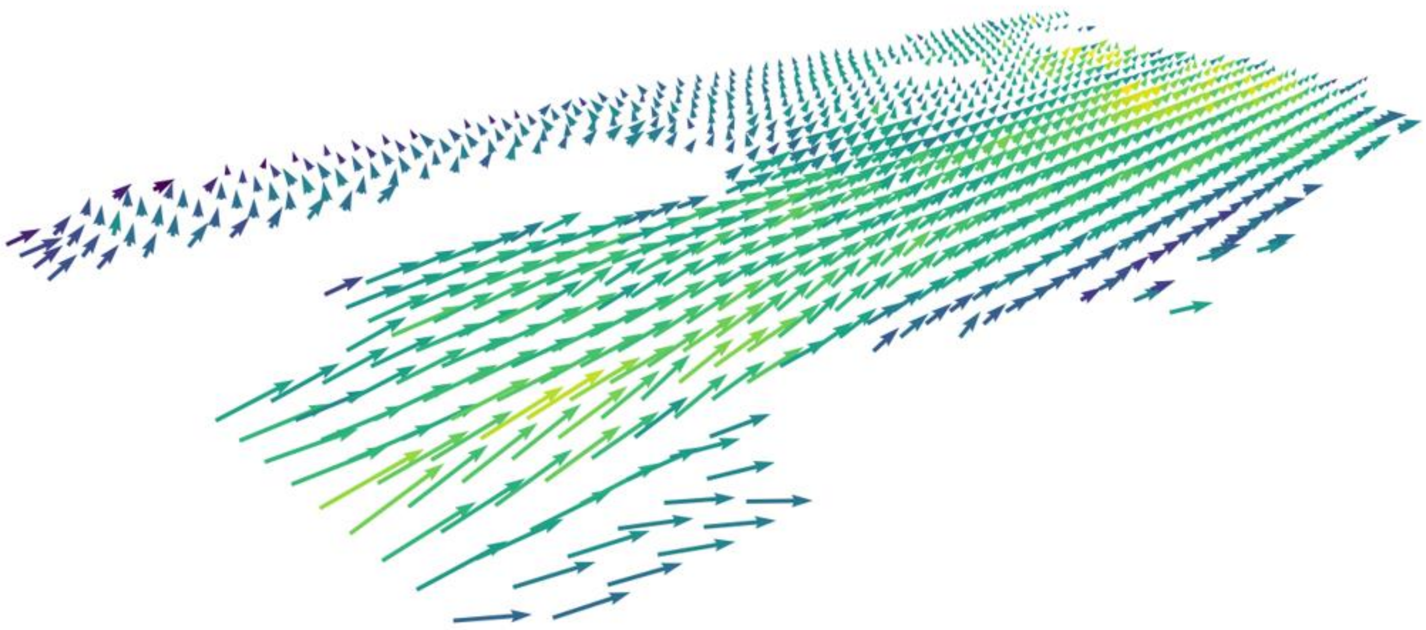
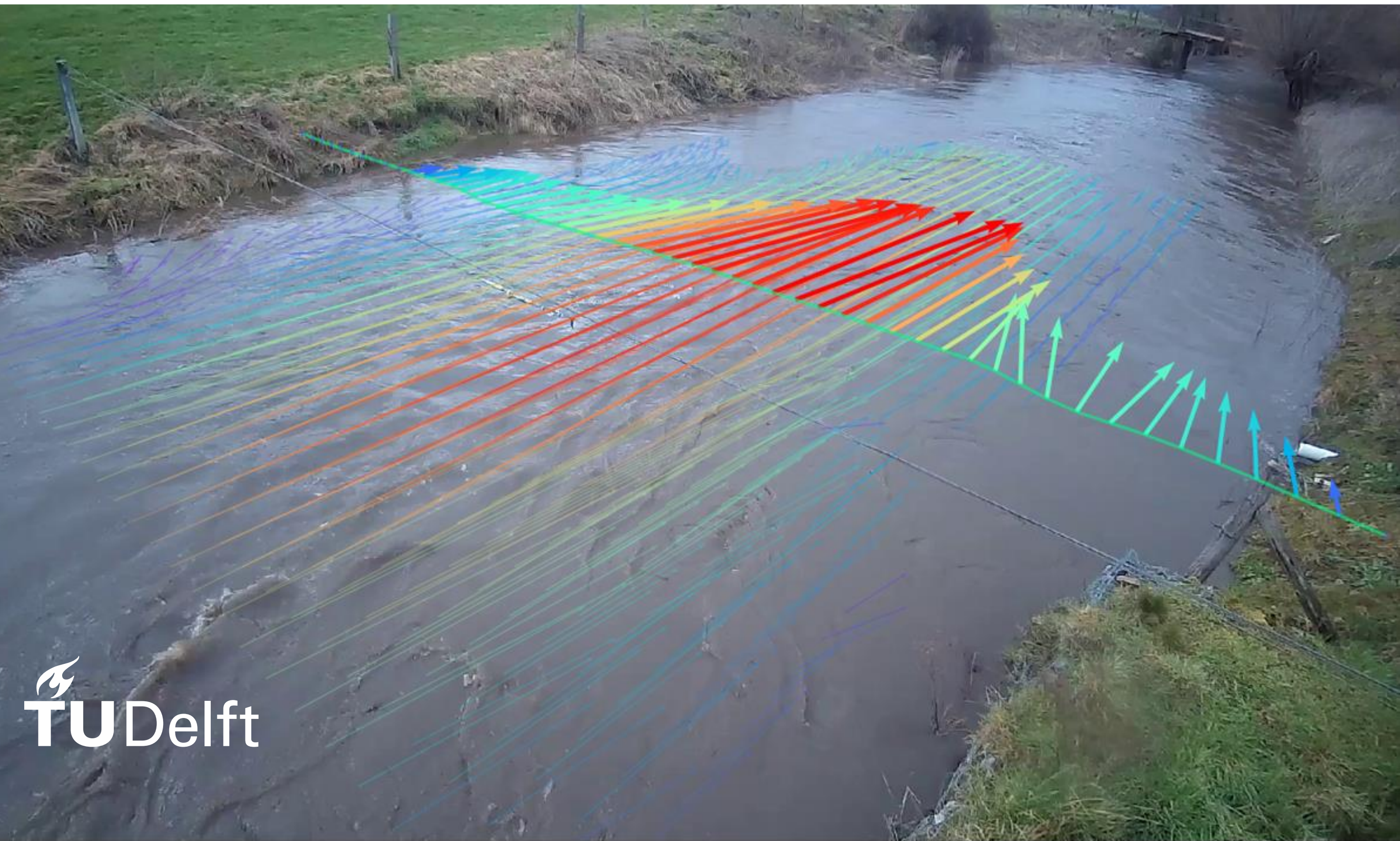
Improving detection of river surface flow using pyOpenRiverCam and AI augmentation

MSc Thesis Civil Engineering

M. Helmich





Improving detection of river surface flow using pyOpenRiverCam and AI augmentation

By

M.Helmich

in partial fulfilment of the requirements for the degree of

Master of Science

in Environmental Engineering

at the Delft University of Technology, Faculty of Civil Engineering,

to be defended publicly on Friday, April 12, 2024 at 15:00.

Supervisors:	Dr. Riccardo Taormina
	Dr. Hessel Winsemius
Thesis committee:	Dr. Xueqin Chen
	Dr. Hessel Winsemius

Preface

In this thesis, I delve into the fascinating realm of river flow analysis through the lens of artificial intelligence (AI) models. With a keen interest in harnessing the power of convolutional neural networks (CNNs) and U-Net architectures, I embarked on a journey to explore their efficacy in reconstructing datasets with missing velocity data. This endeavour stemmed from a profound curiosity about the intersection of AI and hydrology, driven by a desire to contribute to the advancement of scientific understanding in this field.

Throughout this research, I have encountered numerous challenges and triumphs, each contributing to my growth as a researcher and a scholar. From the meticulous analysis of datasets to the implementation of sophisticated AI algorithms, every step of the process has been a learning experience filled with valuable insights and discoveries. I hope that this thesis serves as a testament to the potential of AI in hydrological applications and inspires further exploration and innovation in this exciting field. May it spark curiosity, foster collaboration, and contribute to the collective pursuit of knowledge.

Abstract

This thesis investigates the efficacy of artificial intelligence (AI) models, particularly convolutional neural networks (CNNs) and U-Net architectures, in reconstructing datasets with missing velocity data in river flow analysis. Optical flow and Particle Image Velocimetry (PIV) techniques have emerged as valuable tools for analyzing river flow patterns.

Through a comprehensive literature review, CNNs, and U-Net models are identified as promising tools for this task due to their ability to capture intricate patterns in datasets. The study compares the performance of the U-Net model against a statistics-based hydrological benchmark model, revealing the superior performance of the U-Net model.

Furthermore, the analysis explores how the performance of AI models varies with differing quantities of missing data, by masking available data and comparing reconstructed values against the ground truth, highlighting the importance of data availability.

Additionally, the study investigates the influence of spatial patterns in training data on model performance, including patchy versus random missing data in the field of view, simulating more datasets more likely available in reality. This clarifies the challenges encountered in predicting grid points under different training dataset conditions.

Finally, the study identifies areas within the dataset that are particularly challenging to predict, shedding light on factors contributing to prediction errors. These findings underscore the potential of AI models in hydrological applications and provide valuable insights for future research in the field. Our findings show that U-net is capable of reconstructing velocity fields from a river flow better than an average benchmark, that uses the average values, with varying accuracy depending on input data. The average benchmark model had a relative error close to 0.2 in every instance, whereas the U-Net model showed relative errors ranging from 0.085 to 0.006. Errors from a patchy mask are ranging from 0.098 to 0.031.

Keywords

Artificial Intelligence (AI), Convolutional Neural Networks (CNNs), U-Net architectures, River flow analysis, Optical flow, Particle Image Velocimetry (PIV), Spatial patterns

Contents

1. Introduction

River discharge stands as a paramount variable in hydrological analysis. Initially recorded to anticipate floods and address water scarcity, river discharge time series have evolved to become essential for the design of hydrologic projects. In the present context, monitoring river discharge serves not only in flood anticipation but also aids in detecting climatic and environmental changes. This is due to the fact that the discharge and quality of river water are intricately linked to numerous climatic, biological, geological, and topographic variables within the basin. The ongoing shift in climate is modifying the distribution pattern of atmospheric precipitation both temporally and spatially, alongside altering the occurrence of extreme climatic events. [1]

Global advancements in river gauging networks are crucial to unveiling hydrological trends and changing atmospheric patterns. To have a greater understanding of the discharge patterns, it is crucial to determine the river flow as accurately as possible. Exploration of different measuring and predicting techniques is pivotal for our understanding of hydrological processes in a changing climate. [2][3]

In the current landscape, there is an escalating demand for cost-effective methodologies that offer enhanced accuracy and reliability in river flow estimations, particularly with high spatial and temporal resolution. River discharge measurements during flood events are crucial for establishing reliable stage-discharge relationships. However, conventional measurement techniques pose numerous challenges, including unsafe field conditions and the potential for measurement inaccuracies, particularly in critical situations such as floods. During these events, obtaining precise measurements through traditional in situ methods becomes hazardous and challenging due to the rapidly changing conditions. Consequently, existing models, like hot wire anemometry and laser Doppler velocimetry, often struggle to provide accurate predictions during such times, emphasizing the urgency and importance of enhancing prediction accuracy.[4]

In response to these needs, systems that don't require personnel, such as unmanned aerial systems and fixed camera systems, emerge as essential systems for continuous observations.[5] Widely available and affordable image-based techniques are implemented to achieve these goals. Furthermore, image-based techniques typically require manual post-processing to extract velocity data from captured images, which can be time-consuming and labour-intensive. These limitations underscore the need for innovative approaches, such as Artificial Intelligence-driven methods, to overcome the challenges associated with image-based flow analysis and enhance the accuracy and efficiency of velocity field reconstructions.

1.1 Background and Context

Accurate and reliable predictions of river flow are essential for various applications, including flood forecasting, water resource management, and environmental planning. However, existing methods for acquiring river flow data face significant challenges. Firstly, the methodologies employed must be non-intrusive to mitigate risks to both personnel and equipment involved in the measurement process.

Furthermore, camera-based measurements, while useful for estimations, suffer from quality limitations including image compression artefacts, diminished performance in low-light conditions, and potential issues stemming from corrupted imaging.[6] They are also limited by factors such as particle image size, magnification and optical aberrations.[7] Cloudy or nighttime conditions, in particular, lead to a notable decline in the accuracy of results. Moreover, the lack of information regarding the riverbed's bottom profile poses a significant obstacle, necessitating techniques for depth estimation, possibly through the utilization of stereo-imagery.[8]

Image-based detection methods, although effective in many instances, introduce complexities that can impede accurate measurement in certain scenarios. Additionally, the determination of surface flow is contingent on a prior understanding of the riverbed's geometry. Addressing these challenges may improve the accuracy and reliability of river flow predictions, ultimately contributing to more effective water resource management and flood preparedness.[9]

PyOpenRiverCam, abbreviated as "pyorc," is a fully open-source program dedicated to conducting image-based river flow analysis. At its core, pyorc currently harnesses the capabilities of Large-scale Particle Image Velocimetry (LSPIV) for flow analysis, leveraging the robust OpenPIV library alongside reprojections and image pre-processing functionalities provided by OpenCV. [10]

To improve the capabilities of pyorc, there is an initiative to incorporate Large-scale Particle Tracking Velocimetry (LSPTV) and Space-Time Image Velocimetry (STIV). These extensions aim to address scenarios wherein conditions are less favourable for LSPIV, thereby ensuring pyorc's versatility across a spectrum of hydraulic monitoring scenarios. [10] Combining the velocity fields with a user-provided cross-section results in a calculation of de river flow. Further explanation for this will be given in 3.1 Case Study. Despite its advantages, PIV techniques have inherent limitations that can affect their effectiveness in flow analysis. One limitation is the spatial resolution of PIV measurements, which may not always be sufficient to capture fine-scale flow features or resolve small-scale turbulent structures.

1.2 Objectives

The primary objectives of this thesis are to develop models that are able to handle missing velocity data in areas with files unable to construct a complete velocity field due to varied conditions, such as little light, unclear lens due to droplets or videos taken at nighttime. The ultimate aim is to reconstruct datasets with precision, rendering them suitable for diverse applications, including the accurate calculation of discharge rates.

Reconstructing missing data in datasets can be approached in various ways and the choice of method often depends on the nature of the data and the specific characteristics of the problem. One of the common methods is a mean/mode imputation where missing values in the dataset are replaced with the mean or mode of the observed values in the same variable. This approach can be quick, but it may not capture complex patterns in the datasets. Recognizing the limitations of this technique, this study delves into more advanced techniques from the field of machine learning. One promising avenue involves leveraging convolutional neural networks (CNNs) due to their capability to capture complex spatial patterns and dependencies within data. CNNs are particularly well-suited for tasks involving image data, such as velocity fields in river flow analysis, as they excel at automatically learning hierarchical representations of features directly from the data. Their architecture includes convolutional layers that systematically extract features at different spatial scales, enabling them to effectively capture the intricate patterns present in the dataset. [11,12]

Ultimately, the study aims to demonstrate the practical utility of the developed models by predicting flow Patterns and discharge accurately. By showcasing the models' ability to predict flow patterns and discharge rates with precision, the research seeks to contribute to advancements in hydrological modelling and water resource management practices.

1.3 Significance of the Study

This research holds significant implications in the domain of river flow analysis, particularly concerning the application of artificial intelligence (AI) models in reconstructing datasets with missing velocity data. Unlike previous studies, which primarily focused on the importance of flow measurements, this investigation delves deeper into the efficacy of CNNs and U-Net architectures in this context.

One of the novel aspects of this study lies in its thorough comparison of AI models, particularly the U-Net model, against a statistics-based hydrological benchmark model. By rigorously evaluating the performance of these models, the research sheds light on the superiority of the U-Net model in reconstructing velocity fields from river flow data. This finding not only advances the understanding of AI's potential in hydrological applications but also highlights the practical advantages of employing sophisticated neural network architectures over traditional statistical approaches. The methodologies developed here could potentially be adapted and applied in contexts beyond river flow assessment, contributing to a broader spectrum of environmental monitoring and predictive modelling.[13]

Furthermore, this study explores the influence of varying quantities of missing data on AI model performance. By systematically masking available data and comparing reconstructed values against ground truth, the research underscores the role of data availability in model accuracy. Additionally, the investigation delves into the impact of spatial patterns in training data on model performance, distinguishing between patchy and random missing data scenarios. Such insights are valuable for refining AI-based approaches to river flow analysis and improving predictive capabilities in real-world scenarios.

The accurate detection of river surface flow bears direct implications for water resource management. Precise assessments and allocations of water resources are contingent on reliable data regarding river behaviour. Improving the accuracy of these measurements aids in optimizing resource allocation, which, in turn, impacts a wide range of stakeholders, from agricultural enterprises to urban water utilities.

Finally, an additional application arises in the event of significant geomorphological changes following a flood. A trained model could facilitate the identification of such alterations, signalling the need for recalibrating the system. This adaptive capacity ensures the continued relevance and accuracy of the predictive model in dynamic environmental conditions.

1.4 Research questions

In this section, we outline the primary research question and its corresponding sub-questions that guide our investigation into the application of artificial AI in river surface velocity reconstructions. The primary research question of this study is:

Research question: How effectively can AI be employed to augment missing velocimetry data in river surface velocity reconstructions?

This overarching question can be dissected into several sub-questions, each addressing specific facets of the research:

Sub-question 1: What kind of AI model has the best potential for reconstruction of a dataset?

Sub-question 2: Do AI models outperform a typical hydrology benchmark model?

Sub-question 3: To what extent does the performance of the AI model vary with differing quantities of missing data on the river surface?

Sub-question 4: What is the influence of spatially patchy versus random missing data on the performance of the model training?

Sub-question 5: Which areas are hardest to predict in the case study?

1.5 Thesis Organization

This thesis is structured into several chapters, each dedicated to specific aspects of the research. The literature review forms the foundation, providing an overview of current methodologies in retrieving river flow measurements. It delves into traditional approaches and recent advancements while discussing neural network concepts relevant to the study. Following this, the methodology chapter outlines the research's procedural framework. It details data collection methods, preprocessing steps, and the development of benchmark models and neural network architectures. The subsequent results chapter presents the findings from the analysis. It includes quantitative results, visualizations, and comparisons between different approaches, offering insights into the research outcomes. In the discussion chapter, the results are interpreted in the context of the research objectives. Patterns, trends, and discrepancies observed in the data are analyzed, along with any limitations encountered during the study. Furthermore, potential avenues for future research are suggested. The conclusion chapter summarizes the key findings and reiterates the study's significance. It reflects on the research questions posed initially and provides recommendations based on the outcomes. Finally, the appendix contains supplementary material such as additional figures, tables, or data supporting the main text's findings. It offers readers further context related to the research without being essential for understanding the core arguments.

2. Literature Review

2.1 Image-based Velocimetry Methods

River flow, as a fundamental aspect of hydrology, holds paramount importance in surface hydrology studies, yet direct measurement remains challenging. Velocimetry analysis methods play a critical role in understanding fluid dynamics, particularly in river contexts, by capturing the velocity field on the river surface and providing insights into water flow patterns. To derive depth-averaged velocity from surface velocity measurements, various methods are employed, such as multiplication with empirical coefficients or hydraulic methods based on the entropy maximization principle.

In the context of fluid dynamics and image analysis, the concept of image flow refers to the sequential transformation of images over time, reflecting changes in the underlying physical processes. This transformation can be described in terms of displacement fields, representing the movement of image elements or fluid particles between successive frames. The associated image-flow velocity field captures the rate of change of these displacement fields, providing insights into the dynamics of the system.[14]

2.1.1 Particle Tracking Velocimetry

Particle Tracking Velocimetry (PTV) is a technique used to track individual particles within a fluid over time to understand their movement and velocities. By following the trajectory of each particle through successive frames of recorded data, PTV provides detailed insights into how particles move within the fluid flow. This method allows researchers to capture complex flow behaviours with precision, making it valuable for studying dynamic fluid systems.[15]

In recent years, there has been a shift towards leveraging Large-Scale Particle Tracking Velocimetry (LSPTV) methods to enhance the spatiotemporal resolution of velocimetry datasets. LSPTV combines advanced imaging techniques with particle tracking algorithms to provide a comprehensive view of fluid motion. [16]

Image velocimetry techniques, particularly those based on LSPTV, have emerged as valuable tools for monitoring river flows using remote sensing platforms. While LSPTV methods have demonstrated significant advancements in recent years, there remains a need for comprehensive evaluation and comparison of different image velocimetry algorithms. In response to this need, performance assessments have been conducted.[17][18]

Developments in LSPTV have led to improved accuracy and coverage, due to the integration of algorithms for particle identification, tracking, and data post-processing. State-of-the-art LSPTV methods leverage advanced imaging technologies enabling the extraction of high-resolution velocity fields for detailed analysis of fluid dynamics.[19]

2.1.2 Particle Image Velocimetry

Particle Image Velocimetry (PIV), like PTV, has its roots in the basic observation of particle motion in fluid flows, a concept that can be traced back to ancient times. For instance, individuals observing the movement of debris on the surface of flowing water could intuitively grasp the notion of velocity. PIV represents a technique for precisely and quantitatively measuring fluid velocity vectors across numerous points simultaneously.[20] Notably, PIV has gained popularity due to its affordability and simplicity, allowing its application even from lightweight unmanned aerial vehicles.[21][22]

The utilization of cameras for river flow monitoring, facilitated by image velocimetry techniques, has expanded significantly. Large-scale particle image velocimetry (LSPIV) applies principles from classic PIV to large-scale fluvial conditions.[23] This approach has been successfully employed in various applications, including flash flood monitoring in the French Alps and hydraulic engineering assessments.[24]

The methodology of image velocimetry typically involves four main stages: image acquisition, pre-processing, analysis, and post-processing. During image acquisition, videos or image sequences of the river's surface are recorded, capturing surface features such as boils, ripples, or debris, which serve as natural seeding points. Artificial seeding materials like eco foam or wood chippings may also be introduced into the river channel to facilitate feature detection. These recorded images undergo pre-processing, analysis, and post-processing stages to generate velocity estimates.[25][15]

In the pre-processing phase of image velocimetry, several essential steps are undertaken to prepare the raw imagery for analysis. This includes tasks such as image orthorectification, which corrects for distortions caused by camera lens properties, perspective and terrain variations. [26] Additionally, lens distortions are removed to ensure an accurate representation of features in the image. Another critical aspect is image stabilization, which may be applied if relevant, to compensate for any motion or vibrations in the recording platform, ensuring consistent and stable imagery. [27]

During the velocimetry analysis phase, the processed images are subjected to algorithms that extract velocity information from the captured data. This process involves identifying and tracking features or particles within the images to determine their displacement between successive frames, thereby quantifying the flow velocity at different points in the field of view. Various techniques, such as correlation-based methods or feature-tracking algorithms, are commonly employed for this purpose. Additionally, interpolation methods may be utilized to estimate velocities at points where data is sparse or missing, enhancing the overall accuracy of the velocity field reconstruction.[14,28]

In the post-processing phase, the focus shifts to refining the calculated results. This stage aims to eliminate any spurious vectors or erroneous data points that may have been generated during the analysis. However, it's important to note that not all image velocimetry algorithms require extensive post-processing, as the quality of results may vary depending on the specific algorithm used and the characteristics of the input data.[21,29]

While PTV offers a more detailed understanding of particle movement and flow dynamics, it typically requires more computational resources and data processing compared to PIV. However, PIV's ability to provide simultaneous velocity measurements at numerous points makes it suitable for capturing overall flow patterns efficiently. In this thesis, we are making use of the PIV technique to construct the velocity fields.

