *Characteristics of ERA5 and innovations for ERA6*. URL: https://climate.copernicus.eu/sites/default/files/2022-09/S3_Hans_Hersbach_v1.pdf (visited on Mar. 28, 2024) (cit. on p. 1).

[@8]Stefan Niebler. *Detected Fronts January 2016*. 2021. URL: https://av.tib.eu/media/54716 (visited on Dec. 15, 2021) (cit. on pp. 40, 51, 87, 151).

[@9]NVIDIA. *CUDA C++ Best Practices Guide v12.6*. URL: https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/ (visited on Aug. 20, 2024) (cit. on p. 20).

[@10]NVIDIA. *CUDA C++ Programming Guide v12.4*. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (visited on Mar. 28, 2024) (cit. on p. 20).

[@11]NVIDIA. *NVIDIA H100 Tensor Core GPU Architecture*. URL: https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper (visited on Mar. 28, 2024) (cit. on p. 19).

[@12]*Synoptic Meteorology*. URL: https://www.noaa.gov/jetstream/synoptic (visited on Mar. 28, 2024) (cit. on p. 11).

[@13]Gauß-Allianz e. V. *Deutscher Wetterdienst*. URL: https://gauss-allianz.de/de/profile/Deutscher%20Wetterdienst (visited on Mar. 28, 2024) (cit. on p. 2).

Acronyms

CPU central processing unit. vi, 1–3, 18, 19, 47, 91, 94–96, 102, 105, 119, 121–124, 127, 131, 132, 134, 137, 138, 141, 142, 169, 170

CUDA Compute Unified Device Architecture. 3, 9, 18, 19, 91, 114, 119, 122, 141, 142, 145

DSI dynamical state index. 7, 151

DWD Deutscher Wetterdienst. 8, 97, 99, 151

ECMWF European Centre for Medium-Range Weather Forecasts. 2, 93, 152

ERA5 ECMWF Reanalysis v5. 93, 94, 97, 151

GPU graphics processing unit. v, 2, 3, 9, 18–20, 38, 40, 47, 91, 94–96, 102, 105, 106, 111, 113, 119, 121–123, 127–134, 137, 138, 140–145, 154, 168–170, 172

NWS National Weather Service. 7, 8, 97, 99

TFP thermal front parameter. 5–7, 9, 171

List of Figures

List of Tables

Appendix A

A.1 CSI evaluation sketch

Fig. A.1 shows a sketch of the evaluation and comparison regions used during evaluation of the CSI scores for two exemplary outputs. The comparison region fully contains the evaluation region plus the darker shaded blue region, while the evaluation region consists only of the brighter blue shaded region. Parts of a front are green if the segment is within the evaluation region, red otherwise.

During evaluation, the front segments within the evaluation region of Output 1 (Panel a) are compared against the front segments within the comparison region of Output 2 (Panel b) and vice versa. That means, only the green segments are evaluated, while the combination of green and red segments serves as potential matches.

As front objects are determined within the comparison region, both green segments of front 1 (front a) are correctly identified as being parts of the same front. Not using the comparison regions, these segments would be counted as two separate fronts, potentially skewing the count of matched or unmatched fronts in the end. Additionally, this method allows the algorithm to correctly match front e against the bottom right red front of Output 1. As this front is not located within the evaluation region, e would have otherwise been falsely counted as unmatched. This method is therefore useful to reduce errors in the evaluation introduced by the cropping of the outputs.

