IMPROVING NEURAL NETWORKS TRAINED WITH LIMITED DATA BY DERIVING IMPROVEMENTS BASED ON KNOWLEDGE OF THE UNDERLYING PHYSICS INTERACTIONS

Jose G. Perez

Department of Computer Science

IMPROVING NEURAL NETWORKS TRAINED WITH LIMITED DATA BY DERIVING IMPROVEMENTS BASED ON KNOWLEDGE OF THE UNDERLYING PHYSICS INTERACTIONS

by

Jose G. Perez

THESIS PROPOSAL

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

January 20, 2025

Table of Contents

Page

Table of Contents . . . iii

Abstract . . . v

1 Introduction . . . vi

1.1 The Need For Data In Neural Networks . . . vi

1.2 The Difficulty of Getting Data . . . vi

1.3 Working Around Data Limitations With Few-Shot Learning . . . vii

1.3.1 Transfer Learning . . . viii

1.3.2 Data Augmentation . . . ix

1.3.3 Meta Learning . . . ix

1.4 Few-Shot Learning By Integrating Physics Into Neural Networks . . . xi

1.5 Specific Problems From Other Disciplines That Have Limited Datasets . . . xii

1.5.1 Fluid Flow Velocity Prediction . . . xii

1.5.2 Glacial Ice Segmentation . . . xv

1.6 Thesis Statement . . . xvii

1.7 Expected Contributions . . . xviii

2 Physics-Informed LSTM Network For Velocity Prediction of Fluid Flow Simulations xx

2.1 The Importance of Fluid Flow Velocity Prediction . . . xx

2.2 Long Short-Term Memory Networks (LSTMs) For Time Series Data . . . xxi

2.3 Adding Physics to LSTMs . . . xxii

2.3.1 Fluid Flow Velocity Data From Simulations . . . xxiii

2.3.2 Measuring Network Performance With The Mean Squared Error . . . xxv

2.3.3 Proposed Network Architecture . . . xxvi

2.3.4 Baseline Used To Compare Against Our Proposed Network . . . xxix

2.3.5 Experimental Results . . . xxix

2.3.6 Conclusion xxx

3 Glacial Ice Segmentation of the HKH Region With Physics-Informed Data Augmentation xxxi

3.1 The Importance of Glacial Ice Segmentation xxxi

3.2 Image Semantic Segmentation: Grouping Similar Pixels Together xxxi

3.3 Glacier Mapping Through Segmentation of Ice in Hyperspectral Images . . xxxii

3.4 Convolutional Neural Networks: The Backbone of Almost All Networks That Use Images xxxiii

3.5 UNet: The Standard Network For Image Segmentation xxxiv

3.5.1 Measuring Network Performance With IoU xxxvi

3.6 Adding Physics To Image Segmentation Through Data Augmentation . . . xxxvii

3.6.1 Glacier Mapping Data From ICIMOD And Landsat7 Satellite . . . xli

3.6.2 Baseline Network xli

3.6.3 Experimental Results xlii

3.6.4 Conclusion xlii

4 Physics-Informed Network For Glacial Ice Velocity Predictions xliii

5 Physics-Informed MANet for Glacial Ice Segmentation of the HKH Region . . . xliv

Abstract

Deep neural network models are state-of-the-art for many image, audio, text, and video processing problems in different fields of study and disciplines. However, training many of these networks can require a lot of data and gathering such data can be time consuming, costly, and difficult to set-up. This limiting factor can prevent researchers and engineers that do not have access to a lot of resources from using all the tools and models available in deep learning to tackle novel problems in their respective fields. To work around these data limitations, many techniques and models in the field of Few-Shot Learning have been proposed over the years in the sub-fields of transfer learning, data augmentation, and meta learning. One such model, Physics-Informed Neural Networks or PINNs, relies on the fact that datasets collected in the real world must follow the laws of physics, allowing us to leverage our knowledge of physics to help improve performance on these datasets without requiring more data to be gathered. I propose 4 ways of extending this type of approach that leverages our knowledge of physics through new models and data augmentation techniques in this dissertation, and apply these approaches to two problems that have limited datasets: the problem of Fluid Flow Velocity Prediction in the field of mechanical engineering and the problem of Glacial Ice Segmentation from geology.