2.2 Deep Learning and Neural Networks

Deep learning has emerged as a powerful tool, demonstrating success in tasks ranging from image and speech recognition to genomics. Central to the effectiveness of deep learning are deep neural networks, characterized by their layered structure, which enables them to learn increasingly abstract representations of raw input data at each layer. By leveraging multiple layers, deep neural networks can discern intricate structures within high-dimensional datasets.

For instance, when presented with an image represented as an array of pixel values, the initial layers of a deep neural network may specialize in identifying fundamental features such as edges in different orientations.[30] Subsequent layers then build upon these features, discerning more complex patterns and arrangements until a sophisticated hierarchy of features emerges, ultimately facilitating the recognition of complex objects like faces or road signs.[31]

While traditional deep learning approaches, including neural networks, have been instrumental in handling various types of data, including images, convolutional neural networks (CNNs) have emerged as a more efficient and effective solution for tasks involving image processing.[32]

The main difficulty of training these networks is that due to exponential growth or decay the gradients may either vanish or explode, which is especially problematic with learning sequences with long-term dependencies. [33] [34]

2.2.1 Convolutional Neural Networks (CNNs)

While traditional deep learning approaches, including neural networks, have been instrumental in handling various types of data, including images, CNNs have emerged as a more efficient and effective solution for tasks involving image processing. CNNs are specifically designed to process and analyze visual data, exploiting the spatial correlation present in images by using learnable filters and convolutional layers.[35] In hydrology, the application of CNNs and U-Nets has shown promise in various tasks, ranging from rainfall estimation to flood mapping and river flow predictions. For example, one notable application involves the integration of CNNs with the Incompressible Smoothed Particle Hydrodynamics method for predicting fluid pressure, circumventing the need to directly solve the pressure Poisson's equation. While prior attempts at employing CNNs for solutions within Eulerian formulations, known as the Eulerian CNN framework, have been limited in mesh-based methods, some approaches represent a significant advancement in the field.[36][37]

Additionally, CNNs were employed to predict river flows, with their results compared against those derived from parametric models revealing that CNN-based models yielded more favourable outcomes, underscoring the efficacy of neural network approaches in hydrological prediction tasks. [38]

The effectiveness of CNNs is rooted in their unique properties, including local connectivity and shared weights.[39] CNNs leverage local connectivity, meaning that neurons in each layer are only connected to a small subset of neurons in the preceding layer. This allows the network to focus on localized regions of the input data, facilitating the detection of patterns and features within those regions. [40]

Moreover, CNNs employ weight sharing across the input domain, allowing them to detect features invariant to translation, regardless of their precise location within the input data.[41] This weight-sharing scheme enables CNNs to efficiently recognize recurring patterns and features across the entire input domain, particularly beneficial in arrayed data like images or sequences. Rather than relying solely on specific grid locations for predictions, CNNs learn shared filters across the entire image, facilitating understanding of different features hierarchically. [42]This property is crucial when predicting velocities in a grid using CNNs, enhancing their effectiveness by capturing underlying relationships between neighbouring grid points.[43] Additionally, hierarchical learning of shared filters enables CNNs to discern velocity patterns and variations, contributing to accurate velocity predictions across diverse grid configurations.[44] Overall, weight sharing enhances the effectiveness of velocity prediction tasks by allowing CNNs to learn shared filters across entire images, which are then composed hierarchically to recognize very specific features.[45]

In the context of remote sensing and geospatial data analysis, CNNs have been applied to tasks such as land cover classification, change detection, and environmental monitoring. The ability of CNNs to automatically extract features from complex spatial data has contributed to their effectiveness in various Earth observation applications. [38][46][47]

2.2.2 U-Net Architecture

Originally proposed in 2015, U-Net is a specialized form of CNN tailored for semantic segmentation tasks, particularly in medical image analysis. The architecture of U-Net features a U-shaped structure, comprising an encoder-decoder pathway. The encoder spatial information through down-sampling operations, while the decoder reconstructs high-resolution feature maps through up-sampling.[40]

Despite its initial development for medical imaging, U-Net's unique architecture has found widespread use in other domains, including remote sensing and hydrological modelling. In applications such as river flow predictions and velocity imputation, U-Net excels at capturing intricate spatial dependencies and providing detailed segmentation, making it a valuable tool for image analysis tasks beyond its original scope.[48] [49][50] [51][52]

A standard CNN network comprises three fundamental types of layers: convolution layers, pooling layers, and fully connected layers.

Convolution layers are components of CNNs tasked with extracting features from input images or feature maps derived from previous layers. These layers employ learnable filters, consisting of arrays of weights, to convolve over the input data. Through this process, feature maps are generated via a series of independent convolutions, followed by nonlinear transformations. In a grid representing a velocity field, with each point showing the speed and direction of the flow. Convolutional layers are like filters that slide over this grid, examining small patches at a time. They're looking for specific patterns in the flow, such as areas of high speed or regions where the flow changes direction. These filters learn to detect these patterns during training, helping the network understand the underlying dynamics of the flow. Figure 1 shows an example of a convolution operation.[53]

Following convolutional layers, pooling layers, also known as subsampling layers, are often employed to reduce the dimensions of the feature maps. Common pooling operations include average pooling and max pooling, which apply local nonlinear functions to aggregate information within windows of the feature maps. This down-sampling process results in coarser feature maps, facilitating the extraction of high-level representations in subsequent convolutional layers and reducing the number of learnable parameters to optimize computation time and mitigate overfitting. Pooling layers help simplify the information gathered by the convolutional layers while maintaining the essential features of the velocity field, in a detailed grid showing the flow at every point. Pooling layers reduce the size of this grid by grouping nearby points together and summarizing their velocities. For example, max pooling might keep only the fastest velocity in each group, while average pooling calculates the average velocity. This down-sampling process retains the most critical aspects of the flow while making it easier for the network to process.

Fully connected layers typically constitute the final layers of a CNN model, consolidating information from all hidden units in the preceding layer to make ultimate decisions. These layers synthesize the extracted features into comprehensive representations, aiding in the model's decision-making process.[54] After the pooling layers, the network has extracted the essential features from the velocity field. Fully connected layers analyze these features to reconstruct the complete picture. They consider how different patterns of flow relate to each other and use that information to fill in any missing or incomplete parts of the velocity field. For instance, if the network detects a vortex pattern in one area, it might infer the presence of similar patterns nearby and fill in the velocities accordingly.

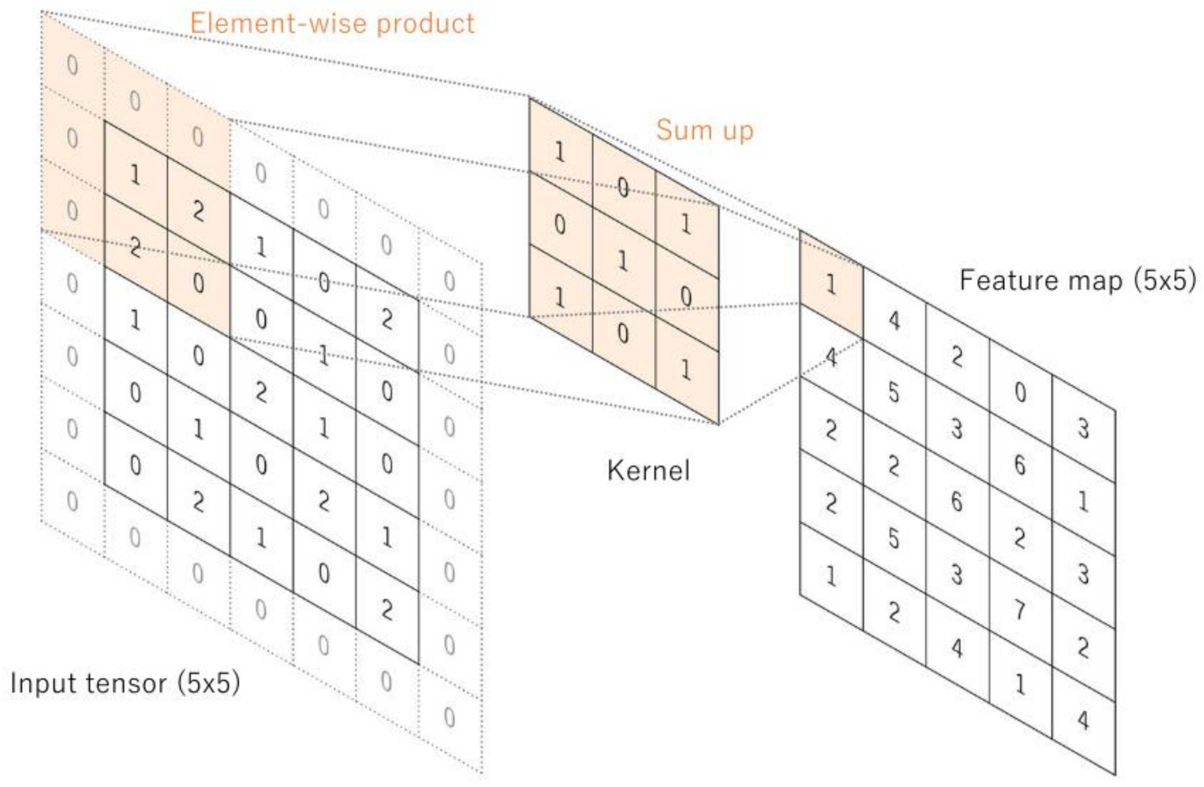


Figure 1 An example of a convolution operation [53]

The U-Net architecture is particularly suited for the task of reconstructing missing resolved flows due to its effectiveness in handling image-to-image tasks. In the context of this thesis, the problem is framed as transforming input images, which represent grids with known and missing velocities, into output images with reconstructed velocities. Unlike traditional RGB images, these input and output images consist of two different types of channels representing velocity components in the y and x directions. U-Net's ability to capture spatial dependencies and hierarchical representations of features makes it well-suited for such tasks, facilitating accurate reconstruction of missing velocity data within the grid structure. It is done by enhancing the typical CNN architecture by introducing an encoder-decoder structure and skip-connections.

The encoder-decoder structure is tailored for image-to-image tasks like velocimetry reconstruction. The encoder compresses the input flow field into a lower-dimensional representation, capturing the essential features of the flow. The decoder then expands this representation back to the original size while preserving the spatial relationships within the flow field. This architecture allows the network to learn both global and local features of the flow, facilitating accurate and detailed reconstruction.

Skip connections in U-Nets are vital for maintaining spatial information during the encoding and decoding process. They enable the network to bypass certain layers and directly connect encoder features to corresponding decoder features. This allows the network to retain fine-grained details of the flow field, ensuring that critical information is not lost during the compression and reconstruction process. By incorporating skip connections, U-Nets enhance the flow field reconstruction accuracy and mitigate issues like gradient vanishing or exploding, commonly encountered in deep learning models. [55][56]

In the context of reconstructing flow patterns, the U-Net architecture, with its encoder-decoder structure and skip connections, works together to capture both the overall picture and the finer details of the flow. This allows the network to understand the flow from different angles and preserves important features. By doing so, it effectively deals with the challenge of maintaining accuracy when processing lower-resolution flow data. This approach combines different types of information to make precise predictions, making U-Net a powerful tool for flow reconstruction compared to traditional CNN methods. In section 3.7 U-Net Model Construction, the U-Net architecture used for this thesis is visualized (Figure 17 U-Net Architecture visualized with the input and the output image for the velocity in the x directionFigure 17).

2.2.3 Training and evaluating a U-Net model

Training a U-Net model involves several key components, including selecting an appropriate optimizer, determining the learning rate, defining the loss function, and splitting the dataset into training and testing subsets. As one layer feeds its output into the next layer, the extracted features undergo a hierarchical refinement, progressively becoming more complex. This iterative refinement process is called the training of neural networks[53], where the goal is to optimize parameters, such as kernel weights, to minimize the disparity between model predictions and ground truth labels. This optimization is achieved through iterative updates using optimization algorithms like backpropagation and gradient descent, among others. These algorithms adjust the model parameters in a direction that minimizes the discrepancy between predicted outputs and the actual ground truth targets, gradually improving the network's performance over successive iterations of training.

The choice of optimizer is crucial as it determines how the model parameters are updated during the optimization process. For the task of reconstructing velocity fields, where the input data is sparse and the objective is to predict continuous values, such as velocity components, using a regression approach, we opt for the Adam optimizer combined with the Mean Squared Error (MSE) loss function.

The Adam optimizer is an adaptive learning rate optimization algorithm that computes individual learning rates for each parameter based on estimates of the first and second moments of the gradients. Adam incorporates bias correction and momentum, enhancing its performance compared to other optimization algorithms like RMSprop and Adagrad. This bias correction suggests that Adam outperforms RMSprop, especially towards the end of the optimization process when gradients become sparser. Given these advantages and the complexity of the U-Net architecture, Adam emerges as a suitable choice for optimizing the model parameters.[57] [32]

Furthermore, the MSE loss function is well-suited for regression tasks where the goal is to minimize the squared differences between predicted values and ground truth labels. In the context of reconstructing velocity fields, where the output consists of continuous velocity components, MSE provides a measure of how well the predicted velocities align with the actual velocities observed in the data.[58] Minimizing the MSE loss helps ensure that the model generates accurate predictions that closely match the true velocities, thereby facilitating the reconstruction of precise velocity fields. [59]

The Adam optimizer, with its adaptive learning rate capabilities and momentum, along with the MSE loss function, which is tailored for regression tasks, offer a good framework for training the U-Net to accurately reconstruct velocity fields from input data. [36]

The Rectified Linear Unit (ReLU) function is a commonly used activation function. It introduces non-linearity by setting all negative values in the input tensor to zero while leaving positive values unchanged. By allowing only positive values to pass through, ReLU helps the network learn complex patterns and relationships in the data. Essentially, ReLU decides which information is relevant to retain and which can be disregarded based on its positivity. [46][31]

The term "3x3 kernel" refers to a small filter matrix applied to input data. During convolution, this kernel moves across the input data, computing dot products with overlapping regions. By using a 3x3 kernel, the network can extract local features and patterns from the input data effectively. This is shown in Figure 1.

Padding involves adding additional border pixels around the input image or feature map. In convolutional layers, padding ensures that the spatial dimensions of input and output volumes remain consistent. With a padding of 1, one layer of zeros is added around the input image or feature map on all sides. This maintains the spatial integrity of the data and prevents information loss at the image borders during convolution.[60][61]

Max pooling is a down-sampling technique used to reduce the spatial dimensions of feature maps. It partitions the input feature map into non-overlapping regions and selects the maximum value from each region. Using a 2x2 kernel with a stride of 2 means that pooling is performed in 2x2 pixel regions, moving by 2 pixels at each step. Max pooling aids in achieving translation invariance and computational efficiency by reducing the spatial resolution of feature maps. Additionally, it helps in identifying dominant features and enhancing the network's robustness to variations in the input data.[32][62]

3. Materials and methods

The methodology employed in this study integrates various techniques to address the research objectives effectively. The process begins with data collection, where datasets containing river flow velocity measurements are gathered from the designated sources. Subsequently, preprocessing steps are undertaken to clean and prepare the data for analysis, ensuring its quality and integrity. Following this, mean velocity and missing values are determined for each grouped water level, providing essential insights into the dataset's characteristics. The study also involves the creation of jack-knifed data to simulate missing data and analyze model performances. Moreover, benchmark and U-Net models are constructed to compare different approaches to missing data reconstruction. Finally, a comprehensive comparison between the benchmark and U-Net models is conducted to evaluate their performance and effectiveness in addressing the research objectives.

3.1 Case Study

The scope of this study is centred around a case study conducted on a small river situated in the southern region of Limburg – The Netherlands (Figure 2 and 3). The selection of this specific location was deliberate, driven by practical considerations that enhance the feasibility and depth of the research: there already was a camera set-up prior to the start of this thesis and there were already more than 6000 datasets, ranging from December 18th 2022 to April 26th 2023, available on which we could perform the analysis. The richness of the dataset spanning multiple months offered a diverse range of conditions and variations in river dynamics, contributing to a robust and comprehensive analysis. Additionally, there were many recordings with a variety in the illumination and optically resolved percentages of the dataset.

It is important to note that while the study's geographical scope is limited to the specific river in Limburg, the findings and methodologies developed herein may have broader applicability to other river systems. The case study serves as an illustrative example, showcasing the practical implications and potential advancements in river flow predictions through the integration of AI techniques.

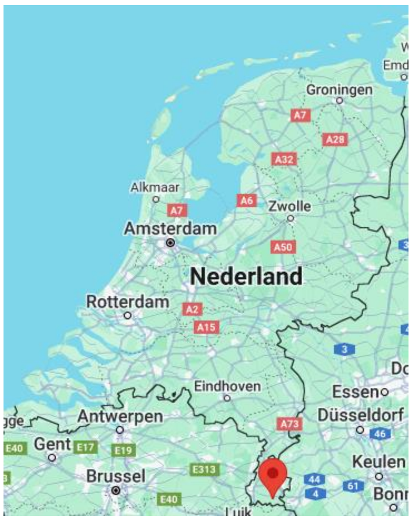


Figure 2 Location of the case study in Google Maps

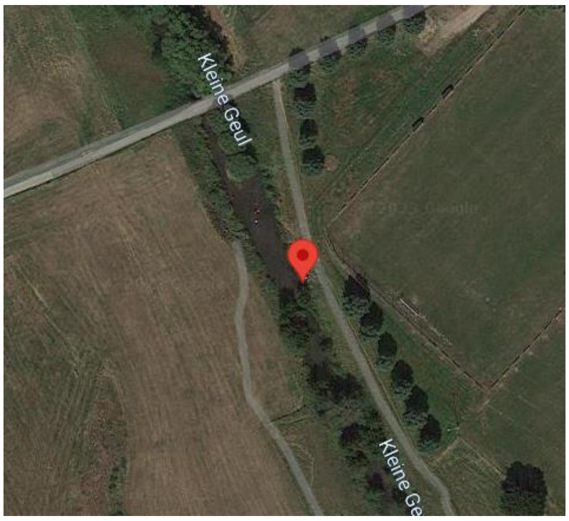


Figure 3 Satellite image of the ricer from the Case study

From a pre-defined camera configuration the control points, the camera lens position and the area of interest are determined (Figure 4) and with some preprocessing of the videos, a velocity field can be created. Figure 5 shows such a velocity field projected on the geographical plane.

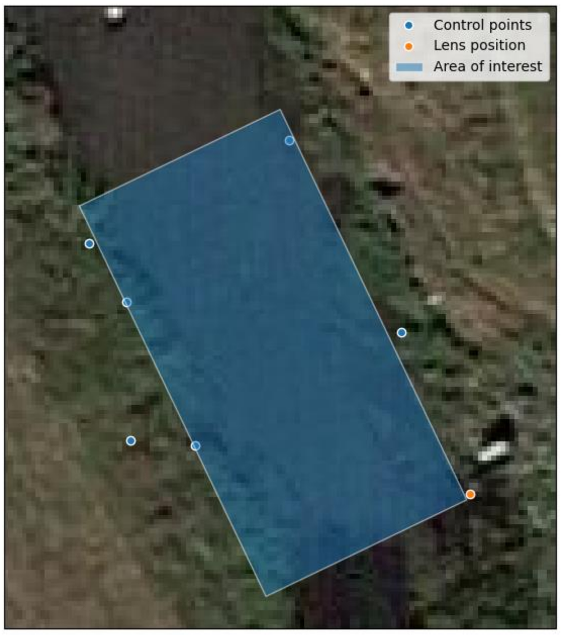


Figure 4 Area of interest with control points and camera

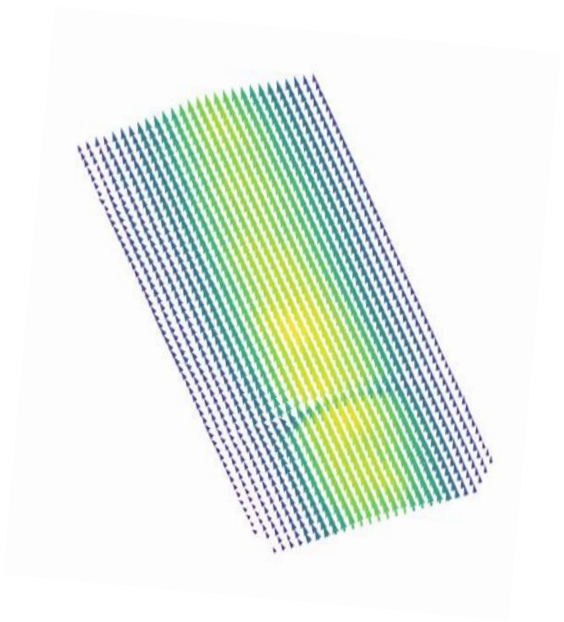


Figure 5 Velocity field in geographical plane

Transects can be extracted from the velocimetry with user-provided cross-sections (Figure 6). From this transect, the river flow can be calculated (**Error! Reference source not found.**7). In the initial case, gaps in the transection are filled by either a logarithmic profile fit or with zeros. This thesis is looking to reduce the gaps in the transection by completely reconstructing the velocity fields.