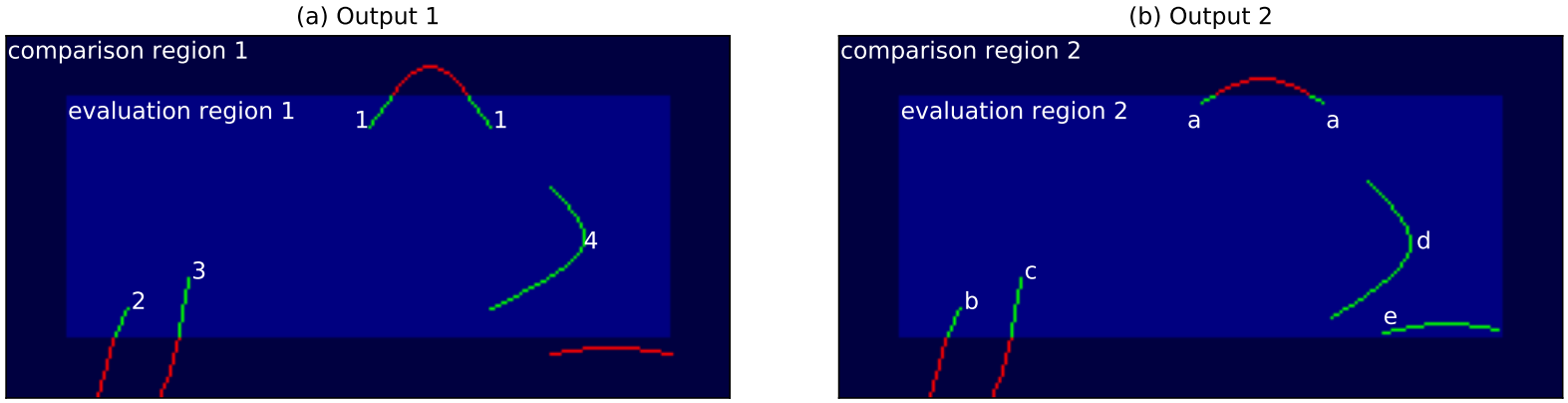


Fig. A.1.: Sketch of evaluation and comparison region used during CSI evaluation for two exemplary outputs (a) and (b). Green line segments are within the evaluation (and comparison) region, while red line segments can only be used during comparison but are not evaluated. Front segments connected within the comparison region are evaluated as a single front, even though they are not connected within the evaluation region alone.

A.2 Evaluation against individual fronts

We carried out an additional evaluation of the CSI, POD and SR scores, where each front is only allowed to match against one front of the corresponding type rather than the complete set. Table A.1 displays the scores for the DWD, Table A.2 for the NWS dataset.

Tab. A.1.: As Table 6.1 but matching can only happen against a single front.

Training region	NWS			DWD			Both		
	CSI	POD	SR	CSI	POD	SR	CSI	POD	SR
Binary	42.9 %	52.9 %	69.5 %	63.3 %	72.4 %	83.4 %	61.7 %	70.8 %	82.8 %
Warm	19.9 %	22.3 %	64.9 %	48.6 %	57.3 %	76.2 %	48.6 %	57.0 %	76.7 %
Cold	38.4 %	47.0 %	67.7 %	55.6 %	67.2 %	76.2 %	55.1 %	65.5 %	77.5 %
Occlusion	34.4 %	42.5 %	64.5 %	50.8 %	67.6 %	67.1 %	51.1 %	65.1 %	70.3 %

Tab. A.2.: As Table 6.2 but matching can only happen against a single front.

Training region	NWS			DWD			Both		
	CSI	POD	SR	CSI	POD	SR	CSI	POD	SR
Binary	62.1 %	74.9 %	78.4 %	44.3 %	50.3 %	79.0 %	63.4 %	76.9 %	78.4 %
Warm	37.2 %	56.2 %	52.4 %	22.4 %	43.8 %	31.4 %	36.2 %	57.9 %	49.2 %
Cold	54.8 %	69.2 %	72.5 %	40.5 %	51.0 %	66.3 %	55.8 %	72.1 %	71.1 %
Occlusion	48.5 %	72.1 %	59.7 %	36.0 %	62.5 %	46.0 %	48.9 %	73.2 %	59.5 %
Stationary	42.3 %	56.7 %	62.5 %		–		40.7 %	53.1 %	63.5 %

The values of all three measures decrease in comparison to the evaluation of the fronts with matching to the complete set. For the SR and classification results, this effect is less pronounced.

A.3 Cross-sections on NWS data

Here, we present additional cross-sections of physical properties (Fig. A.2 and A.3) relative to the front position provided by the NWS dataset. Qualitatively, the same behavior as for the DWD fronts can be seen. Note that the NWS data contains stationary fronts, thus cross-sections for these are indicated here by a solid yellow line.