A detailed explanation of the importance of data gathering, difficulties and limitations, and my expected contributions for this area along with a timeline are presented in Chapter 1. The remaining chapters each describe one of the four expected contributions of my dissertation in chronological order.

Chapter 1

Introduction

1.1 The Need For Data In Neural Networks

Deep neural network models dominate as the state-of-the-art machine learning (ML) models that have the best performance across many different image [1] [2] [3], text [4], and video processing tasks applied to multiple disciplines and fields of study. From text generation with large language models (LLMs) such as GPT-3 [5], object tracking used by the space sector to analyze the sun [6] and other galactic bodies, object detection used by self-driving cars [7], to the rise of mobile apps for synthetic audio and image generation that has caught the attention of regular everyday people. Although there are different architectures and different ways to train these networks depending on the task being considered, there is a big commonality among all machine learning (ML) models which is the need to have a lot of data to train these models.

1.2 The Difficulty of Getting Data

Data gathering is one of the most fundamental and important problems in machine learning (ML), as without data there is nothing to train your models with and the quality and quantity of your data makes a big impact on the performance of your given model. The state-of-the-art text generation model GPT-3 [5] was trained on 45TB of text data crawled from the publicly available Internet, Google trained their MLP-Mixer [8] and a few other models a 300 million image private subset of the images extracted from their search engine database called JFT-300M, and NVIDIA's StyleGan2 [9] research used the FFHQ dataset which has 70,000 images and the LSUN dataset which has 10 categories with each having from 120,000 to 3,000,000 images.

Although there have been efforts made in different fields to create uniform and useful datasets, data gathering is still one of the main limiting factors preventing researchers of other disciplines from using all the tools and models we have available in deep learning. Data gathering is typically very time consuming, difficult to set-up, and costly in terms of monetary expenses. Big tech companies don't struggle with this problem as much as university researchers and smaller companies do as they have bigger budgets and more staff available for data collection purposes. Even though the research groups as these companies sometimes make their datasets publicly available for others to use [10] [11] [12], these datasets which usually consist of RGB photographs are not useful to people in other fields of study to solve problems in their respective disciplines. The data needed to solve problems in neuroscience, geology, medicine, and other disciplines will often require advanced equipment, domain experts, and be stored in a different digital format than regular camera photographs.

1.3 Working Around Data Limitations With Few-Shot Learning

The challenges of data gathering and the eagerness of researchers to use neural networks has steadily given rise to a field called "Few-Shot Learning" over the years. The idea behind few-shot learning is to train deep neural network models with a "few" labeled data samples but still achieve good performance for the given problem. This way you can leverage as much power from these deep neural networks that you can with as little data as possible, making deep neural networks more accessible. Proposed few-shot learning approaches typically focus on taking one component of the traditional deep neural network training pipeline and adapting it to perform better with few training samples. There are

3 main types of approaches, and although I present these approaches as separate fields in the area of "few-shot learning" it is not uncommon to have one approach from each field as part of your final model.

1. **Transfer Learning** - Approaches where you take pre-trained models trained on large datasets for a similar problem to yours and then "transfer" some of the learned knowledge to your limited data problem.
2. **Data Augmentation** - Approaches that propose ways to generate more data samples from your existing ones to increase your dataset size.
3. **Meta Learning** - Approaches where your network "learns to learn" or extracts some useful knowledge on how to train your network for your problem by training other networks on subsets of other similar data and similar problems in what is called "episodic training".

1.3.1 Transfer Learning

In transfer learning the goal is to take models that are already trained and then "transfer" as much useful knowledge as possible to your specific problem. This is done by simply loading a trained network, freezing some of the layers so they do not get updated while training anymore, removing the last layer which is usually problem dependent, and adding a last layer that fits your specific problem.