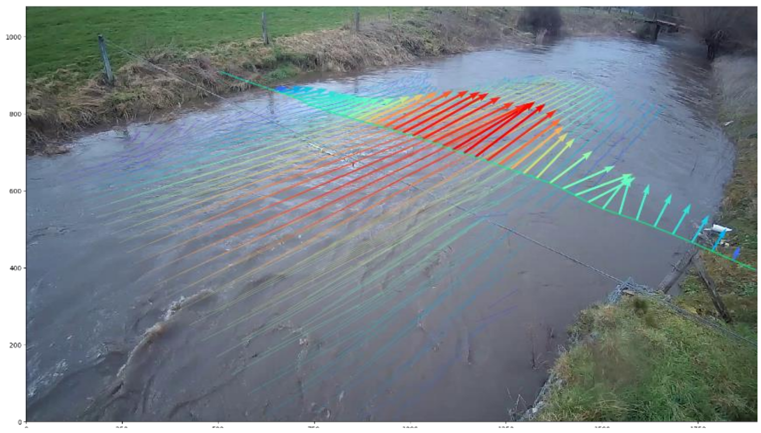


Figure 6 Velocity field from a camera with a transect of velocity

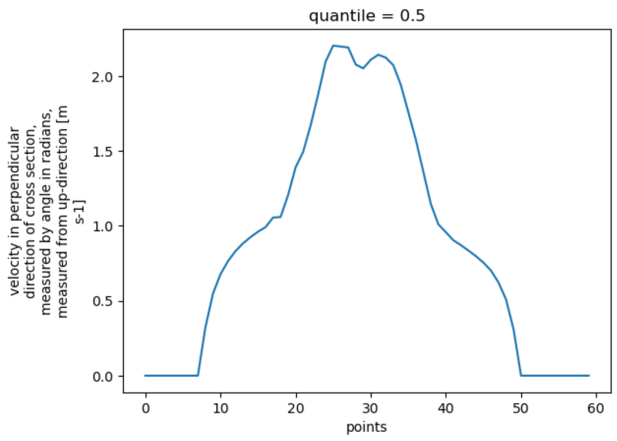


Figure 7 River flow obtained from the transect

3.2 Data Collection

The primary dataset for this study consists of 6000+ recordings obtained from pyOpenRiverCam (pyorc). [10] These recordings were captured using one stationary camera, and processed with one single configuration providing velocimetry in the form of grids [29x58]. Each grid contains flow velocity information in both the x and y directions. The recordings were collected at a rate of 125 frames per 5 seconds, resulting in a frame captured every 0.04 seconds. It is noteworthy that certain datasets exhibit missing data points on the grid, particularly in conditions of darkness or when the view is obscured. Figure 8 shows such a velocity field constructed on top of the image. It is clear that some part of the lens was obscured by a droplet which led to missing data in the velocimetry.

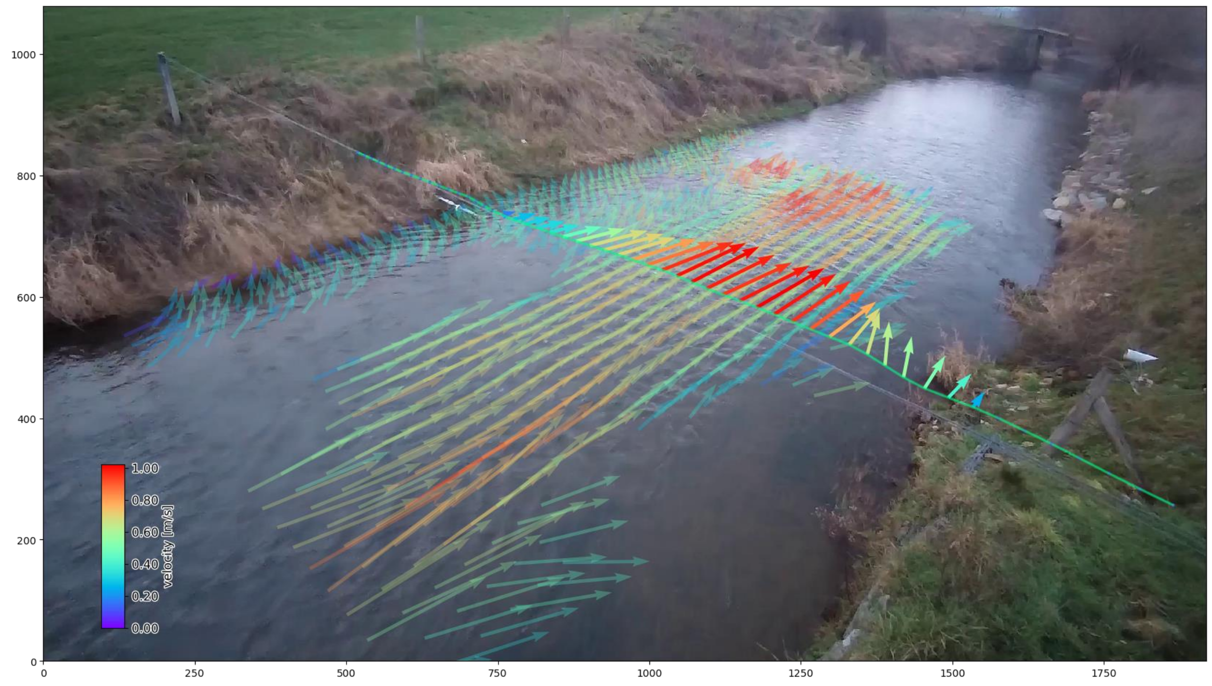


Figure 8 Image captured by the camera with the velocity field visualized on top of the image.

The velocimetry obtained from the video object provides raw velocities after orthorectification done by pyorc. This velocimetry information is structured and organized into an xarray dataset, a powerful data structure in Python designed for handling multi-dimensional labelled arrays. As illustrated in Figure 9, the projection represents a velocity field on a rectangular grid. These datasets are used throughout this thesis.

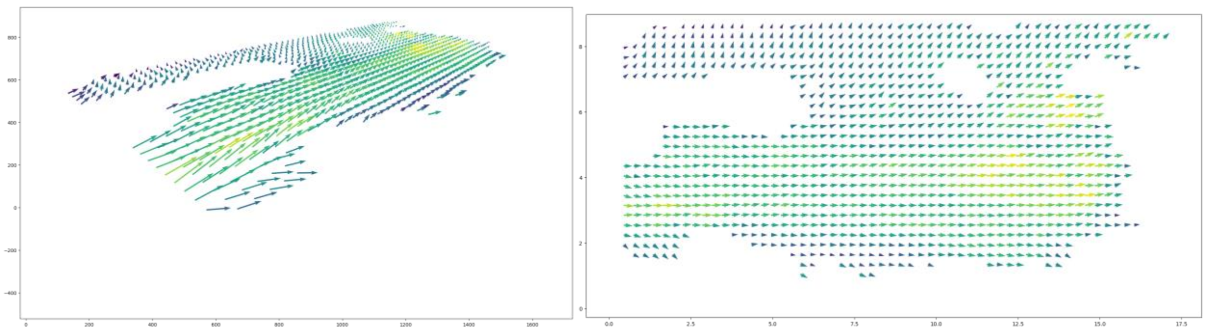


Figure 9 A velocimetry from the camera projected on a rectangle with missing data (left: camera view, right: rectangular projection)

3.3 Preprocessing

The preprocessing phase played a crucial role in refining the raw dataset before the development of the models. The initial step involved calculating the mean velocity at each grid point over the 5-second recording period. This process aimed to capture the central tendency of flow velocities and reduce the impact of transient fluctuations in the data. Additionally, to ensure the integrity of the dataset, velocities that did not align with physical principles were filtered out. This included instances of backward flows or velocities deemed impossible in the given context. The filtering process contributed to the elimination of erroneous data points and enhanced the overall quality of the dataset. Finally, the datasets were grouped based on the measured water level NAP, operating under the assumption that flows at similar water levels exhibit comparable behaviours. This grouping facilitated an analysis, allowing the AI models to capture variations in flow patterns specific to distinct water level ranges. [63] Figure 10 shows the overall distinction in flow velocity with different water levels across all the datasets.

Segmenting the velocimetry fields based on similar hydraulic conditions serves two primary purposes. Firstly, it ensures that comparisons between datasets are fair and meaningful by minimizing the influence of varying flow dynamics. By focusing on datasets with comparable hydraulic conditions, we can better isolate the effects of other factors on the analysis results, leading to more reliable conclusions. Secondly, segmentation helps streamline the training process by creating subsets of data with reduced complexity. By concentrating on specific ranges of water levels or flow conditions, we can simplify the training data, making it more manageable for model training. This approach enhances the efficiency of the training process and improves the overall performance of the models developed using the segmented datasets. Ultimately, segmentation based on hydraulic conditions enhances the quality and accuracy of analysis and predictions derived from velocimetry fields.[63][64] While segmenting the velocimetry fields based on hydraulic conditions enhances the quality and accuracy of analysis for the selected water level, caution must be exercised when applying these findings to different water level scenarios. In practice, the ultimate goal is to develop models capable of accurately predicting velocities across various water levels, necessitating further research and validation efforts beyond the confines of the current study.

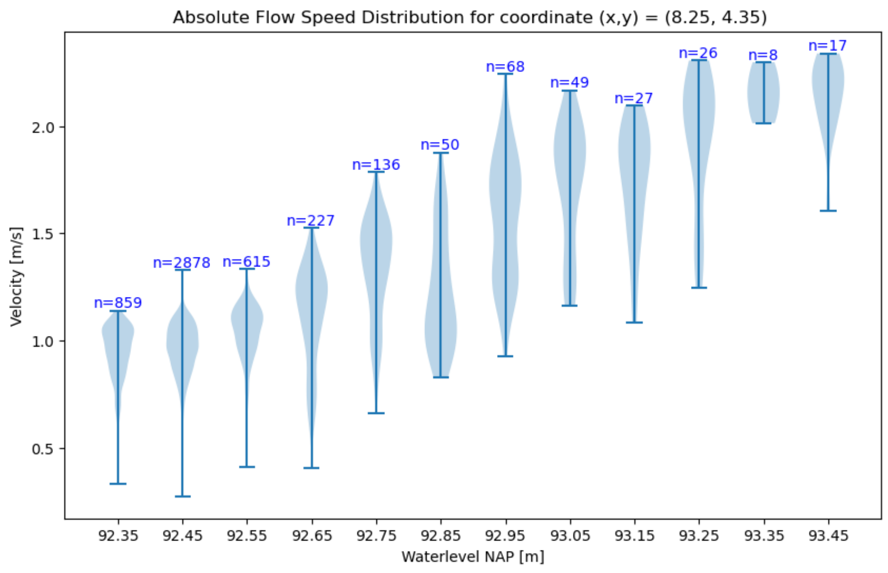


Figure 10 Flow velocities for different water levels in the dataset

3.4 Determination of mean velocity and missing values for each grouped water level

Prior to constructing the benchmark model, it was crucial to establish the mean velocity values for each grouped water level. This process entailed calculating the mean velocity at every grid point within each specific water level group. By averaging the velocities across all samples within the same grouped water level, we obtained mean velocities that will be used for reconstructing velocity fields for the benchmark model.

Furthermore, In our methodology, a key component was recognizing and examining missing values in the datasets. NaN (Not a Number) values occurring at each grid point indicate areas where data is either missing or undefined. These values commonly arise due to various during the data acquisition process. Locations with numerous NaN values present instances where the most improvement can be made, as these areas likely correspond to regions of high flow variability or measurement uncertainty. Conversely, grid points with low average NaN values likely contain valid velocity data across all datasets, suggesting regions with more stable flow conditions or reliable measurement outcomes. Therefore, understanding the distribution of NaN values across the velocity field is essential for identifying areas requiring further data refinement or interpolation techniques to enhance the overall quality and reliability of the velocimetry dataset. Figure 11 shows the average amount of missing values across all the datasets for water level 92.55m NAP. Appendix A: Average NaN values across all the datasets, gives a visualization across all the water levels.

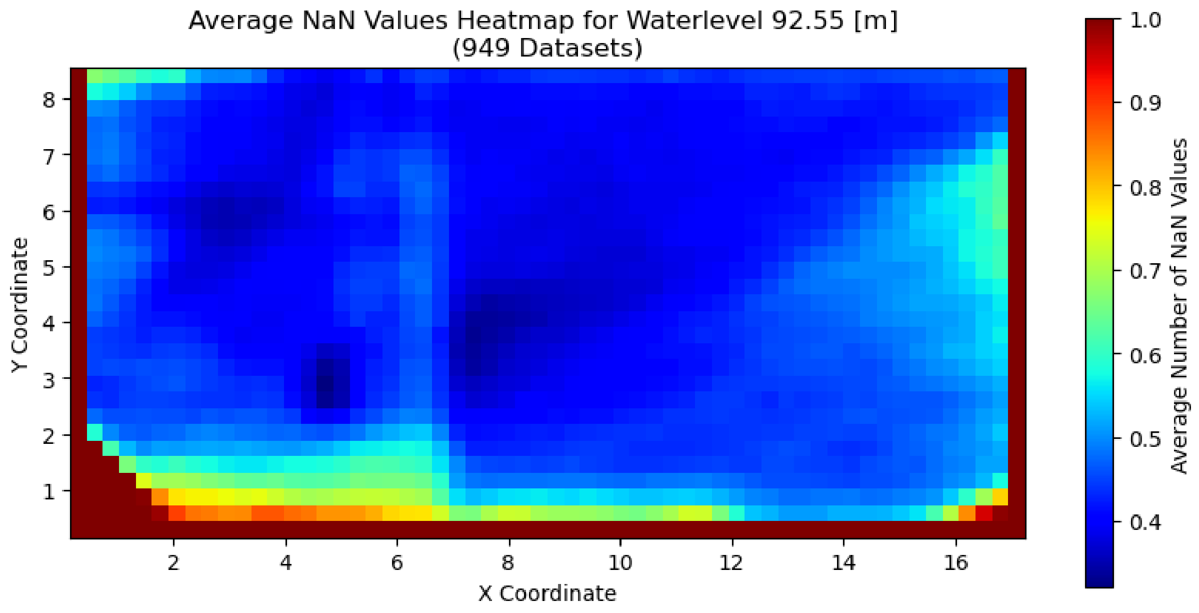


Figure 11 Visualization of the average NaN values in the datasets from water level 92.55m

In training the model, a subset of velocimetry fields centred around a specific water level, approximately 92.55 meters, was chosen for several reasons. Firstly, this subset contained a sufficient number of datasets, ensuring robust training, validation, and testing procedures. Adequate sample size is critical for training machine learning models, as it allows for meaningful pattern recognition and generalization

A specific subset of datasets was chosen where the optical velocimetry data provided information for at least 80% of the flow grid points. The reason for using only datasets with at least 80% of resolved data is to ensure a fair comparison between the reconstructed data and the ground truth. Even though the aim is to be able to predict velocimetry with very low input data, in the initial phase of training the model, it is important to have enough grid points where the ground truth is known so that it can be compared to the predicted value of the model. With datasets containing low optical resolution, it would indeed be possible to reconstruct them. However, without sufficient known ground truth data points, it becomes impossible to evaluate the accuracy of the model's predictions. This raises questions about the trustworthiness of the model's prediction.[59] Figure 12 visualizes the average amounts of missing data in the datasets with over 80% resolved data that will be used to train the U-Net model and evaluate the performance of both the benchmark and the U-Net model.

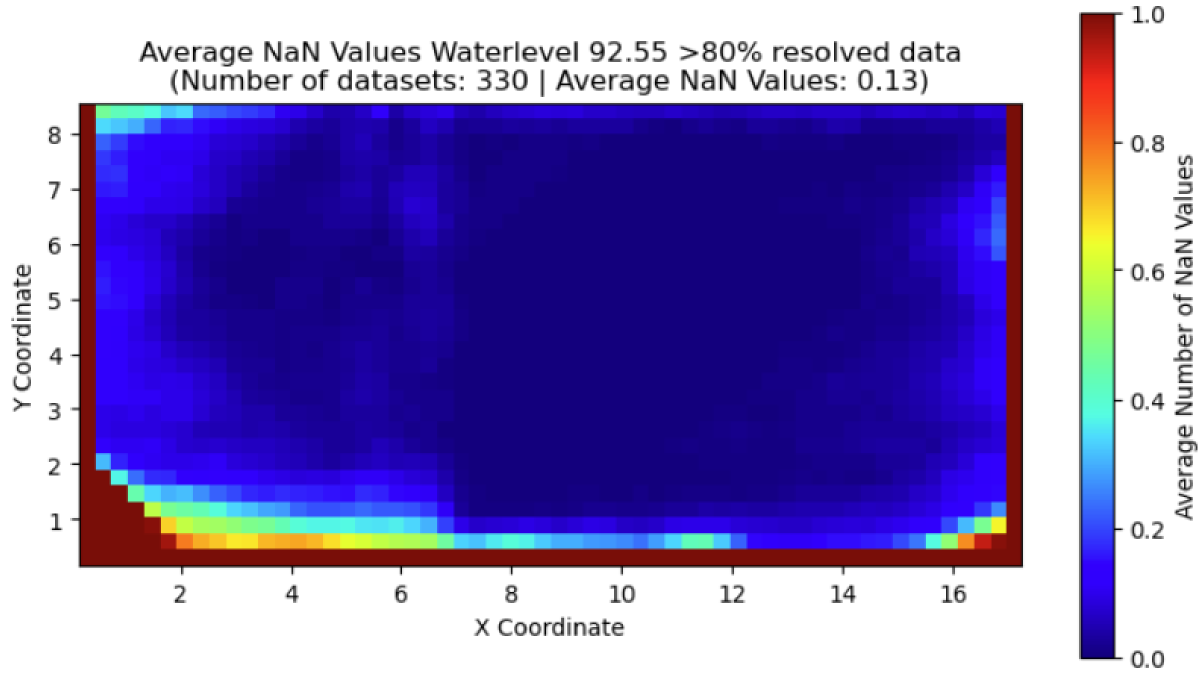


Figure 12 Visualization of the average NaN values, but only for over 80% resolved datasets

3.5 Masking the data

Missing data are an inherent aspect of observational records, arising from diverse factors such as equipment malfunction, measurement errors, or other external influences.[65]

During the data preparation for model training, it is necessary to adjust the input files to resemble the velocity fields intended for reconstruction. The model's performance will be evaluated by assessing its ability to predict the missing velocities within these fields accurately. To simulate real-world conditions, masking techniques are employed to generate velocimetry datasets with some data intentionally missing, thereby challenging the models to reconstruct them.

In this study, two distinct masking approaches are employed. The first method involves random masking, where data is systematically removed from the original velocimetry dataset in increments of 10%. Figure 13 shows such an approach where from one original complete velocimetry, multiple velocimetries are created with different percentages of the data available. While this method serves as a foundational proof of concept, it's recognized that real-world velocimetry datasets typically exhibit more structured patterns of missing data. For instance, certain areas of the velocimetry may have data available while others are obscured, in the case of a lens droplet affecting only specific regions of the data. Similarly, factors such as consistent lighting on a particular area of the river may result in more data points being present in those sections of the velocimetry.

A more advanced masking technique is employed to make the input velocimetry datasets more closely resemble real-world scenarios. In the second approach, instead of randomly removing data, patches were strategically chosen to mask based on their likelihood of containing missing values. By focusing on patches with fewer missing values on average, situations where data is consistently present across all the datasets were simulated, mirroring the typical distribution of missing values in actual velocimetry datasets. Figure 14 visualizes the modified datasets where values are removed in order of the likelihood of being available in the original dataset.

Masks were constructed using the gridpoints that still contained values, and these masks were utilized as inputs in both the benchmark model and the U-Net model.