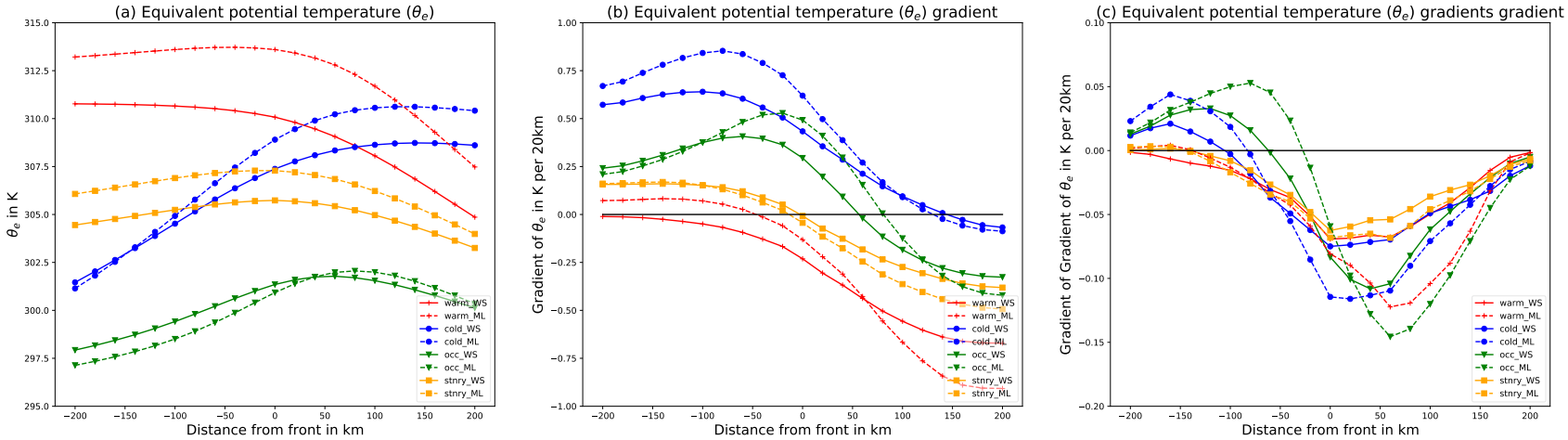


Fig. A.2.: Mean of (a) equivalent potential temperature (θ_e) and (b) θ_e gradient and (c) gradient θ_e gradient with front positions determined by NWS manual analysis (solid, WS) and by our network (dashed, ML). Physical quantities are evaluated at 850 hPa.

A.4 Video supplement

The provided video supplement shows the predicted and classified fronts for January 2016 at each hour. The background consists of the equivalent potential temperature at 850 hPa. Color channels are chosen as follows:

- red: warm front
- blue: cold front
- pink: occlusion
- green: stationary front
- yellow: unclassified front

Fronts are created as described in Section 5.4.2.

In some cases, a classification may not be exclusive for a pixel, resulting in a potential overlap in the color channels. This effect may occur when one type of front transitions into another. E.g. a transition from a warm to a cold front may appear pink, due to mixing in the blue and red color channel. Weakly expressed fronts may appear fragmented, due to the filtering threshold.

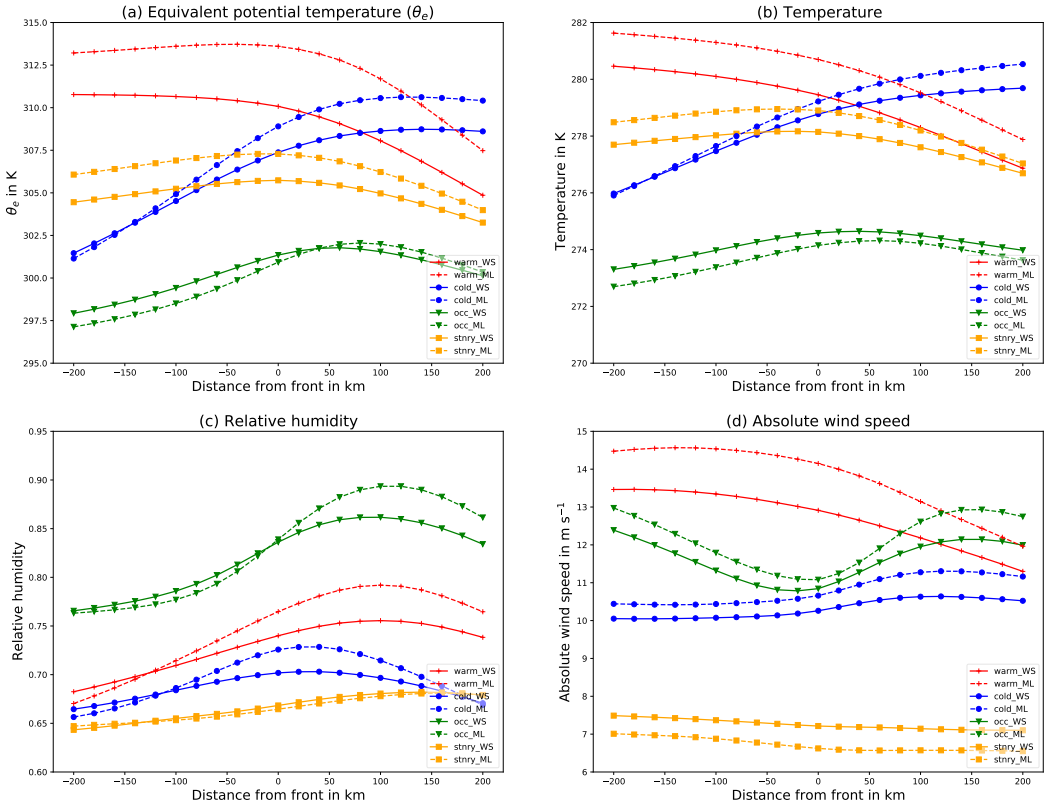


Fig. A.3.: Mean of (a) equivalent potential temperature (θ_e), (b) temperature, (c) relative humidity and (d) absolute wind speed with front positions determined by NWS manual analysis (solid, WS) and by our network (dashed, ML). Physical quantities evaluated at 850 hPa.

Declaration

I hereby declare that I have written the present thesis independently and without use of other than the indicated means. I also declare that to the best of my knowledge all passages taken from published and unpublished sources have been referenced. The paper has not been submitted for evaluation to any other examining authority nor has it been published in any form whatsoever.

Mainz, September 11, 2024

Stefan Niebler