However, this only works well when the network is pre-trained on data that is similar to yours. The bigger the gap between the domains and problems used for the pre-trained network and your own, the worst the performance will be with transfer learning. Because of this big problem, the cases where you can actually use transfer learning and get good results are few unless you are working with RGB images for the common tasks of object classification or detection. If you are using a different type of input data for a different task, the odds that you will find a pre-trained model with similar data to yours becomes much lower.

1.3.2 Data Augmentation

In data augmentation the goal is to take your existing limited data and generate "new" samples from them to increase the size of your dataset. If you have image data, one of the simplest and most popular approaches is to perform simple image processing operations like rotation, cropping, re-sizing, mirroring, and more. However, this requires careful thinking as not all operations will produce good augmentations. For example, for the MNIST [13] hand-written digit dataset it does not make sense to flip or mirror images as the new images will not represent the same digit as the original (a flipped 6 can become a 9). This is one of the disadvantages of data augmentation, the augmentation must still represent the same label as the original data so if your data is not based on images or is complex it may be difficult to come up with rules for data augmentation.

1.3.3 Meta Learning

In meta-learning the goal is to extract useful information that can be used to train your model by training other models and learning how and what those models learned (meta-data). For example, let's take the random initialization of networks. Although you can use a uniform or normal distribution, MAML [14] proposes learning an initialization that leads to maximal performance on a new problem with a few training steps and a small amount of data. This is done by feeding entire "problems" as input data and optimizing the model to produce good results with as few training steps as possible. Therefore you train this model on other problems, and then you feed your specific problem to it so you can also produce good results with few training steps. The drawback of this approach is that it can be computationally expensive to train "entire problems" multiple times while optimizing this architecture, the problem of vanishing/exploding gradients is still present in this network, and selecting which extra problems and datasets to use can be a problem as well.

Another popular meta-learning approach is called metric learning. The idea is pretty simple, you might only have a few pictures of zebras and a few pictures of dogs so to classify a new picture you can just compare it to a few of each and find which is the most similar. Thus the main principle is to train models that can compute similarities through metrics. Siamese Networks [15] are one of the earliest few-shot learning methods proposed to tackle the limited data problem and this approach focuses on metric learning. Siamese Networks take three inputs, a sample from your dataset called an anchor, a sample similar to your anchor called the positive sample, and a sample different to your anchor called a negative sample. Then the network is trained to produce embeddings (1D vectors) such that similar samples are close in that metric space and different samples are further apart, allowing you to use Euclidean distance to find similar samples. This idea was further improved in Prototypical Networks [16], where the idea is now to find "prototypes" or samples that represent a general class of similar objects instead of having the triplets. For example, you may find one of your pictures is a good representation of "mammals" and can make it easier to identify new mammals in the future. In Relation Networks [17] this metric learning idea is generalized even more. Specifically, the authors propose an end-to-end two-part network that is trained to learn a custom distance metric used to compare pairs of images called a "relation score" instead of always using Euclidean distance. Then when a new image needs to be classified, we can compare it against our few labeled samples and find the most similar one or the one with the highest "relation score". As the distance metric used for comparison is custom learned, the model performs better than if you used general distance metrics like Euclidean distance or Cosine similarity like in [16] or [15]. These type of architectures, however, are typically designed for image classification problems and cannot be used as-is for more complex tasks like image segmentation or object detection or for other types of input data that are not regular camera RGB images.

1.4 Few-Shot Learning By Integrating Physics Into Neural Networks

One of the reasons why humans might be able learn new things with few examples and do few-shot learning is because we have a lot of previous knowledge that we leverage in our learning process and we never truly start from scratch as deep neural networks do. My research focus is then on how we can integrate our knowledge of physics to improve neural network performance and have models more "easily learn" with few examples, mimicking humans. Physics is a widely studied area of science that has been researched for hundreds of years for which we not only have a lot of mathematical equations but also a lot of expertise and knowledge in the field. As many problems across different disciplines are based on real-world data they must follow the "laws of physics" and so we can leverage our knowledge of physics to help improve performance on limited datasets that deal with real world data.