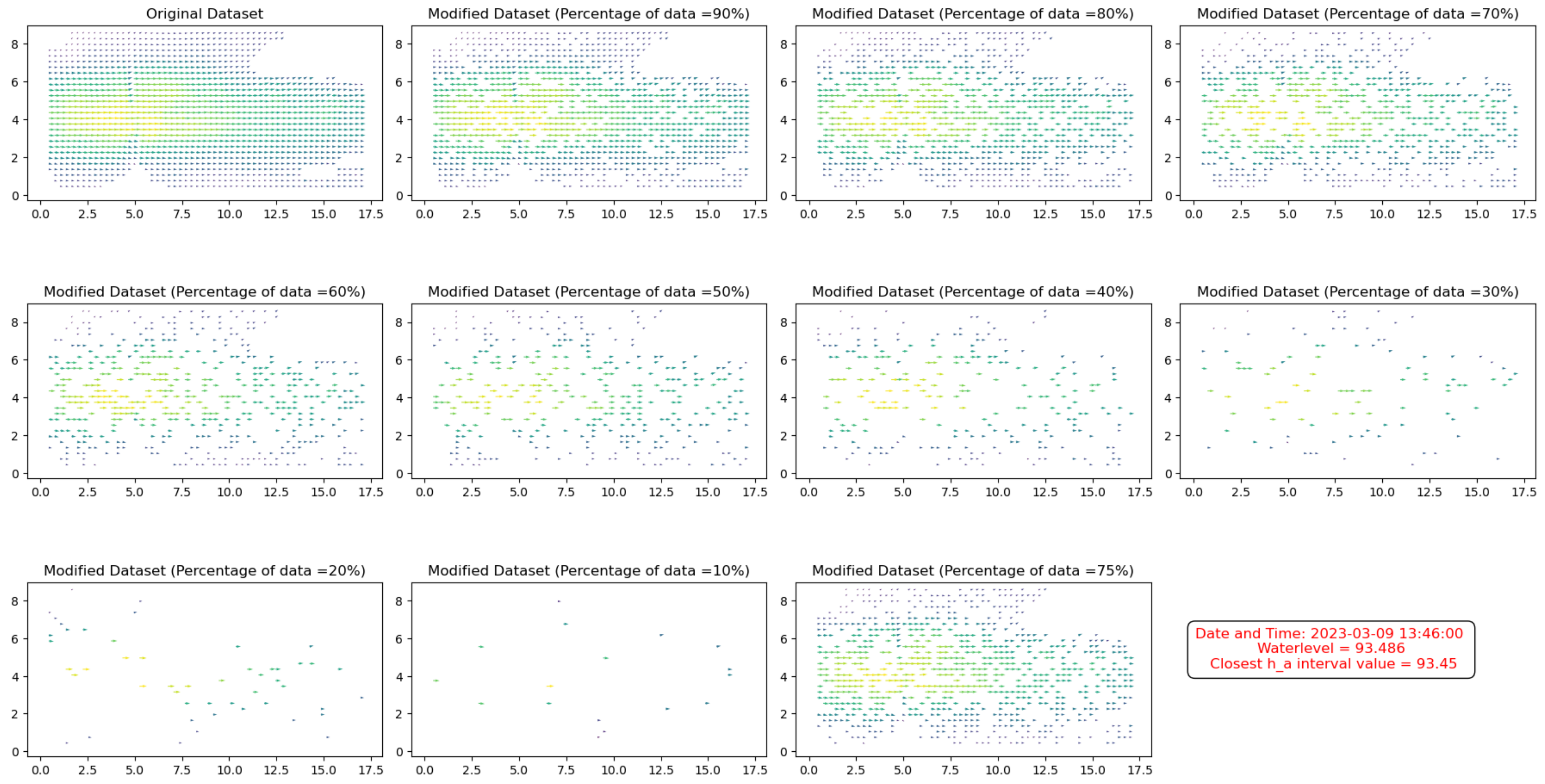


Figure 13 Velocimetry with randomly removed data

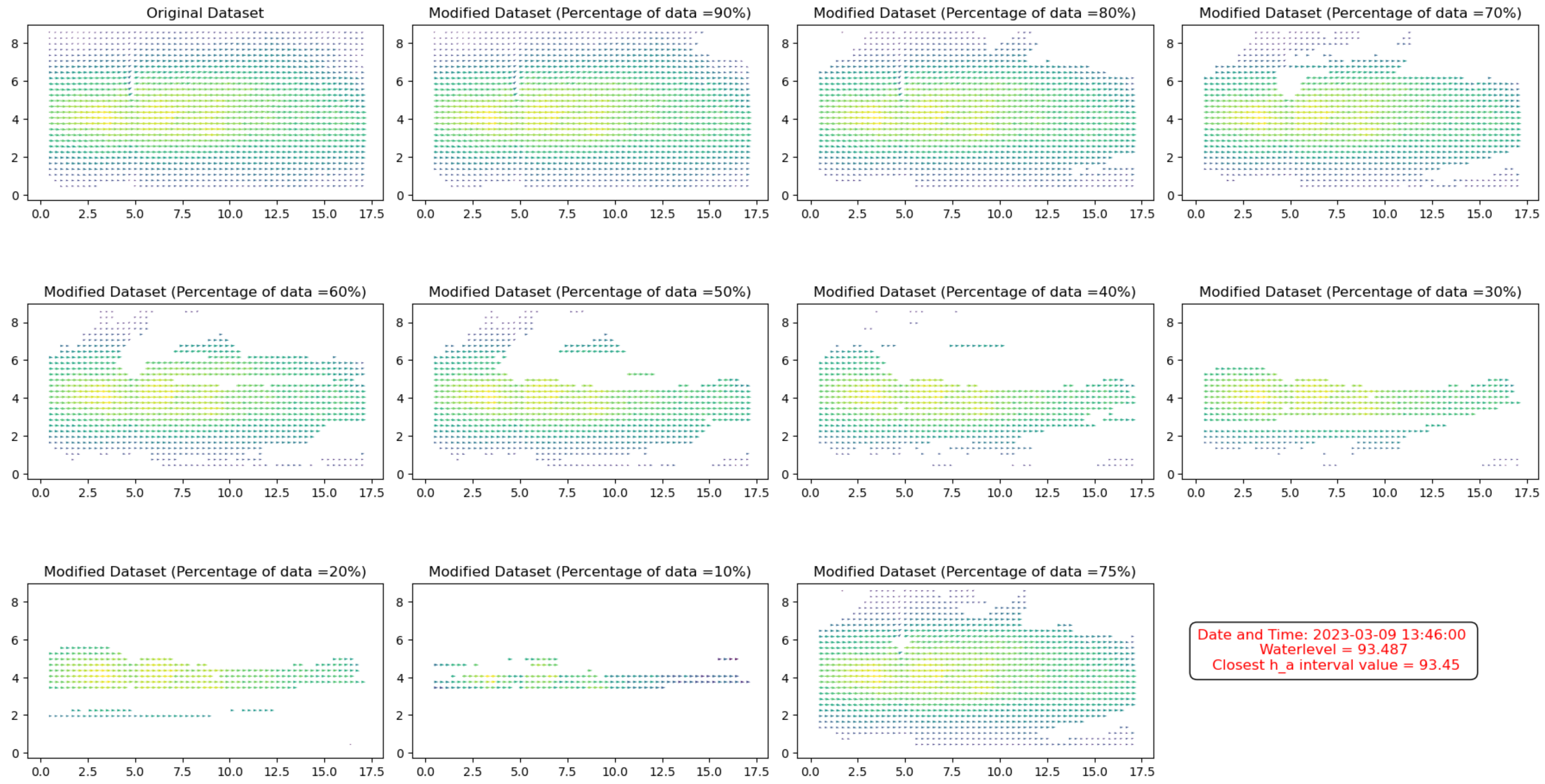


Figure 14 Velocimetry with data removed in order of the missing values

3.6 Benchmark Model Construction

The initial phase of model development involved the construction of a benchmark model. The purpose of this benchmark model was to establish a baseline for performance comparison with the subsequently developed AI model. The benchmark model was designed to utilize a simple yet effective method here referred to as the mean method. By comparing the results of the benchmark to the generated results by the AI model, we can analyze whether the implementation of the AI improves the model based on the existing limits.

Constructing a benchmark baseline model before developing an AI model is because it allows for a comprehensive comparison of the AI model's performance against an established standard, providing insights into the effectiveness of the AI techniques employed. Furthermore, the benchmark model helps identify the limitations of existing approaches, guiding the development of more advanced AI models.

The benchmark model employed the mean method for reconstructing missing values in the dataset. Specifically, for each grid point within a sample, missing values were replaced with the mean values calculated from the other exact grid points in the datasets with the same grouped water level. This approach leveraged the assumption that similar water levels exhibit the same flow patterns, allowing the mean method to provide a reasonable estimation of missing velocity values.

For each mask generated in the preceding section (3.5 Masking the data), the remaining data points serve as input for the benchmark model. The target for this model comprises the original dataset, that is expected to be the ground truth. To fill in the missing data points in the target, which is absent in the input, mean values from all other datasets with the same water level are utilized. Subsequently, the differences between these reconstructed points and the original ground truth points are calculated. It is important to note that for a fair evaluation of the model, only the grid points that are reconstructed are taken into account for the calculation of the differences. Figure 15 illustrates the sequential steps involved in constructing the benchmark model using a random mask containing 10% input data, whereas Figure 16 illustrates the same steps but with a patchy mask with 10% of input data.

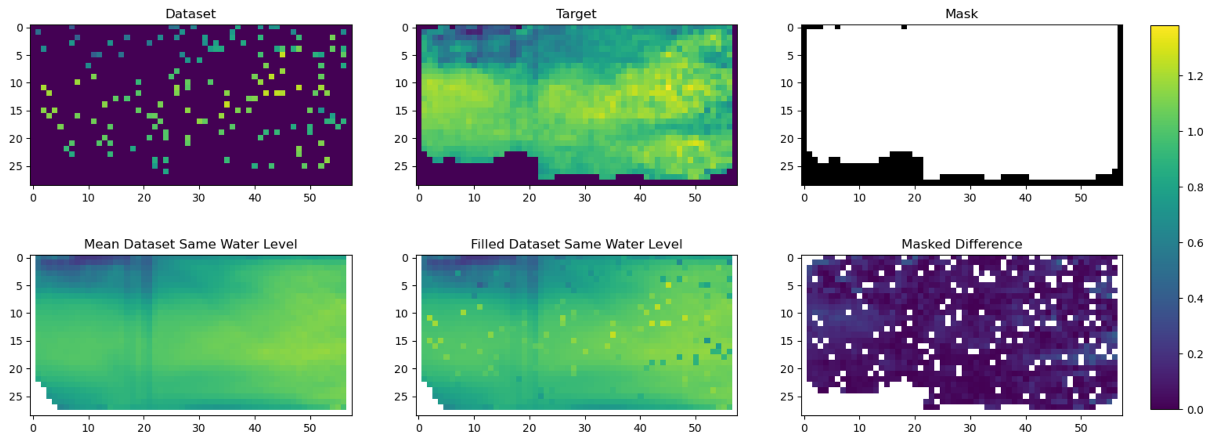


Figure 15 Visualization depicting the sequential steps involved in constructing the benchmark model with a random mask of 10% data

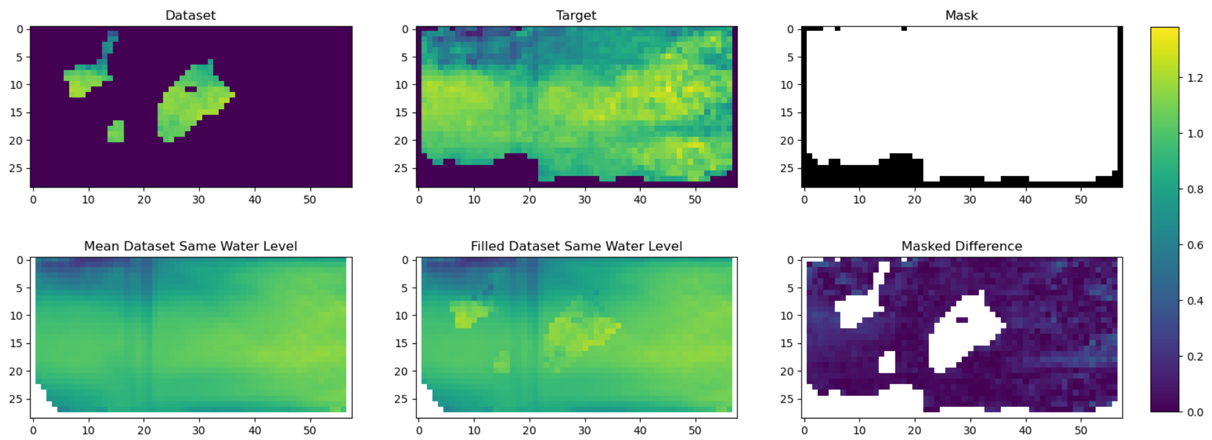


Figure 16 Visualization depicting the sequential steps involved in constructing the benchmark model with a patchy mask of 10% data

3.7 U-Net Model Construction

Following the establishment of the benchmark model, the next phase involved the development of an advanced AI model utilizing the U-Net architecture described in section 2.2.2 U-Net Architecture.

The U-Net model comprised an encoder-decoder structure, enabling the extraction and reconstruction of features with a focus on preserving spatial relationships. The encoder section downscaled the input, extracting high-level features, while the decoder section performed upsampling to recreate spatial details. Appendix B: Code for the U-Net Architecture, shows the code used for the U-Net Architecture.

The input to the U-Net model consisted of the same modified datasets used in the benchmark model. The targets for the U-Net model were set as the original datasets, creating a supervised learning framework for the model to learn from ground truth information. Each input sample to the U-Net model comprised dual channels representing velocity in the x and y directions. This dual-channel configuration enabled the U-Net architecture to consider both components of the velocity field, enhancing its capacity to capture spatial dependencies effectively. The exact architecture for the velocity in the x direction of the model is visualized in Figure 17. The training objective was to minimize the difference between the predicted velocity values and the ground truth values. This was done by comparing the output of the model to the ground truth velocimetry as shown in Figure 18. The term "loss" refers to a measure of how well a model is performing on a given task. It represents the discrepancy between the predicted outputs of the model and the actual target values. The goal during training is to minimize this loss. To guide the training process, the Mean Squared Error Loss was selected as the loss function. This function computed the mean squared difference between predicted and ground truth velocity values for each grid point.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

The Adam optimizer with a learning rate of 0.001 was utilized for efficient parameter updates during training.

It is crucial to divide the dataset of the model into training, validation and testing. In this case, a division of 80% training, 10% validation and 10% testing is used. The importance of this is to make sure that the model is not getting overfitted with data it already has seen. During the training, weights and biases in the kernels are updated to get the predicted output as close to the target as possible. The validation tracks the performance of the training without updating these weights and biases. The frequency of the validation is after each training cycle, known as epoch. If the loss in the validation is not improving, a decision can be made to stop the training. In machine learning, patience is a hyperparameter used to control the training duration based on the model's performance on a validation set. It is very possible a model is performing worse after a training cycle but will improve afterwards. Patience is commonly used to stop the training early after the performance does not improve after several cycles. The idea behind patience and early stopping is to prevent the model from training for too long and potentially overfitting the training data. It helps to find a balance between training long enough to learn useful patterns and stopping early enough to avoid overfitting.

After the training and validation, the model should be tested. During testing, a dataset not previously seen by the model is used. This way we can see the performance of the model for unseen data. This technique is used to avoid overfitting the model and it provides a final evaluation of the model's generalization ability. So, validation results during training are used to guide the training process, and test results after training are used to get a final assessment of the model's performance.

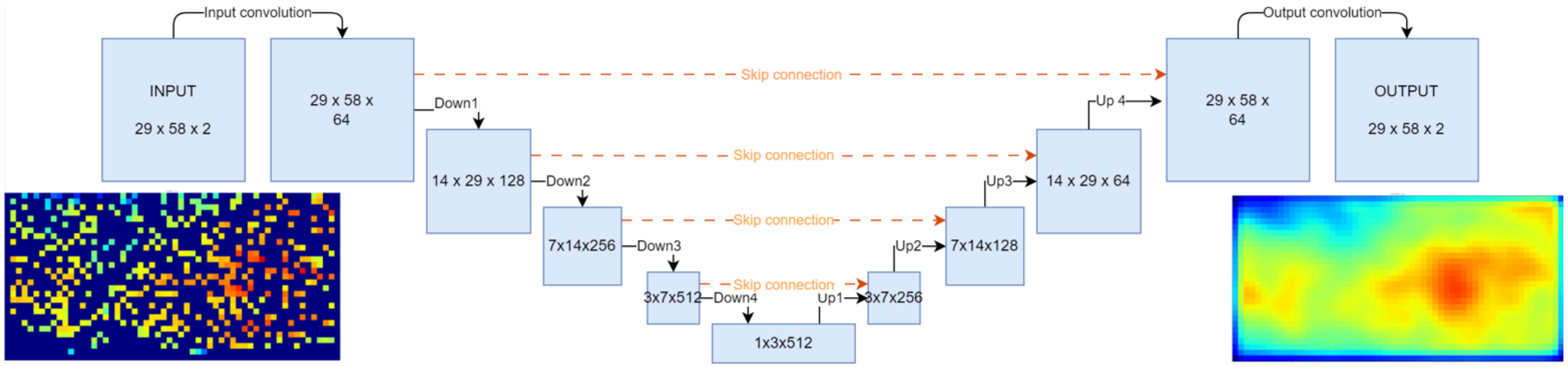


Figure 17 U-Net Architecture visualized with the input and the output image for the velocity in the x direction

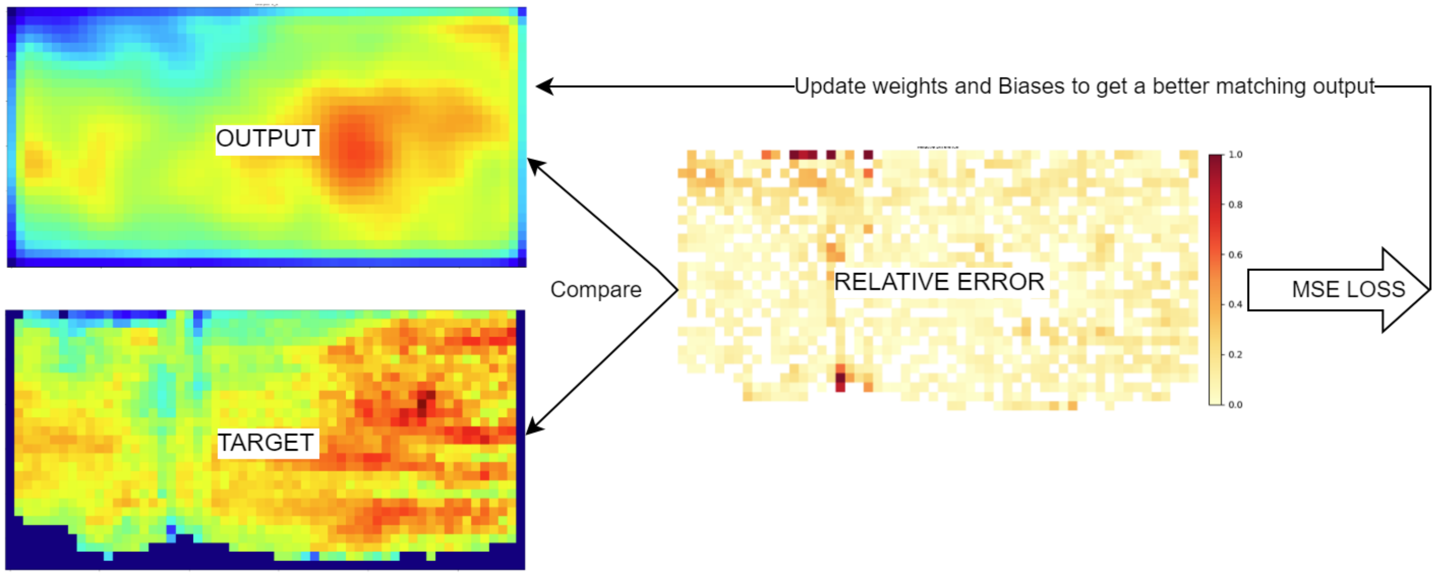


Figure 18 Computing the loss by comparing the output and the target values for the filled-in grid points

3.8 Evaluating the performance

To quantify the performance of the benchmark model and the U-Net model, the root mean squared error (RMSE) and the relative error for each grid point and the entire velocimetry are computed. The relative error is the RMSE divided by the velocity on that grid point. To ensure a fair and unbiased comparison, identical datasets are utilized to test both the benchmark model and the AI model. The error for the benchmark model can be computed in a single go since it is the average of all other datasets. The errors of the U-Net model ideally are lowered with every training epoch.

Appendix C: Training progress Results after 1,6, 11 and 20 epochs, shows the results of the training progress after multiple epochs. Figure 19 shows how the output of the model changes after training. When evaluating the performance of the U-net model, only the results of the fully trained model are considered.

$$RMSE = \sqrt{\frac{1}{N}\sum\nolimits_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

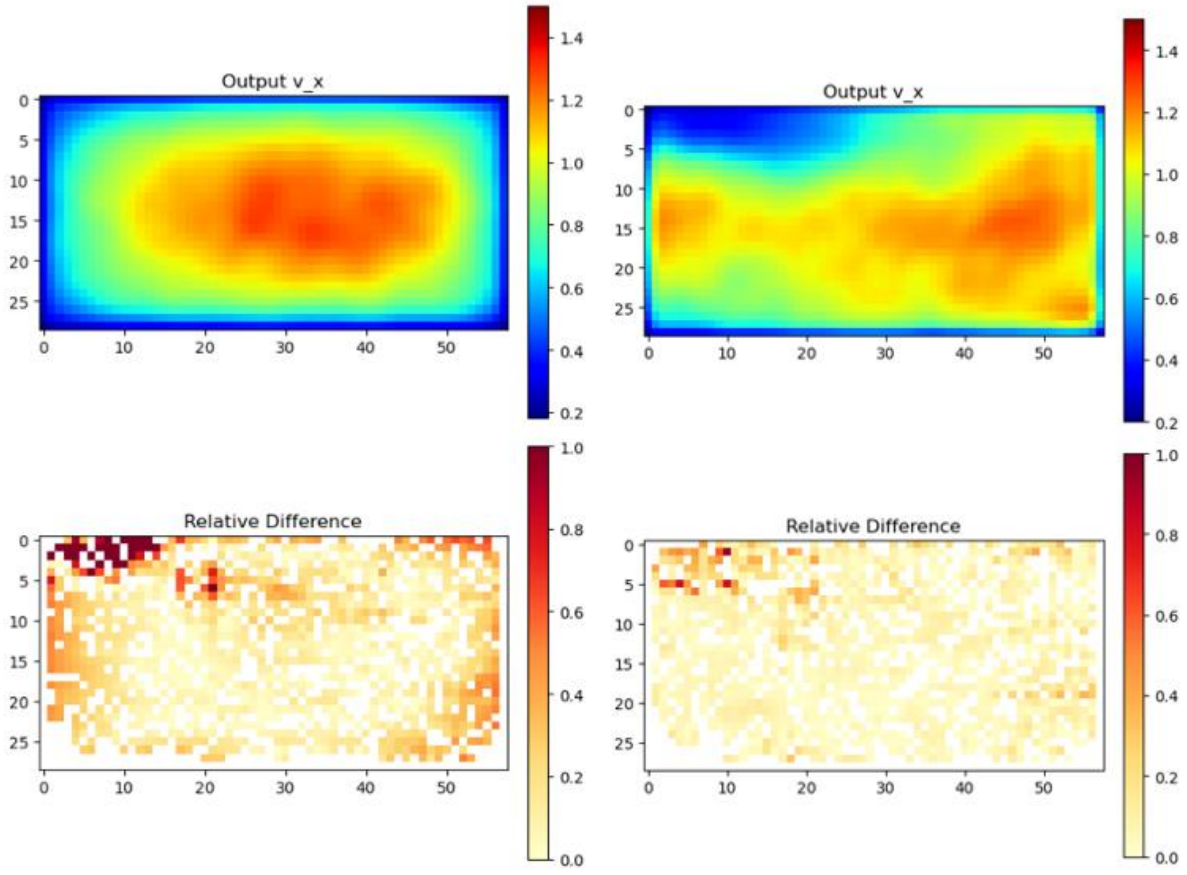


Figure 19 Model output after 2 epochs on the left vs 16 epochs on the right

For further analysis, only the fully trained models are used. To see the influence of the discharge. The input data (Figure 20), target data(Figure 21) and output data(Figure 22) are inserted in the datasets to retrieve the transect and subsequently the data.

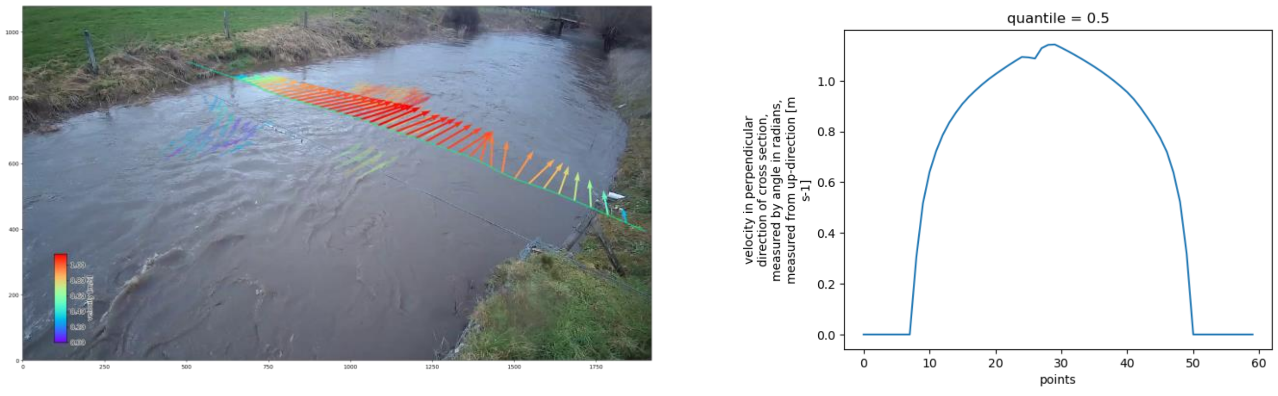


Figure 20 Transect and discharge of the input data

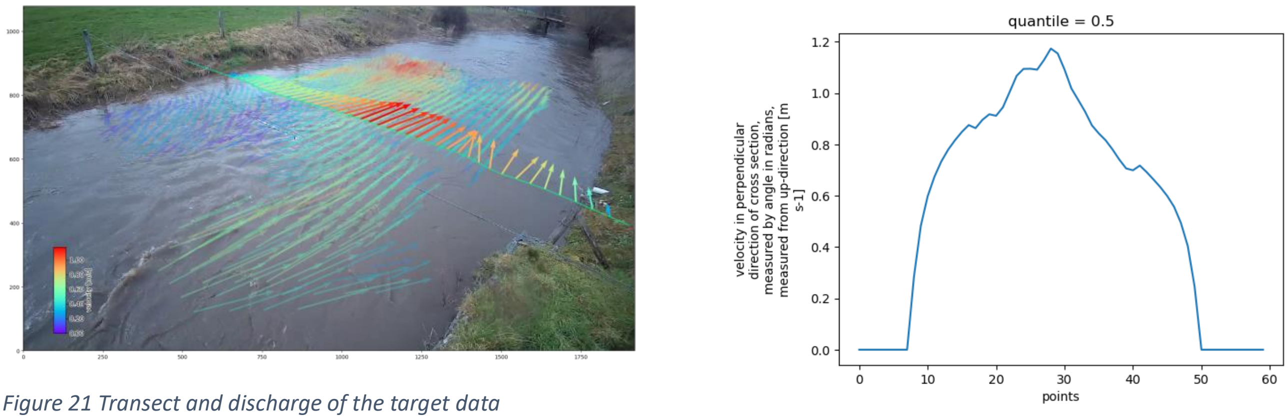


Figure 21 Transect and discharge of the target data

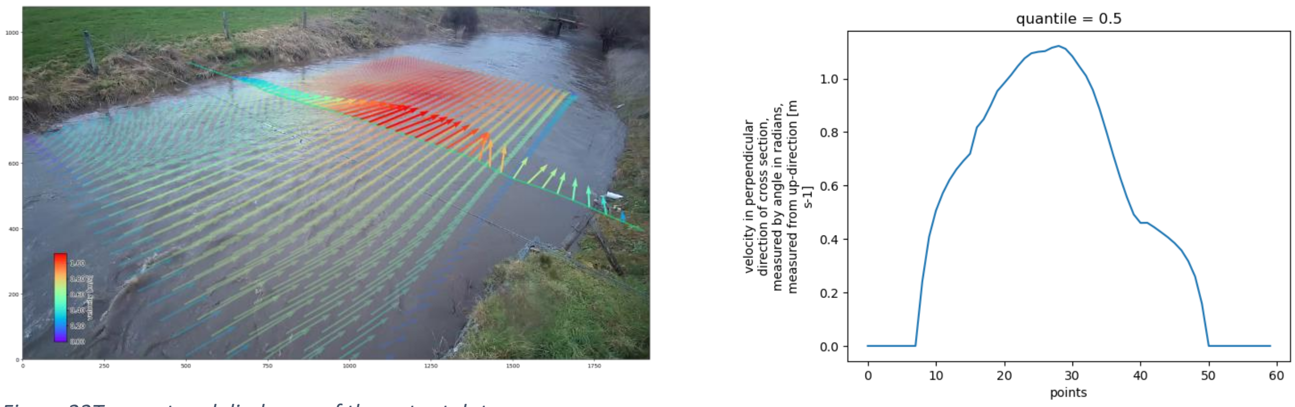


Figure 22Transect and discharge of the output data

4. Results

This chapter presents the findings obtained from the evaluation of the benchmark model and the U-Net model across different masking scenarios, specifically focusing on 92.35 meter NAP with an optical resolution of 80%. The rationale behind this choice lies in the abundance of available samples within this subset, providing a robust foundation for initial comparisons. By narrowing the focus to a specific water level and maintaining an optimal optical resolution, the analysis aims to provide an understanding of the model's performance under controlled conditions. This strategic selection allows for an insightful exploration of the U-Net model's capabilities, particularly in comparison to the benchmark model.

4.1 Performance of the benchmark model

The benchmark model served as a fundamental reference point, constructed using a mean filling method. An analysis of the benchmark model's performance serves as a baseline for evaluating the effectiveness of the subsequent U-Net model.

To assess the performance of the benchmark model, various percentages of input data are examined. The evaluation includes calculating the average RMSE across all considered grid points, along with determining the relative error in comparison to the ground truth. The results are given in Table 1.

Table 1 Results benchmark model

% input data	Grid points considered	Average RMSE [m/s]	Relative Error [-]
10	43399	0.1782	0.1998
20	38423	0.1751	0.1997
30	33476	0.1786	0.1998
40	28902	0.1783	0.1996
50	24039	0.1785	0.2000
60	19184	0.1777	0.1997
70	14447	0.1777	0.2002
80	9552	0.1788	0.1994
90	4750	0.1784	0.2017

Immediate observations reveal minimal variance in error rates across different percentages of input data. The model does not seem to derive substantial benefits from increased input, suggesting that the error is primarily influenced by how well the target datasets approximate the overall mean datasets.

Spatially in Appendix D: Relative Errors from the Benchmark Model, the relative RMSE for all different masks is visualized. An example of the spatial distribution is given in Figure 23. Here the masks for 50% of the input data are used. The model demonstrates reasonable accuracy, as indicated by lower average relative RMSE values. However, in the top-left edge of the river, higher average relative RMSE values are observed, suggesting poorer model performance in that specific region.

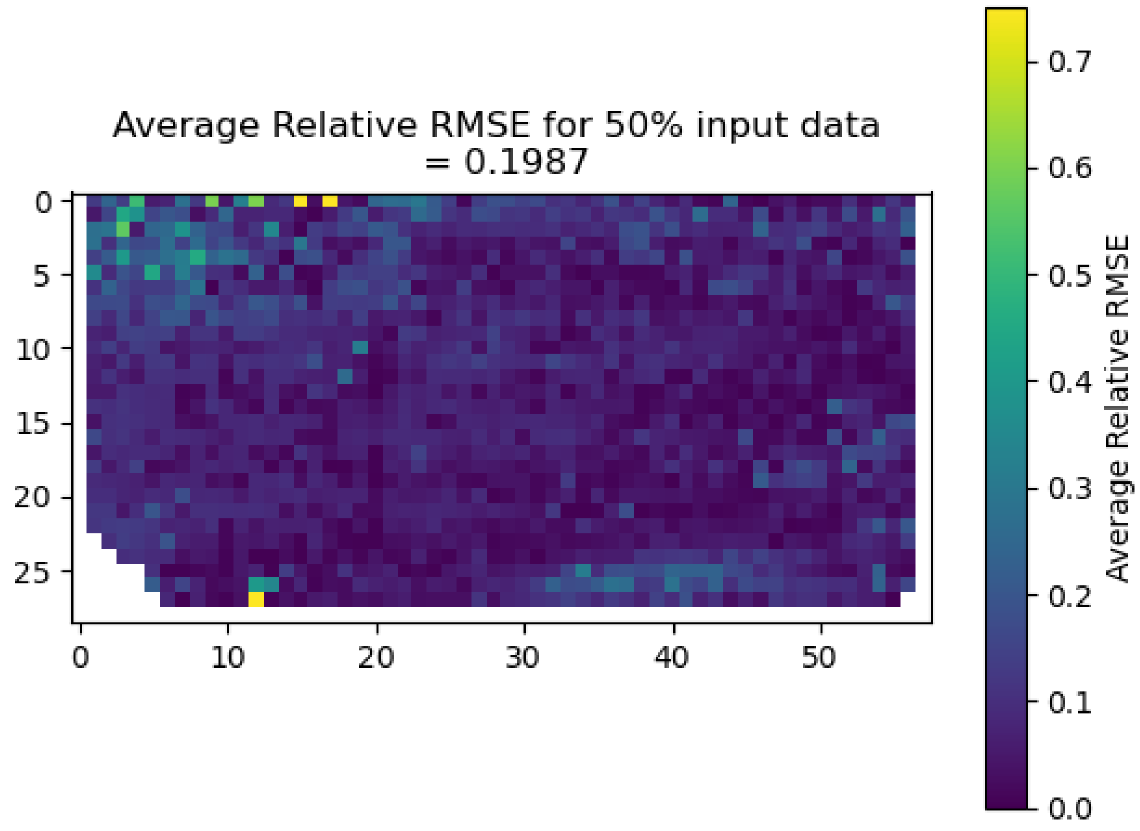


Figure 23 Visualization of the benchmark model with the 50% input data samples

4.2 Performance of U-Net with Random Masks

To assess the U-Net model's performance with random masks, similar to the benchmark model, varying percentages of input data are considered. In Appendix E: Complete set of relative errors of U-Net predicted data with 10% of data input, the relative errors for all samples with input data of 10% are displayed. The evaluation includes calculating the average RMSE across all considered grid points, along with determining the relative error in comparison to the ground truth. The results are given in Table 2.

Table 2 Results of the U-Net model with random masks

% input data	Grid points considered	Average RMSE [m/s]	Relative Error [-]
10	43399	0.0680	0.0850
20	38423	0.0601	0.0740
30	33476	0.0459	0.0583
40	28902	0.0332	0.0409
50	24039	0.0279	0.0344
60	19184	0.0224	0.0276
70	14447	0.0151	0.0183
80	9552	0.0090	0.0108
90	4750	0.0047	0.0057

The results show that the model can predict more accurately the more data is available. Spatially, as visualized in Figure 24 for 50% input data, the model shows the same difficulties as the benchmark model in predicting the upper edge of the river. It should be noted that the U-Net model outperformed the benchmark model in these areas. Appendix F: Relative Root Mean Squared Errors of the U-Net model with different percentages of random masks, shows the average relative RMSE for all the input data.

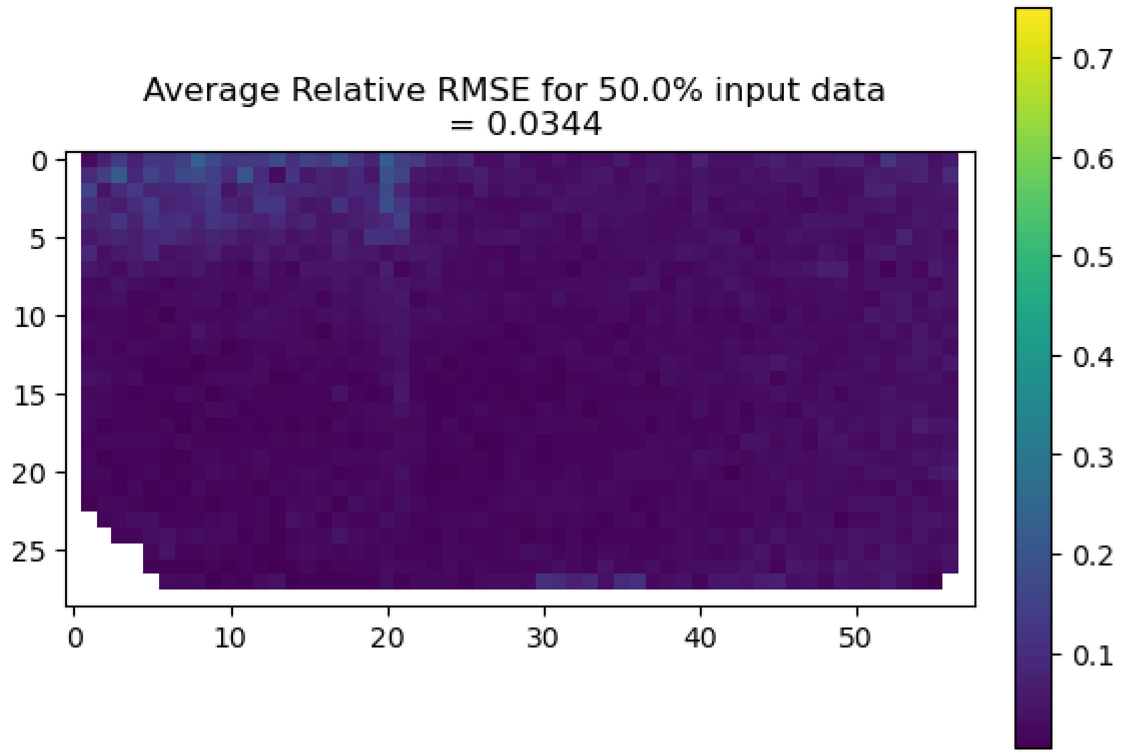


Figure 24 Visualization of the U-Net model with the random 50% input data samples

4.3 Performance with Patchy Masks

Training and evaluation of the model trained with the patchy input mask was done in the same way as the random input mask, calculating the average RMSE across all considered grid points, along with determining the relative error in comparison to the ground truth. These results are shown in Table 3.

Table 3 Results from the U-Net model with patchy masks

% input data	Grid points considered	Average RMSE[m/s]	Relative Error [-]
10	42482	0.0767	0.0955
20	36989	0.0773	0.0979
30	31553	0.0779	0.0980
40	26114	0.0709	0.0875
50	20783	0.0731	0.0898
60	15417	0.0707	0.0858
70	10157	0.0658	0.0808
80	4982	0.0540	0.0663
90	486	0.0226	0.0310

What we can observe is that similarly to the random mask model, the U-Net model trained on the patchy masks improves when more data is available. This would suggest a positive correlation between data volume and model performance. However, it's important to note that at very high percentages of input data, the model was not required to predict the upper-left corner, which spatially proved to be the most challenging area. Figure 25 provides a spatial visualization of the model's performance with the 50% input data. Appendix G: Result of Relative RMSE averaging over all 33 datasets for patchy masks, shows the average relative RMSE for all input data.

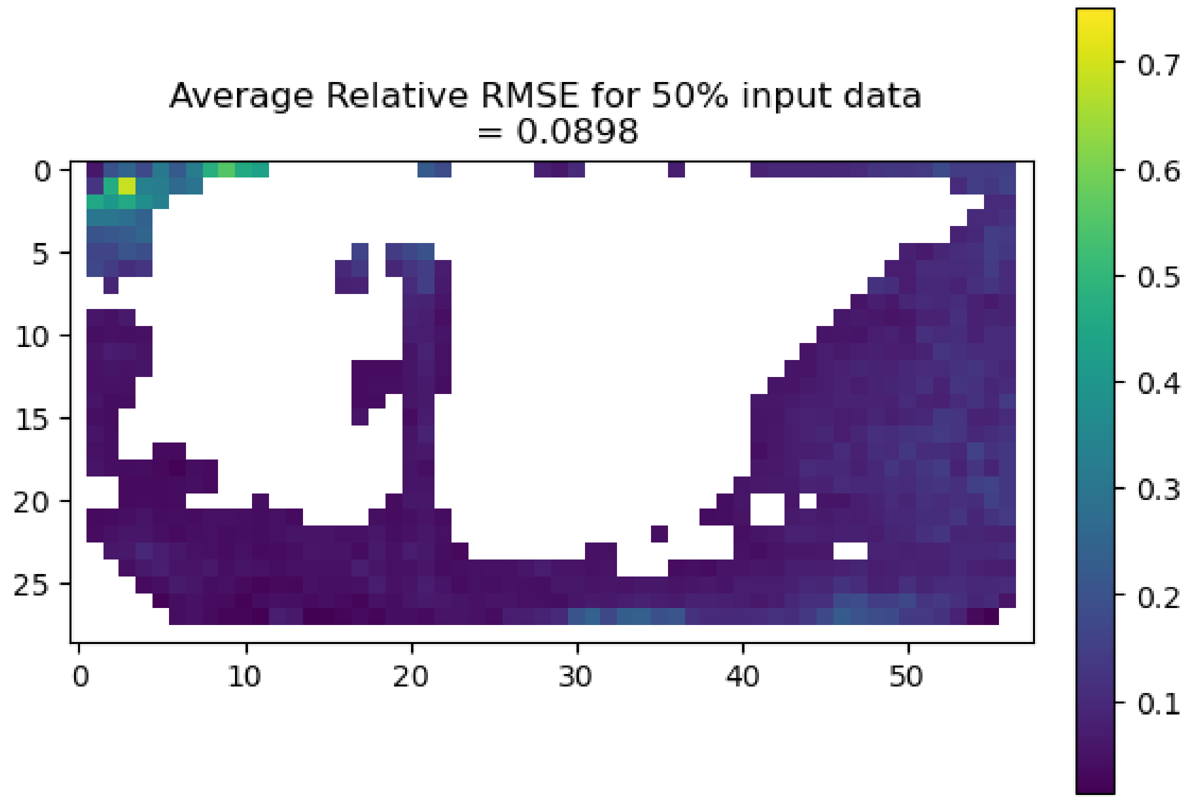


Figure 25 Visualization of the U-Net model with the patchy 50% input data samples

4.4 Comparison between the models

In this section, we compare the performance of the benchmark model with that of the U-Net model trained on random and patchy masks. The evaluation is based on the relative error which quantifies the difference between predicted and actual velocities for each input percentage of the data. By assessing how closely the predicted velocities match the ground truth values, we can determine the effectiveness of each model in reconstructing missing data. Figure 26 illustrates the performance of the benchmark model and the U-Net models trained on random and patchy masks with varying percentages of input data. In the case of the benchmark model, the line remains horizontal, indicating that its performance does not significantly improve with an increase in input data percentage. Conversely, for the U-Net models, there is a noticeable trend of improved performance with higher percentages of input data. This improvement is reflected in the decreasing trend of relative error as more input data is utilized. Overall, the U-Net models outperform the benchmark model, demonstrating their efficacy in reconstructing velocity fields. Additionally, the diminishing relative error suggests that the models achieve better accuracy as they are provided with more input data.

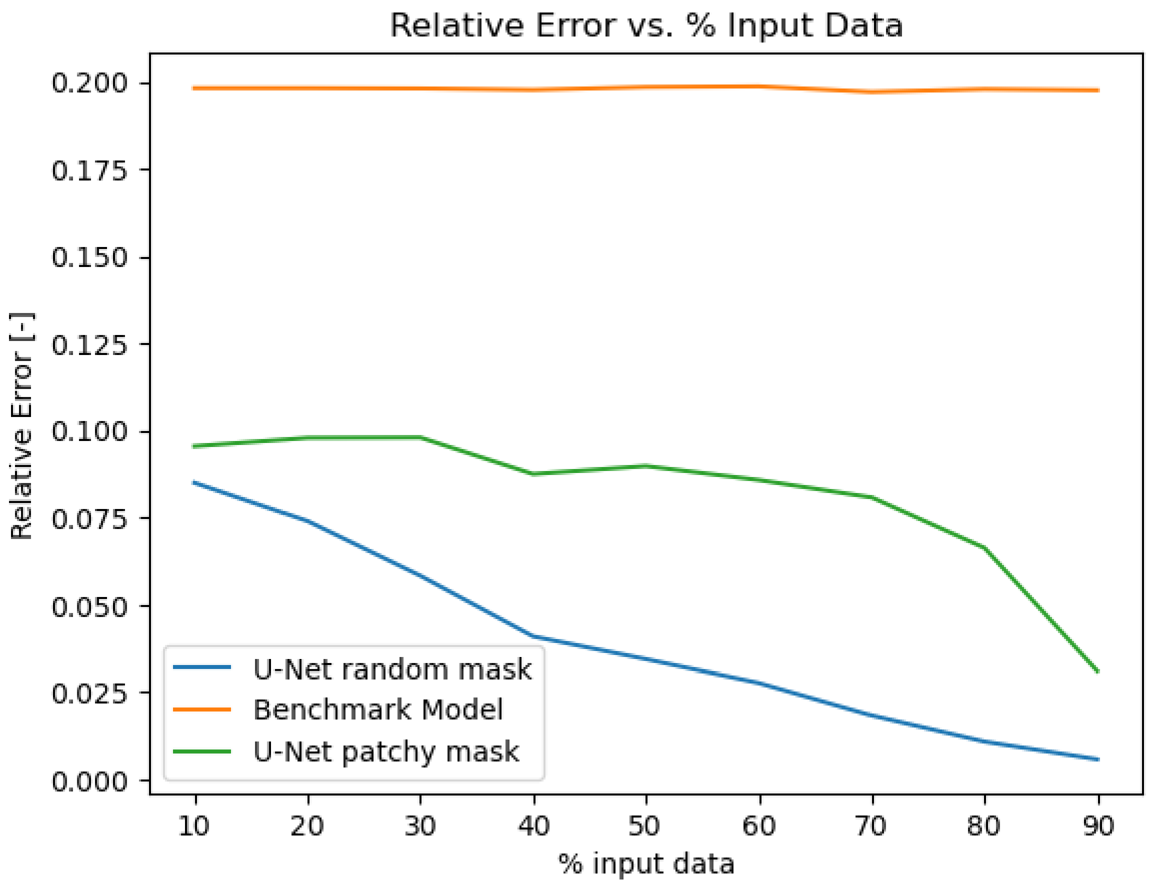


Figure 26 Comparison of the Benchmark model vs the U-Net model.

4.5 Discharge comparison

In this section, the river flow for the patchy mask 10% was calculated, as described in section 3.8 Evaluating the performance based on the input data, target data, and output data. Subsequently, the river flow values were plotted in a graph for comparison. The graph presented in Figure 27 Visualization of the calculated river flow values for 10% input data illustrates the comparison of river flow values obtained from the input, target, and output data for the patchy mask 10%. The analysis of the 10% mask was conducted to evaluate the model's performance in predicting river flow under conditions where a significant portion of the data was missing. By specifically examining the 10% mask, we aimed to assess the model's ability to reconstruct river flow values with limited input data, simulating scenarios where data availability is scarce

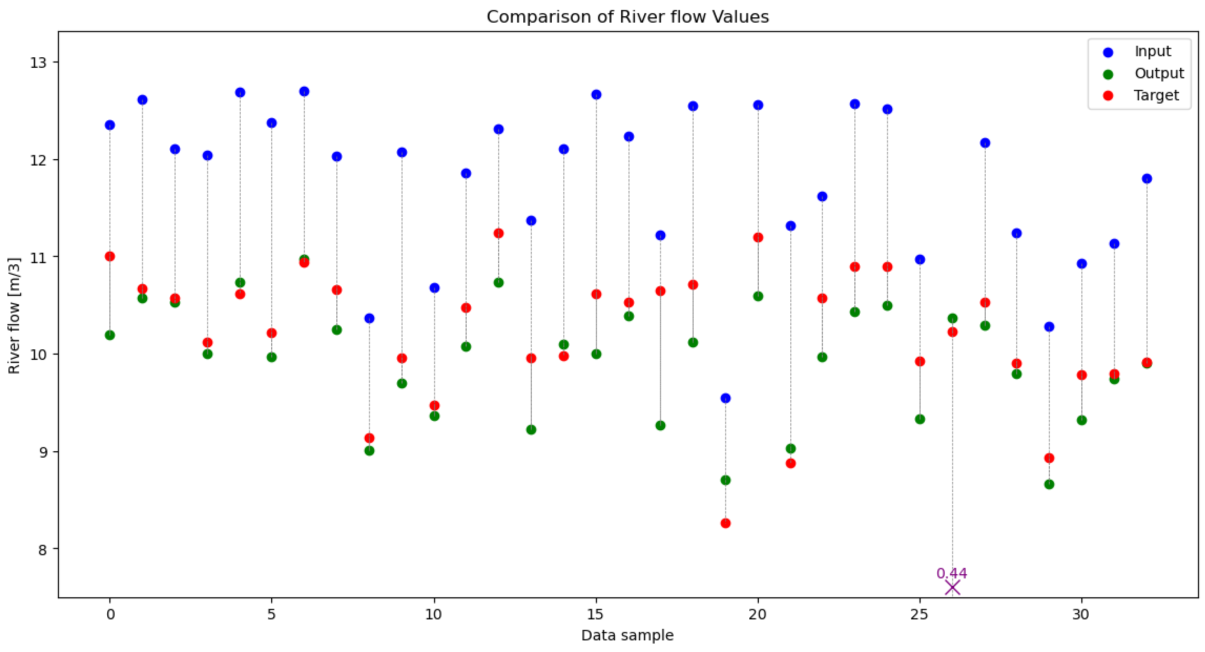


Figure 27 Visualization of the calculated river flow values for 10% input data

Upon analyzing the river flow values derived from the input, target, and output data, it was observed that the input river flow values were notably higher than the target values. This discrepancy indicated that the available velocities were potentially overvalued, leading to inflated river flow calculations. In contrast, the output river flow values were closer to the target values, suggesting an improvement in river flow calculations achieved by the model.

It is noteworthy that a single outlier (sample 26) was identified within the dataset. Further investigation into this outlier could provide insights into the model's predictive capabilities. As depicted in Figure 28, the transect and river flow for this specific sample demonstrate challenges in predicting river flow when insufficient values are available in the dataset grid points. Figures 29 and 30 illustrate the target (considered the ground truth) and the output of the model, respectively, showcasing a closer resemblance between the model's output and the target. This particular case highlights the model's ability to improve predictions in challenging scenarios.

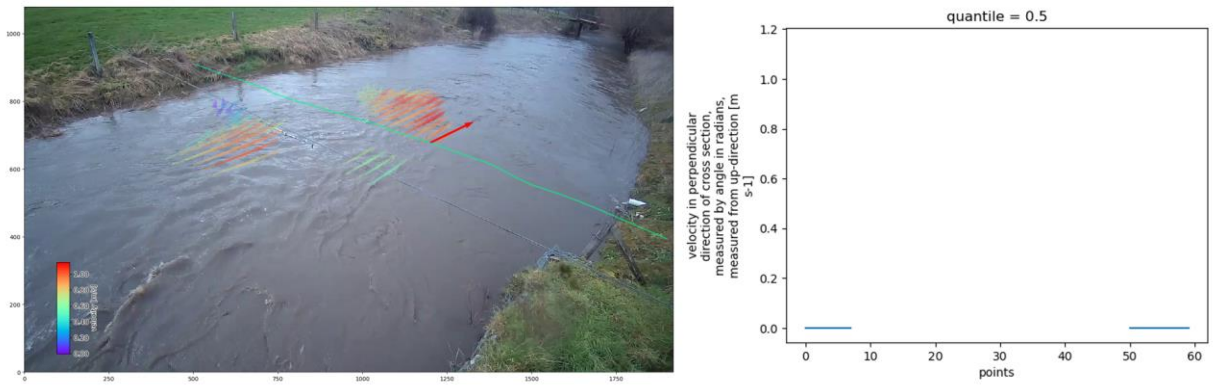


Figure 28 Transect and river flow of the input data from the outlier sample

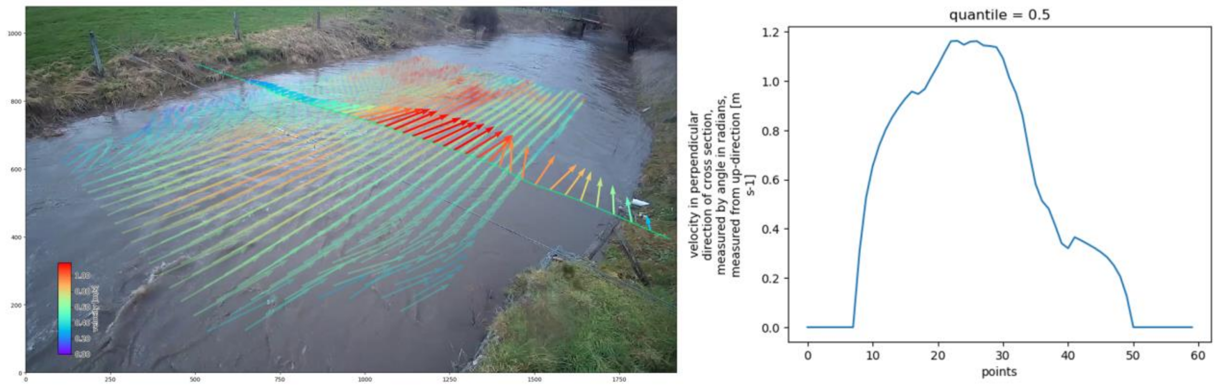


Figure 29 Transect and river flow of the target data from the outlier sample

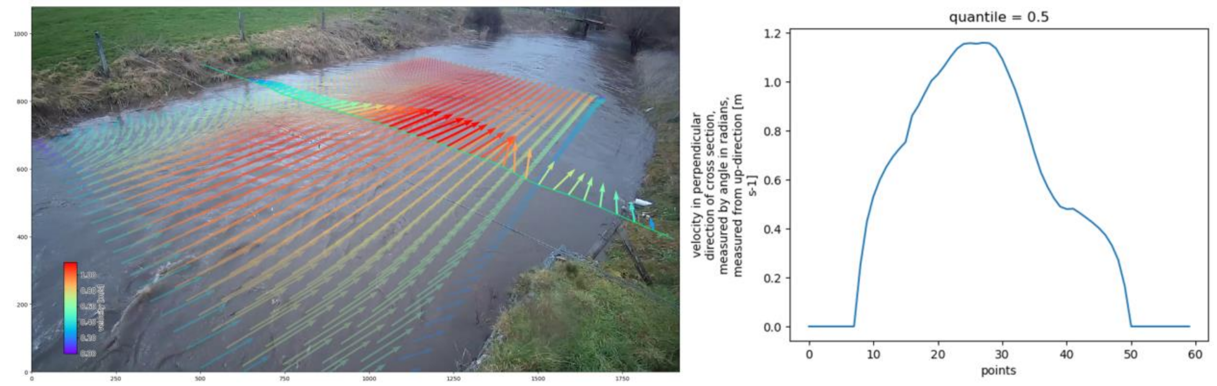


Figure 30 Transect of the output data from the outlier sample

5. Discussion

From the findings gathered in the literature review, the decision to implement a U-Net model for dataset reconstruction appears to be well-founded. The utilization of the U-Net architecture resulted in the generation of usable datasets even from inputs with limited information, closely resembling the target datasets.

The benchmark model, while demonstrating satisfactory performance under certain conditions, exhibited limitations that hindered its effectiveness. Its performance was notably influenced by the similarity between the target dataset and the average data. Consequently, the model tended to perform poorly when presented with outlier datasets that deviated significantly from the average, whereas it performed relatively well with datasets closely resembling the average. This disparity in performance can be attributed to the inherent limitations of the model's simplistic approach, which lacked the flexibility to adapt to diverse datasets and handle outliers effectively.

In contrast, the U-Net model demonstrated significant performance improvements, largely due to its ability to assign weights and biases to available grid points and its training under similar conditions. The U-Net architecture's design, characterized by an encoder-decoder structure, facilitates the extraction and reconstruction of features while preserving spatial relationships. During training, the model learns to adjust the weights and biases associated with each grid point through optimization algorithms like the Adam optimizer. This process enables the model to effectively capture complex patterns and variations in the input data, thereby enhancing its predictive capabilities.

5.1 Limitations

The decision to employ the U-Net architecture proved justified as it facilitated the generation of usable datasets even from inputs with limited information, closely resembling the target datasets. However, it's essential to acknowledge that while some preprocessing steps were applied to the data, uncertainties regarding its accuracy remained, as it was assumed to represent ground truth. The model is trained to match the target but has no capability of determining if the target is accurate itself.

Furthermore, the effectiveness of the benchmark model was notably influenced by the degree of similarity between the target dataset and the average data. However, it's crucial to acknowledge that this reliance on similarity could pose significant limitations, especially in scenarios where substantial differences exist between the two datasets. In such cases, where the target dataset deviates significantly from the average data, the performance of the benchmark model may be compromised, leading to less reliable outcomes.

Despite its notable performance gains, the U-Net model encountered challenges similar to the benchmark model in certain regions, particularly at the edges of the dataset. These challenges may stem from visibility issues in recording velocities, leading to measurement errors, or from the model's limited consideration of specific features such as embankments. Addressing these challenges could further enhance the model's performance in capturing fine-grained details and improving accuracy in regions prone to errors.

Another notable limitation stems from the training of the model on a subset of the datasets resulting in the model being predominantly trained on homogeneous flow conditions. This lack of exposure to extreme conditions during the training phase could potentially restrict the model's ability to accurately predict outcomes in scenarios with significantly different flow dynamics.

In essence, the model's performance may be constrained by its limited exposure to diverse and extreme scenarios, leading to potential challenges in generalizing its predictions to real-world conditions characterized by greater variability and complexity.

5.2 Recommendations for Future Research

Moving forward, it is recommended to expand the diversity of the dataset utilized in the study. While the model has shown promising results, incorporating a broader range of datasets featuring different water levels could offer a more comprehensive assessment of its performance across various hydrological conditions. If the model is also trained on different areas with distinct flow patterns, it would enhance its adaptability and robustness in predicting velocities across diverse river systems. This expansion would provide valuable insights into how the model adapts to different scenarios and its generalizability to real-world applications.

Moreover, there is a need to refine techniques both before and after model implementation to optimize performance. Further exploration and refinement of preprocessing methods for data input and post-processing techniques for improving model output could lead to enhanced accuracy and reliability in predicting river discharge.

Another avenue for future research involves incorporating principles from the laws of physics into the model training process. By integrating physics-based constraints, such as the conservation of mass and momentum, researchers can enhance the model's ability to simulate realistic flow patterns and dynamics in river systems. This approach not only improves the model's predictive accuracy but also provides valuable insights into the underlying physical processes governing river behaviour. By refining the model with physics-based constraints, researchers can achieve a more comprehensive understanding of fluid dynamics and enhance the model's applicability to diverse hydrological scenarios. Future research could focus on simulating and testing the model under extreme hydrological patterns and scenarios. By assessing its performance in such conditions, researchers can evaluate the model's robustness and identify areas for further improvement, ensuring its reliability in real-world applications.

Lastly, future research could be focused on the development of a more comprehensive framework for predicting river flow velocity instead of solely focusing on reconstruction. This entails integrating fundamental physical principles into the model training process to supplement existing data-driven approaches. The success of Physics-Informed Neural Networks in related fields [60][61], such as computational fluid dynamics [68] and blood flow analysis[69], which incorporate Partial Differential Equations into training losses, offer a promising avenue for enhancing model inference by considering initial and boundary conditions. However, a critical aspect to address will be selecting the most relevant physical laws, such as the Navier-Stokes equations or standard conservation laws, to guide the model development effectively. Embracing this physics-informed approach could lead to significant advancements in the understanding of river dynamics and improve the predictive capabilities of our models.

6. Conclusion

In this concluding chapter an analysis of the findings and insights gathered throughout this thesis are provided addressing the main research question and its associated sub-questions.

Research question: How effectively can AI be employed to augment missing velocity data on the river surface?

Sub-question 1: What kind of AI model has the best potential for reconstruction of a dataset?
Based on the literature review, it is evident that convolutional neural networks (CNNs) and specifically U-net models emerge as a potent tool for reconstructing datasets with missing images. The inherent capabilities of CNNs in capturing complex patterns make them well-suited for this task.

Sub-question 2: Do AI models outperform a typical hydrology benchmark model?
The results demonstrate that AI and in particular the U-Net model indeed outperforms the typical hydrological benchmark model, such as the mean/mode model. The extent of improvement exhibited by the U-Net model depends on factors such as the availability of data and the duration of training.

Sub-question 3: To what extent does the performance of the AI model vary with differing quantities of missing data on the river surface?
The performance of the AI model shows variation with differing quantities of missing data on the river surface. As more data becomes available, the model's performance in filling in the missing parts improves, highlighting the importance of data availability in enhancing model accuracy.

Sub-question 4: What is the influence on the performance of the model training on patchy data compared to random data?
The influence of model training on patchy data versus random data is notable. While the model trained with random data may exhibit a lower loss score, it faces a relatively easier task compared to the model trained on patchy data. The latter encounters challenges in predicting grid points that are inherently more difficult to estimate.

Sub-question 5: Which areas are hardest to predict in the case study?
The analysis identifies certain areas within the case study that are particularly challenging to predict. These areas, typically located in the corners of the image, often exhibit significant prediction errors. Factors contributing to these errors may include inaccuracies in velocity estimations due to recording conditions or the absence of flow in certain instances.

In summary, this research underscores the significant potential of AI models in hydrological applications, particularly in augmenting missing velocity data on the river surface.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Riccardo Taormina and Hessel Winsemius, whose guidance, expertise, and unwavering support have been invaluable throughout this research journey. Riccardo with immense motivation and knowledge of Artificial intelligence helped introduce this new topic to me. Hessel for the datasets, all the background information on pyOpenRivercam and for helping me understand the initial code.

I also want to thank Xeuqin “Ned” Chen for some extra guidance, particularly on setting up the U-Net, and Nick van der Giesen for joining the committee on short notice to make the graduation possible.

Special thanks are extended to my fellow students who helped me with motivation and stood by me through the ups and downs of the thesis period.

I would like to express my heartfelt appreciation to my mother, Marijke Naber, for her unwavering encouragement, understanding, and patience throughout my entire study career. To my uncle, Bas Naber, helping me with some personal obstacles along the way. To my sister, Laura Helmich, for being a great example.

I wish to acknowledge my late father, Henny Helmich, whose absence served as a reminder of the importance of perseverance and determination. His memory continues to motivate me to strive for excellence, and I am forever grateful for the opportunities he made possible through his sacrifices.

Finally to all my friends for their unwavering encouragement, understanding, and patience throughout this endeavour. Their love and support have been a source of strength and inspiration.

This thesis is dedicated to all those who have played a role, however small, in shaping my academic journey. Thank you.

Declaration of Authorship

I, Max Helmich, hereby declare that this thesis entitled "Improving detection of river surface flow using pyOpenRivercam and Artificial Intelligence augmentation" is entirely my work. It is an original research endeavour, and I have not previously submitted any part of this work for any academic award. All sources used or referred to in this thesis have been properly cited and acknowledged. Any assistance received during the research and writing of this thesis has been appropriately acknowledged in the Acknowledgments section.

Date: 12-04-2023

References

[1] P.J. Depetris, The Importance of Monitoring River Water Discharge, Front. Water. 3 (2021) 1–7. https://doi.org/10.3389/frwa.2021.745912.

[2] J.E. Costa, R.T. Cheng, F.P. Haeni, N. Melcher, K.R. Spicer, E. Hayes, W. Plant, K. Hayes, C. Teague, D. Barrick, Use of radars to monitor stream discharge by noncontact methods, Water Resour. Res. 42 (2006) 1–14. https://doi.org/10.1029/2005WR004430.

[3] F. Tauro, R. Piscopia, S. Grimaldi, Streamflow Observations From Cameras: Large-Scale Particle Image Velocimetry or Particle Tracking Velocimetry?, Water Resour. Res. 53 (2017) 10374–10394. https://doi.org/10.1002/2017WR020848.

[4] J. Westerweel, G.E. Elsinga, R.J. Adrian, Particle image velocimetry for complex and turbulent flows, Annu. Rev. Fluid Mech. 45 (2013) 409–436. https://doi.org/10.1146/annurev-fluid-120710-101204.

[5] P. Buchhave, Particle image velocimetry-status and trends, Exp. Therm. Fluid Sci. 5 (1992) 586–604. https://doi.org/10.1016/0894-1777(92)90016-X.

[6] R. Sánchez-González, S.W. North, Nitric Oxide Laser-Induced Fluorescence Imaging Methods and Their Application to Study High-Speed Flows, 2017. https://doi.org/10.1016/B978-0-12-811220-5.00019-8.

[7] C.J. Kähler, S. Scharnowski, C. Cierpka, On the resolution limit of digital particle image velocimetry, Exp. Fluids. 52 (2012) 1629–1639. https://doi.org/10.1007/s00348-012-1280-x.

[8] W. Li, Q. Liao, Q. Ran, Stereo-imaging LSPIV (SI-LSPIV) for 3D water surface reconstruction and discharge measurement in mountain river fl ows, J. Hydrol. 578 (2019) 124099. https://doi.org/10.1016/j.jhydrol.2019.124099.

[9] M.T. Perks, KLT-IV v1.0: Image velocimetry software for use with fixed and mobile platforms, Geosci. Model Dev. 13 (2020) 6111–6130. https://doi.org/10.5194/gmd-13-6111-2020.

[10] V. Techniques, H. Winsemius, F. Annor, R. Hagenaars, N. Van De Giesen, Towards Open Access and Open Towards Open Access and Open Source Software for Image-Based Velocimetry Techniques, (2023). https://doi.org/10.20944/preprints202308.0896.v1.

[11] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, (2015). https://doi.org/10.1038/nature14539.

[12] M. Reichstein, G. Camps-valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, Deep learning and process understanding for data-driven Earth system science, Nature. (n.d.). https://doi.org/10.1038/s41586-019-0912-1.

[13] K. Yu, S. Kim, D. Kim, Correlation analysis of spatio-temporal images for estimating two-dimensional flow velocity field in a rotating flow condition, J. Hydrol. 529 (2015) 1810–1822. https://doi.org/10.1016/j.jhydrol.2015.08.005.

[14] P.T. Tokumaru, P.E. Dimotakis, Image correlation velocimetry, 19 (1995).

[15] S. Pearce, R. Ljubicic, S. Peña-Haro, M. Perks, F. Tauro, A. Pizarro, S.F. Dal Sasso, D. Strelnikova, S. Grimaldi, I. Maddock, G. Paulus, J. Plavšic, D. Prodanovic, S. Manfreda, An evaluation of image velocimetry techniques under low flow conditions and high seeding densities using unmanned aerial systems, Remote Sens. 12 (2020) 1–24. https://doi.org/10.3390/rs12020232.

[16] E. Rozos, P. Dimitriadis, K. Mazi, S. Lykoudis, A. Koussis, On the uncertainty of the image velocimetry method parameters, Hydrology. 7 (2020).

https://doi.org/10.3390/HYDROLOGY7030065.

[17] A. Eltner, H. Sardemann, J. Grundmann, Technical Note : Flow velocity and discharge measurement in rivers using terrestrial and unmanned-aerial-vehicle imagery, (2020) 1429–1445.

[18] E. Rozos, K. Mazi, A.D. Koussis, Probabilistic evaluation and filtering of image velocimetry measurements, Water (Switzerland). 13 (2021). https://doi.org/10.3390/w13162206.

[19] J.D. Creutin, M. Muste, A.A. Bradley, S.C. Kim, A. Kruger, River gauging using PIV techniques : a proof of concept experiment on the Iowa River, 277 (2003) 182–194. https://doi.org/10.1016/S0022-1694(03)00081-7.

[20] R.J. Adrian, Twenty years of particle image velocimetry, (2005) 159–169. https://doi.org/10.1007/s00348-005-0991-7.

[21] M.T. Perks, A.J. Russell, A.R.G. Large, Technical note: Advances in flash flood monitoring using unmanned aerial vehicles (UAVs), Hydrol. Earth Syst. Sci. 20 (2016) 4005–4015. https://doi.org/10.5194/hess-20-4005-2016.

[22] M. Detert, E.D. Johnson, V. Weitbrecht, Proof-of-concept for low-cost and non-contact synoptic airborne river flow measurements, Int. J. Remote Sens. 38 (2017) 2780–2807. https://doi.org/10.1080/01431161.2017.1294782.

[23] M. Muste, I. Fujita, A. Hauet, Large-scale particle image velocimetry for measurements in riverine environments, 44 (2008). https://doi.org/10.1029/2008WR006950.

[24] M. Muste, A. Hauet, I. Fujita, C. Legout, H. Ho, Advances in Water Resources Capabilities of Large-scale Particle Image Velocimetry to characterize shallow free-surface flows, 70 (2014) 160–171. https://doi.org/10.1016/j.advwatres.2014.04.004.

[25] S. Fortunato, D. Sasso, A. Pizarro, P. Vuono, S. Manfreda, Accuracy of Large-Scale Particle Image Velocimetry (LSPIV) techniques applied on low seeding density flows Accuracy of Large-Scale Particle Image Velocimetry (LSPIV) techniques applied on low seeding density flows, (2019).

[26] M.T. Perks, S. Fortunato Dal Sasso, A. Hauet, E. Jamieson, J. Le Coz, S. Pearce, S. Peña-Haro, A. Pizarro, D. Strelnikova, F. Tauro, J. Bomhof, S. Grimaldi, A. Goulet, B. Hortobágyi, M. Jodeau, S. Käfer, R. Ljubičić, I. Maddock, P. Mayr, G. Paulus, L. Pénard, L. Sinclair, S. Manfreda, Towards harmonisation of image velocimetry techniques for river surface velocity observations, Earth Syst. Sci. Data. 12 (2020) 1545–1559. https://doi.org/10.5194/essd-12-1545-2020.

[27] R. Le Boursicaud, L. Pénard, A. Hauet, F. Thollet, J. Le Coz, Gauging extreme floods on YouTube: Application of LSPIV to home movies for the post-event determination of stream discharges, Hydrol. Process. 30 (2016) 90–105. https://doi.org/10.1002/hyp.10532.

[28] F. Scarano, Theory of non-isotropic spatial resolution in PIV, Exp. Fluids. 35 (2003) 268–277. https://doi.org/10.1007/s00348-003-0655-4.

[29] A. Pizarro, S.F. Dal Sasso, M.T. Perks, S. Manfreda, Identifying the optimal spatial distribution of tracers for optical sensing of stream surface flow, Hydrol. Earth Syst. Sci. 24 (2020) 5173–5185. https://doi.org/10.5194/hess-24-5173-2020.

[30] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2019) 525–545. https://doi.org/10.1007/s00466-019-01740-0.

[31] F.J. Gonzalez, M. Balajewicz, Deep convolutional recurrent autoencoders for learning low-

dimensional feature dynamics of fluid systems, (2018). http://arxiv.org/abs/1808.01346.

[32] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T.L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, O. Ronneberger, U-Net: deep learning for cell counting, detection, and morphometry, Nat. Methods. 16 (2019) 67–70. https://doi.org/10.1038/s41592-018-0261-2.

[33] L. Ma, S. Kashanj, S. Xu, J. Zhou, D.S. Nobes, M. Ye, Flow Reconstruction and Prediction Based on Small Particle Image Velocimetry Experimental Datasets with Convolutional Neural Networks, Ind. Eng. Chem. Res. 61 (2022) 8504–8519. https://doi.org/10.1021/acs.iecr.1c04704.

[34] C. Wei, G. Xue-bao, T. Feng, S. Ying, W. Wei-hong, S. Hong-ri, K. Xuan, Seismic velocity inversion based on CNN-LSTM fusion deep neural network *, 18 (2021). https://doi.org/10.1007/s11770-021-0913-3.

[35] F. Aksan, Y. Li, V. Suresh, CNN-LSTM vs . LSTM-CNN to Predict Power Flow Direction :, (2023).

[36] D. Wang, C. Xiang, Y. Pan, A. Chen, X. Zhou, Y. Zhang, A deep convolutional neural network for topology optimization with perceptible generalization ability, Eng. Optim. 54 (2022) 973–988. https://doi.org/10.1080/0305215X.2021.1902998.

[37] N. Zhang, S. Yan, Q. Ma, X. Guo, Z. Xie, X. Zheng, A CNN-supported Lagrangian ISPH model for free surface flow, Appl. Ocean Res. 136 (2023) 103587. https://doi.org/10.1016/j.apor.2023.103587.

[38] C.W. Dawson, R.L. Wilby, A comparison of artificial neural networks used for river flow forecasting, (1999).

[39] L. Li, K. Ota, M. Dong, Everything is Image : CNN-based Short-term Electrical Load Forecasting for Smart Grid, (2017). https://doi.org/10.1109/ISPAN-FCST-ISCC.2017.78.

[40] J. Kim, C. Lee, Prediction of turbulent heat transfer using convolutional neural networks, (2020). https://doi.org/10.1017/jfm.2019.814.

[41] Q. Zhang, Q. Jin, J. Chang, S. Xiang, C. Pan, Kernel-Weighted Graph Convolutional Network : A Deep Learning Approach for Traffic Forecasting, 2018 24th Int. Conf. Pattern Recognit. (2018) 1018–1023.

[42] X. Lu, B. Li, Y. Yue, Q. Li, J. Yan, Grid R-CNN, (n.d.) 7363–7372.

[43] X. Jin, P. Cheng, W. Chen, H. Li, Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder, (2018). https://doi.org/10.1063/1.5024595.

[44] A. Zhu, X. Li, Z. Mo, H. Wu, Wind Power Prediction Based on a Convolutional Neural Network, (2017).

[45] H. Fan, R. Wang, Y. Huo, H. Bao, Real-time Monte Carlo Denoising with Weight Sharing Kernel Prediction Network, 40 (2021). https://doi.org/10.1111/cgf.14338.

[46] N. Barkataki, B. Tiru, U. Sarma, A CNN model for predicting size of buried objects from GPR B-Scans, J. Appl. Geophys. 200 (2022) 104620. https://doi.org/10.1016/J.JAPPGEO.2022.104620.

[47] J. Seo, R.K. Kapania, Topology optimization with advanced CNN using mapped physics-based data, Struct. Multidiscip. Optim. 66 (2023) 1–20. https://doi.org/10.1007/s00158-022-03461-0.

[48] N. Thuerey, K. Weißenow, L. Prantl, X. Hu, Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows, AIAA J. 58 (2020) 25–36. https://doi.org/10.2514/1.J058291.

[49] Y. Liao, X. Huang, Y. Geng, Q. Yuan, D. Hu, UNet-Based Framework for Predicting the Waveform of Laser Pulses of the Front-End System in a Current High-Power Laser Facility, Photonics. 10 (2023). https://doi.org/10.3390/photonics10111244.

[50] X. Chen, X. Yao, Z. Zhou, Y. Liu, C. Yao, K. Ren, DRs-UNet: A Deep Semantic Segmentation Network for the Recognition of Active Landslides from InSAR Imagery in the Three Rivers Region of the Qinghai–Tibet Plateau, Remote Sens. 14 (2022). https://doi.org/10.3390/rs14081848.

[51] S. Sun, Y. Zheng, G. Li, Z. Guo, Z. Bie, J. Ma, Missing Data Reconstruction Method of Distribution Network based on RES-AT-UNET, China Int. Conf. Electr. Distrib. CICED. 2022-Septe (2022) 508–512. https://doi.org/10.1109/CICED56215.2022.9929094.

[52] J. Zheng, J. Li, Y. Li, L. Peng, A benchmark dataset and deep learning-based image reconstruction for electrical capacitance tomography, Sensors (Switzerland). 18 (2018). https://doi.org/10.3390/s18113701.

[53] R. Yamashita, M. Nishio, R. Kinh, G. Do, K. Togashi, Convolutional neural networks : an overview and application in radiology, (2018) 611–629.

[54] M. Sadeghi, P. Nguyen, K. Hsu, S. Sorooshian, Improving near real-time precipitation estimation using a U-Net convolutional neural network and geographical information, Environ. Model. Softw. 134 (2020) 104856. https://doi.org/10.1016/j.envsoft.2020.104856.

[55] S. Guan, K.-T. Hsu, M. Eyassu, P. V. Chitnis, Dense Dilated UNet: Deep Learning for 3D Photoacoustic Tomography Image Reconstruction, (2021) 1–9. http://arxiv.org/abs/2104.03130.

[56] J. Feng, J. Deng, Z. Li, Z. Sun, H. Dou, K. Jia, End-to-end Res-Unet based reconstruction algorithm for photoacoustic imaging, Biomed. Opt. Express. 11 (2020) 5321. https://doi.org/10.1364/boe.396598.

[57] S. Ruder, An overview of gradient descent optimization, (2016) 1–14.

[58] M. Despotovic, V. Nedic, D. Despotovic, S. Cvetanovic, Evaluation of empirical models for predicting monthly mean horizontal diffuse solar radiation, 56 (2016) 246–260. https://doi.org/10.1016/j.rser.2015.11.058.

[59] B. Julian, L. Roland, U-FLOOD – Topographic deep learning for predicting urban pluvial flood water depth, 603 (2021). https://doi.org/10.1016/j.jhydrol.2021.126898.

[60] H. Taniguchi, N. Kohira, T. Ohnishi, H. Kawahira, M. von und zu Fraunberg, J.E. Jääskeläinen, M. Hauta-Kasari, Y. Iwadate, H. Haneishi, Improving convenience and reliability of 5-ALA-induced fluorescent imaging for brain tumor surgery, 2015. https://doi.org/10.1007/978-3-319-24574-4_25.

[61] Z. Chao, F. Pu, Y. Yin, B. Han, X. Chen, Research on real-time local rainfall prediction based on MEMS sensors, J. Sensors. 2018 (2018) 1–9. https://doi.org/10.1155/2018/6184713.

[62] M. Cheng, A. Galimzianova, Ž. Lesjak, Ž. Špiclin, C.B. Lock, D.L. Rubin, A multi-scale multiple sclerosis lesion change detection in a multi-sequence mri, 2018. https://doi.org/10.1007/978-3-030-00889-5_40.

[63] D.J. Booker, M.J. Dunbar, Predicting river width , depth and velocity at ungauged sites in

England and Wales using multilevel models TH, 4057 (2008) 4049–4057. https://doi.org/10.1002/hyp.

[64] A. Hauet, T. Morlot, L. Daubagnan, Velocity profile and depth-averaged to surface velocity in natural streams : A review over a large sample of rivers, 06015 (2018).

[65] I. Dolzhikova, B. Abibullaev, A. Zollanvari, A Jackknife-Inspired Deep Learning Approach to Subject-Independent Classification of EEG, Pattern Recognit. Lett. 176 (2023) 28–33. https://doi.org/10.1016/j.patrec.2023.10.011.

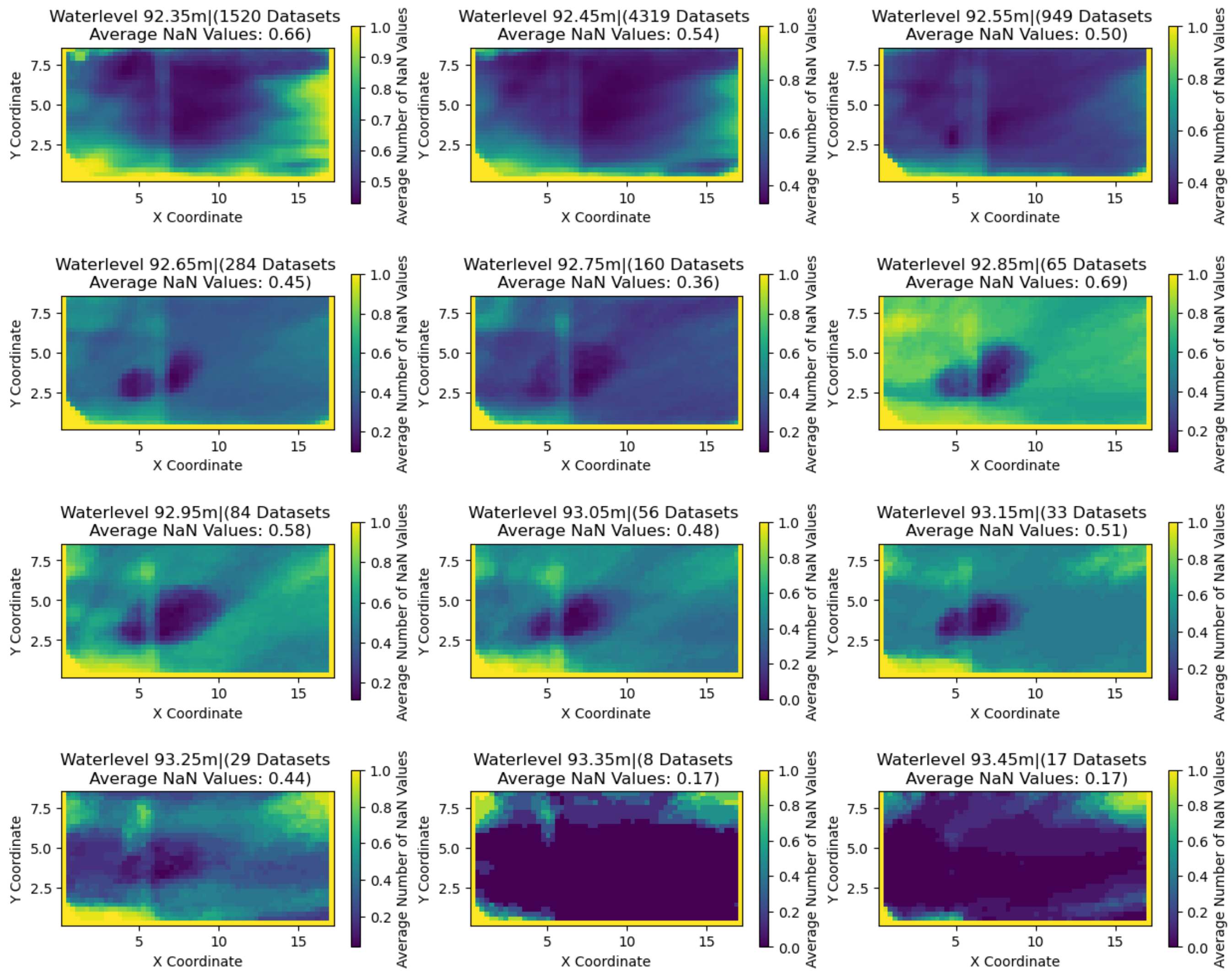
[66] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next, J. Sci. Comput. 92 (2022) 1–62. https://doi.org/10.1007/s10915-022-01939-z.

[67] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, Acta Mech. Sin. Xuebao. 37 (2021) 1727–1738. https://doi.org/10.1007/s10409-021-01148-1.

[68] S. Choi, I. Jung, H. Kim, J. Na, J.M. Lee, Physics-informed deep learning for data-driven solutions of computational fluid dynamics, Korean J. Chem. Eng. 39 (2022) 515–528. https://doi.org/10.1007/s11814-021-0979-x.

[69] M.H. Malone, N. Sciaky, L. Stalheim, K.M. Hahn, E. Linney, G.L. Johnson, Laser-scanning velocimetry: A confocal microscopy method for quantitative measurement of cardiovascular performance in zebrafish embryos and larvae, BMC Biotechnol. 7 (2007) 1–11. https://doi.org/10.1186/1472-6750-7-40.

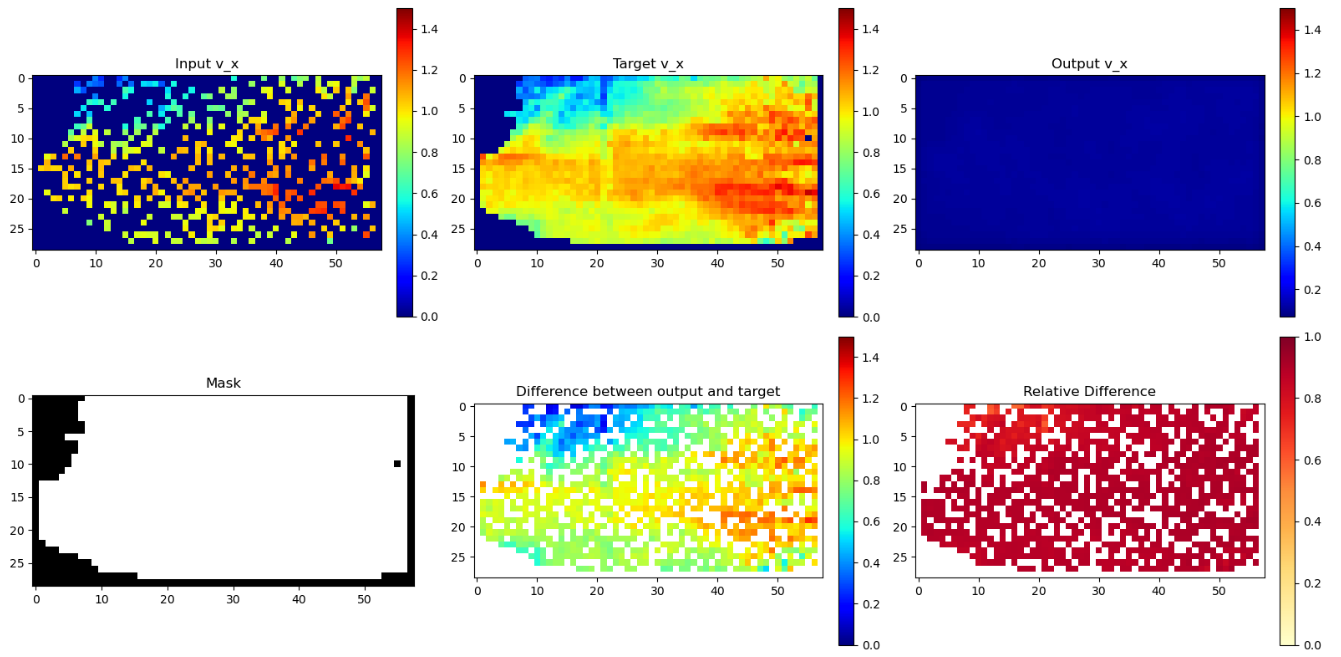
Appendix A: Average NaN values across all the datasets

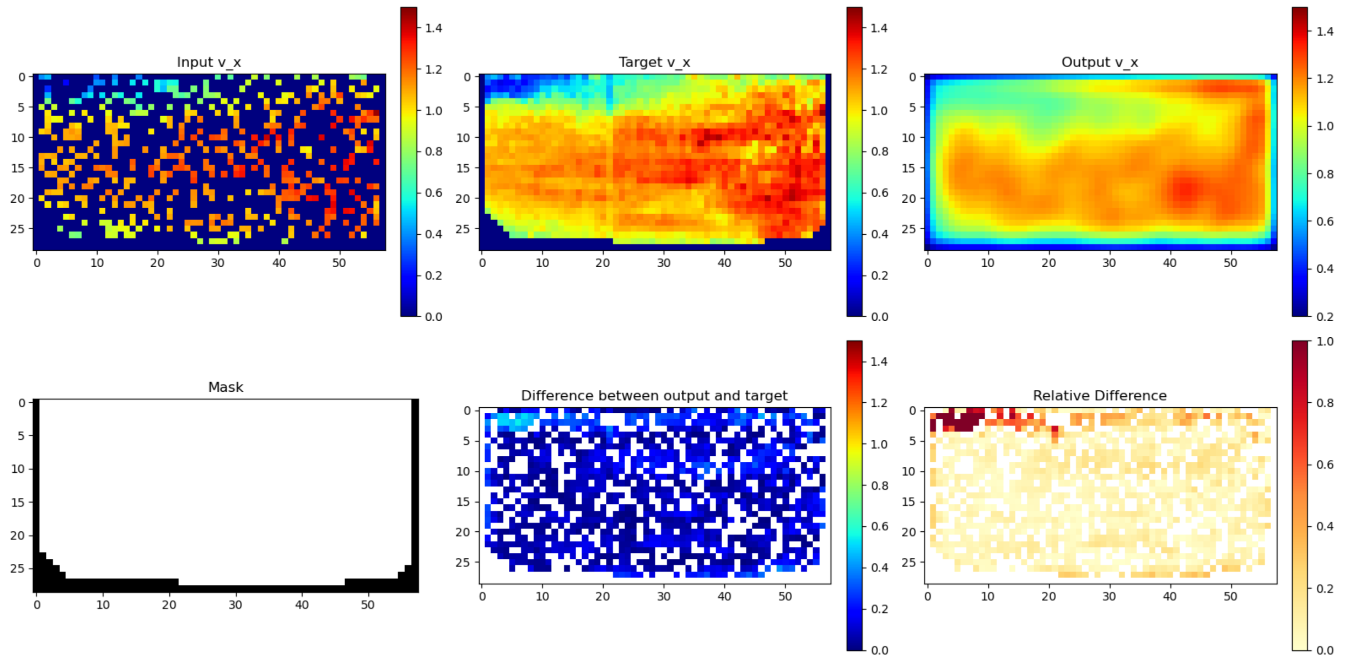


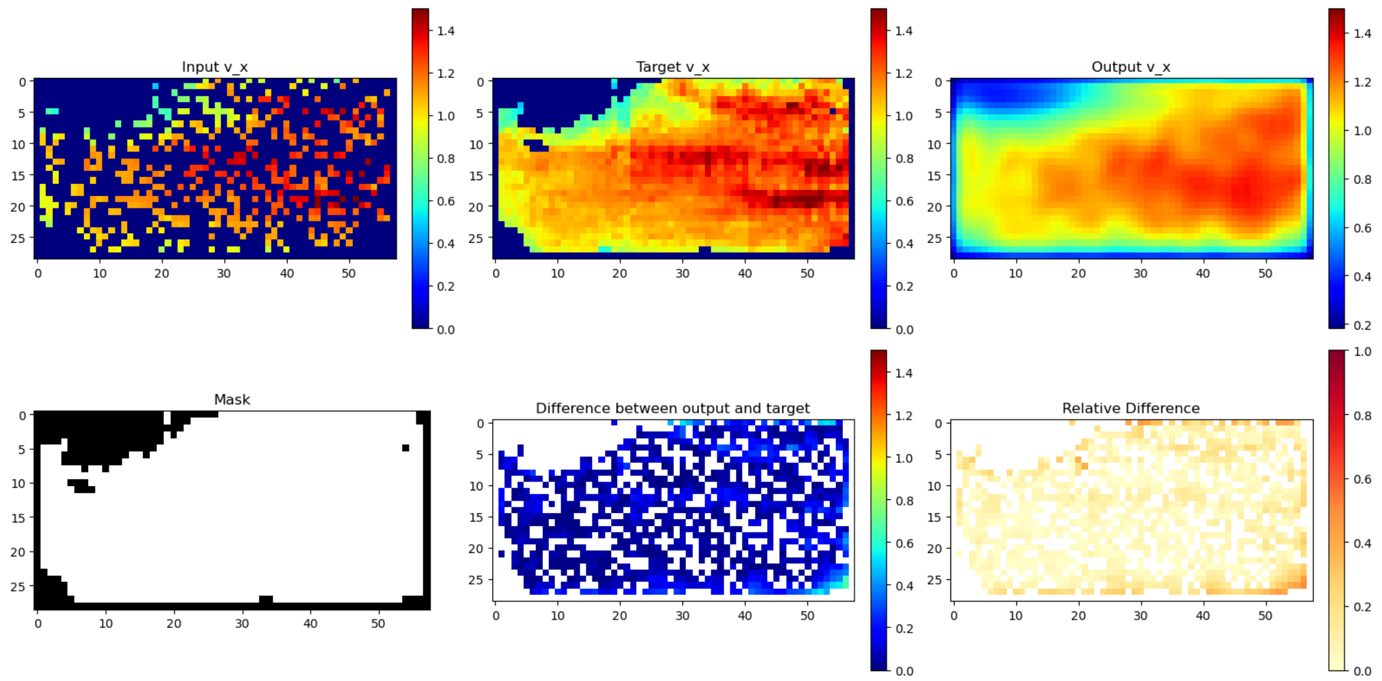
Appendix B: Code for the U-Net Architecture

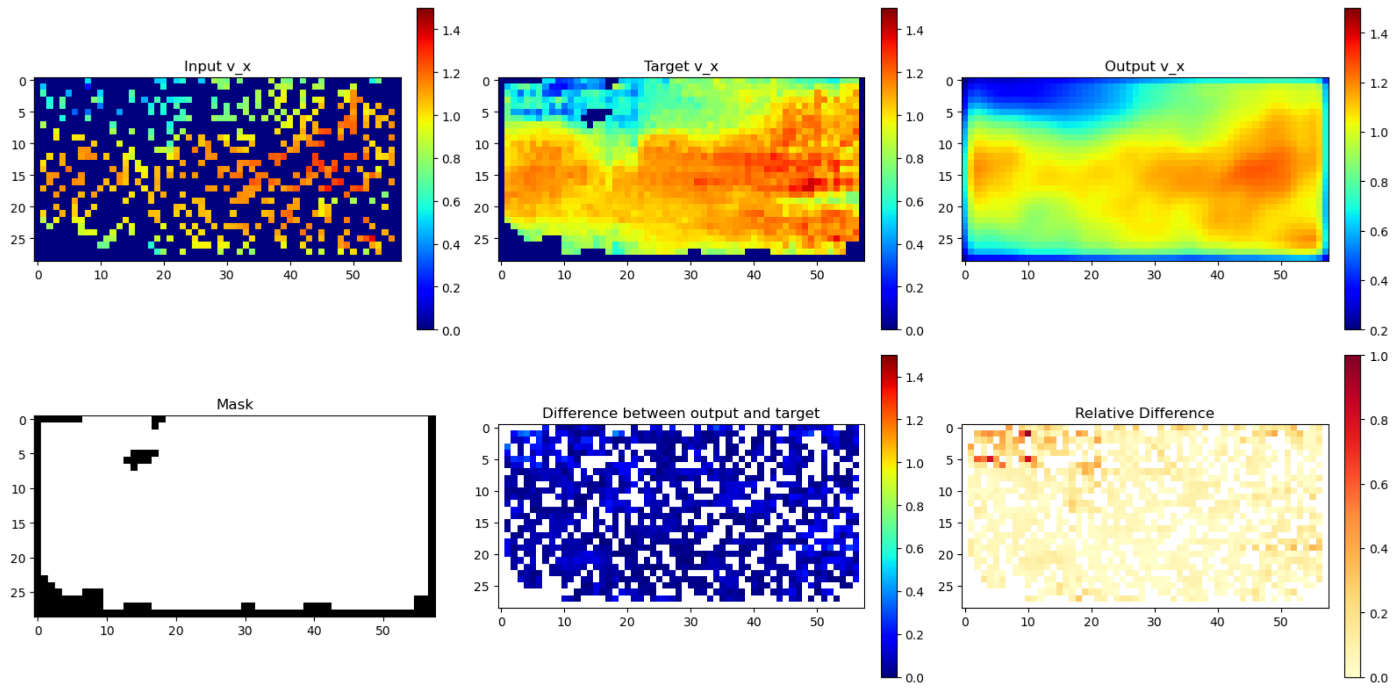
```
class DoubleConv(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(DoubleConv, self).__init__()
        self.double_conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        return self.double_conv(x)

class Down(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(Down, self).__init__()
        self.maxpool_conv = nn.Sequential(
            nn.MaxPool2d(2),
            DoubleConv(in_channels, out_channels)
        )

    def forward(self, x):
        return self.maxpool_conv(x)

class Up(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(Up, self).__init__()
        self.up = nn.Upsample(scale_factor=2, mode='bilinear',
align_corners=True)
        self.conv = DoubleConv(in_channels, out_channels)

    def forward(self, x1, x2):
        x1 = self.up(x1)
        diffY = x2.size()[2] - x1.size()[2]
        diffX = x2.size()[3] - x1.size()[3]
        x1 = nn.functional.pad(x1, [diffX // 2, diffX - diffX // 2, diffY //
2, diffY - diffY // 2])
        x = torch.cat([x2, x1], dim=1)
        return self.conv(x)

class OutConv(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(OutConv, self).__init__()
        self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=1)

    def forward(self, x):
        return self.conv(x)

class UNet(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(UNet, self).__init__()
        self.inc = DoubleConv(in_channels, 64)
        self.down1 = Down(64, 128)
        self.down2 = Down(128, 256)
        self.down3 = Down(256, 512)
        self.down4 = Down(512, 512)
        self.up1 = Up(1024, 256)
        self.up2 = Up(512, 128)
        self.up3 = Up(256, 64)
        self.up4 = Up(128, 64)
        self.outc = OutConv(64, out_channels)

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits

# Instantiate the U-Net model
model_unet = UNet(in_channels=2, out_channels=2) # 2 input channels (v_x, v_y)
and 2 output channels
```

Appendix C: Training progress Results after 1,6, 11 and 20 epochs

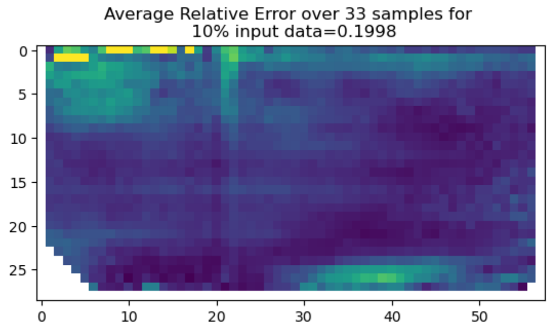


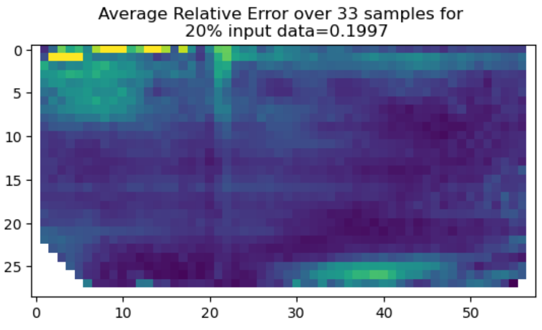


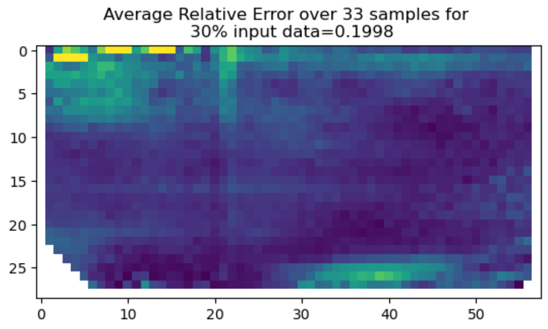

Input v_x
Target v_x
Output v_x
Mask
Difference between output and target
Relative Difference

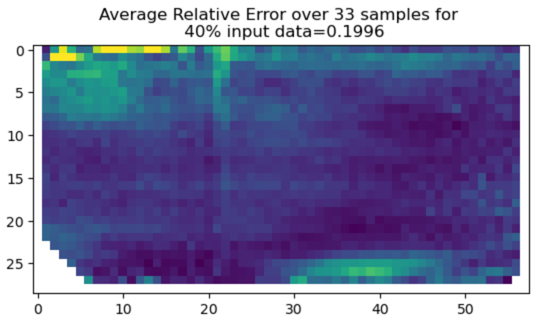

Input v_x
Target v_x
Output v_x
Mask
Difference between output and target
Relative Difference

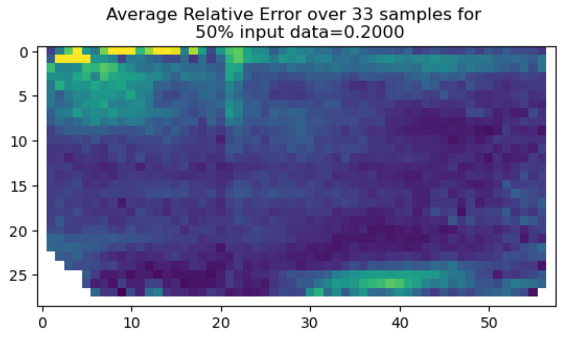
Appendix D: Relative Errors from the Benchmark Model

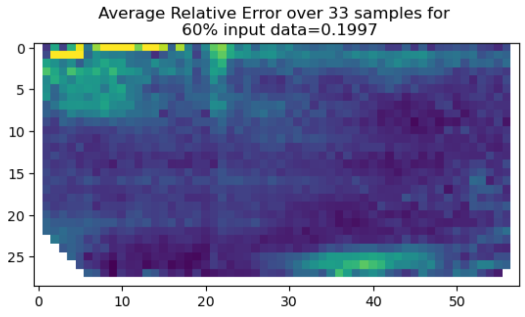


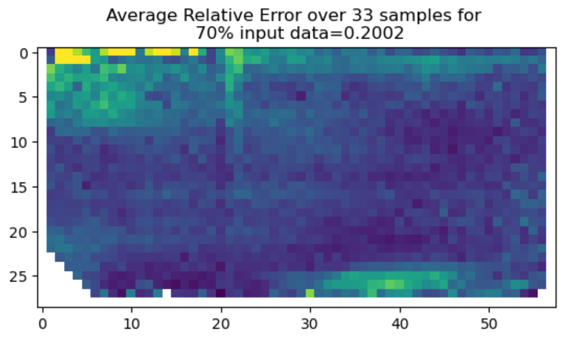


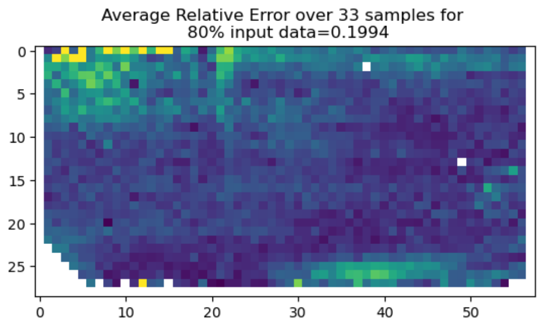


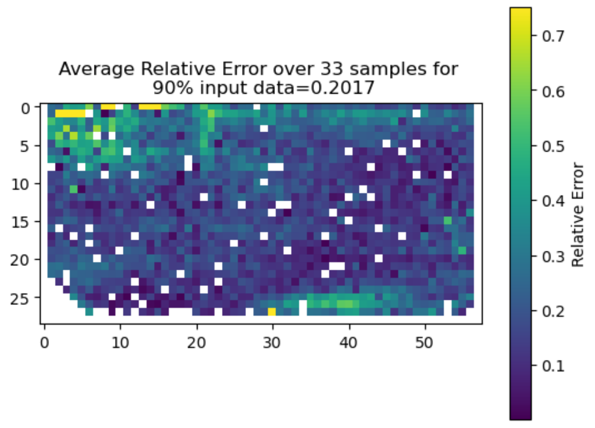




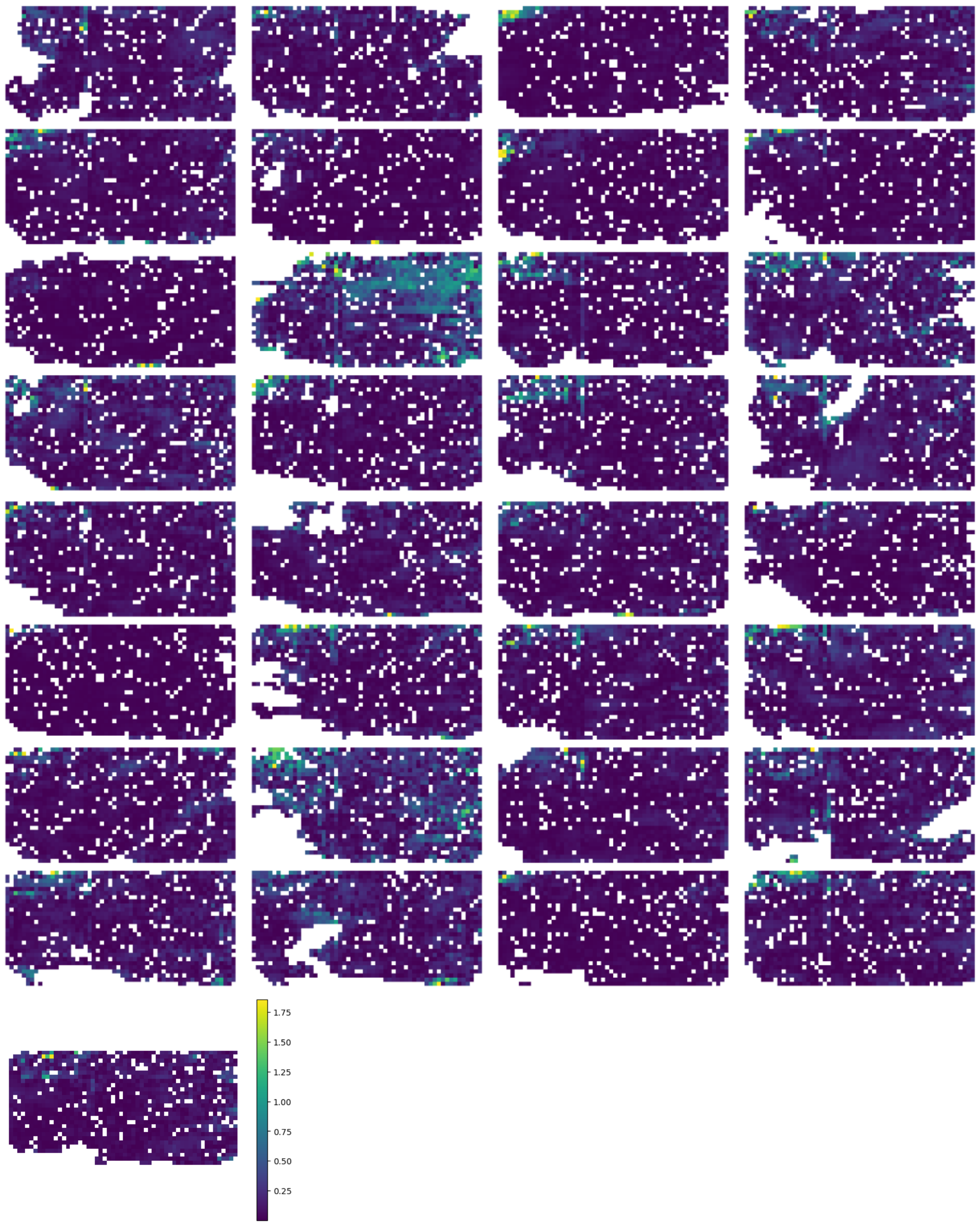




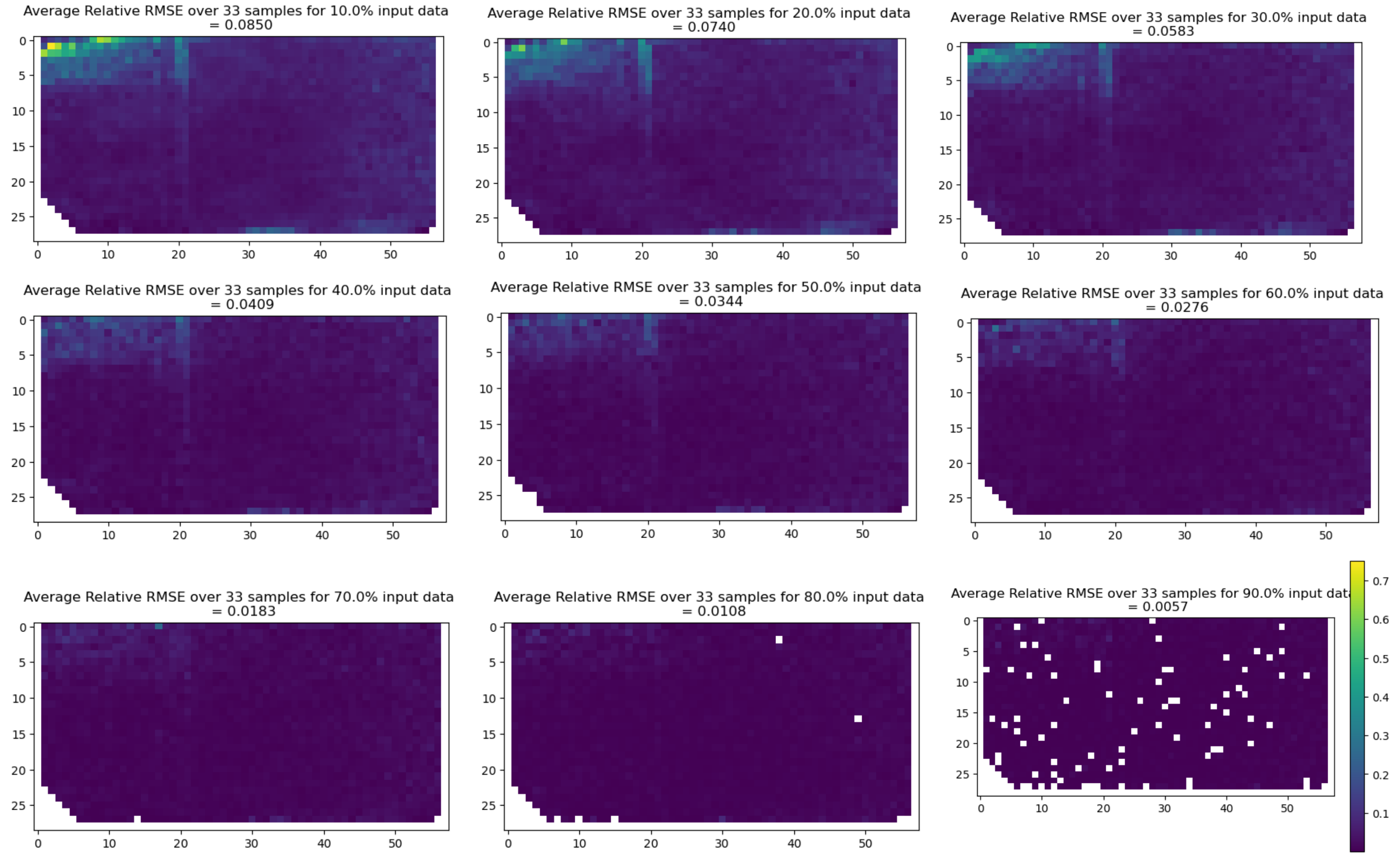




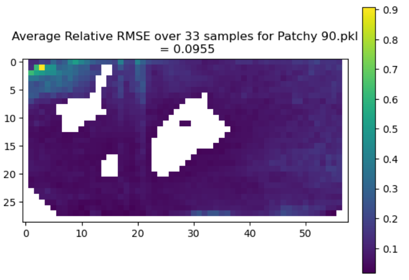
Appendix E: Complete set of relative errors of U-Net predicted data with 10% of data input

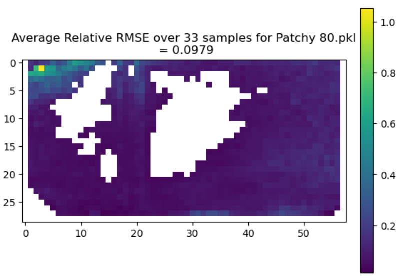


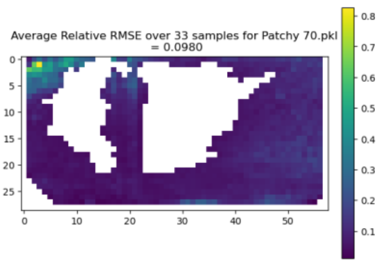
Appendix F: Relative Root Mean Squared Errors of the U-Net model with different percentages of random masks

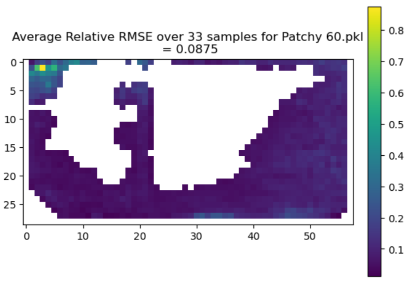


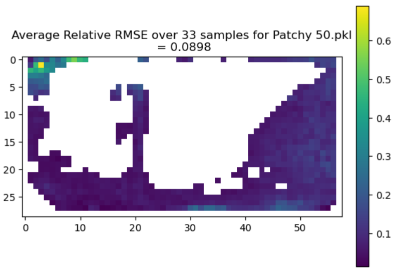
Appendix G: Result of Relative RMSE averaging over all 33 datasets for patchy masks

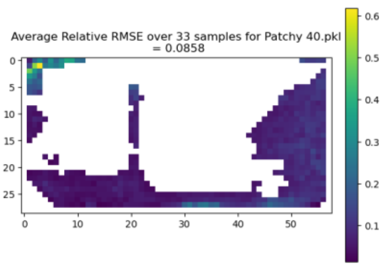

Average Relative RMSE over 33 samples for Patchy 90.pkl
= 0.0955

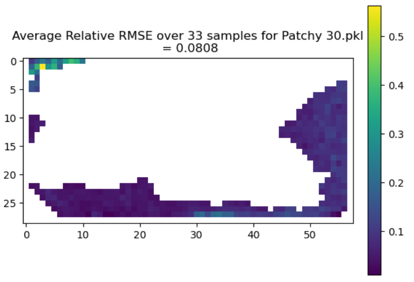

Average Relative RMSE over 33 samples for Patchy 80.pkl
= 0.0979

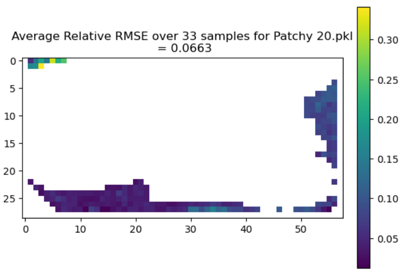

Average Relative RMSE over 33 samples for Patchy 70.pkl
= 0.0980

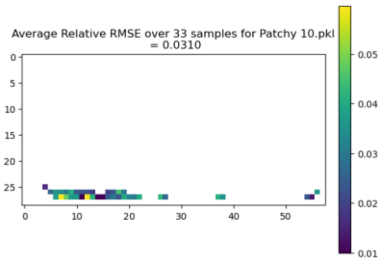

Average Relative RMSE over 33 samples for Patchy 60.pkl
= 0.0875


Average Relative RMSE over 33 samples for Patchy 50.pkl
= 0.0898


Average Relative RMSE over 33 samples for Patchy 40.pkl
= 0.0858


Average Relative RMSE over 33 samples for Patchy 30.pkl
= 0.0808


Average Relative RMSE over 33 samples for Patchy 20.pkl
= 0.0663


Average Relative RMSE over 33 samples for Patchy 10.pkl
= 0.0310