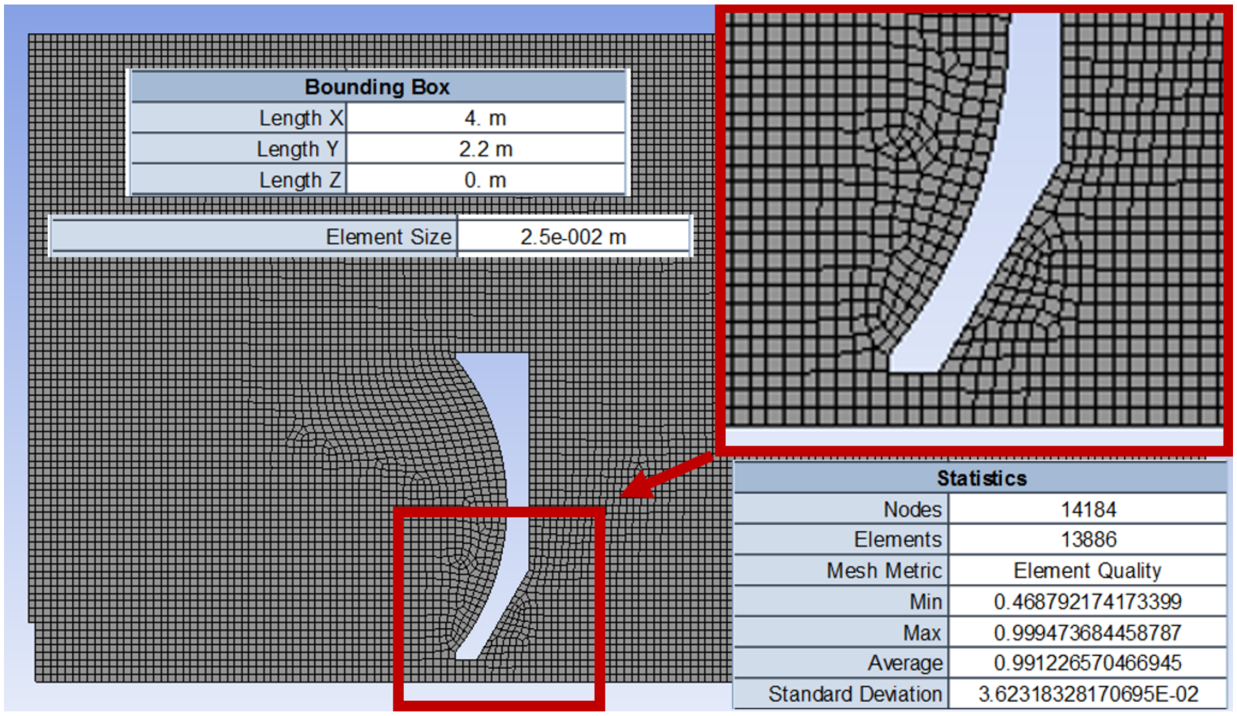
There exists an area of study called Physics-Informed Neural Networks (PINNs) [18] where the aim is to integrate physics that can be described as systems of partial differential equations (PDEs) directly into the loss function of a network. Unless you are working with simulation data, it will be close to impossible to collect all the variables and data needed to describe how your specific problem behaves exactly at all times. There is just too much data at too many different scales, from quantum effects all the way to space-time gravitational influences. Due to this, scales and times are often discretized when working with these PDEs like when using the Finite Element Method (FEM) to compute solutions for these PDEs. PINNs allow you to model some of these intrinsic variables of your data and use them along with the laws of physics (PDEs) that describe how they interact to increase your model performance. I will be extending this area by creating new network models and loss functions that will allow us to integrate PINNS and other neural network models to solve problems from two disciplines that have limited datasets.

1.5 Specific Problems From Other Disciplines That Have Limited Datasets

As my research focus is on applying physics to neural networks to improve model performance on problems with limited datasets I selected problems from two disciplines that had limited data available and whose data must follow the “laws of physics” in some way. Those problems were “Fluid Flow Velocity Prediction” from mechanical engineering and “Glacial Ice Segmentation” from geology.

1.5.1 Fluid Flow Velocity Prediction

Understanding how fluids flow is very important for the study and development of airplanes, cars, boats, rockets, and much more. One of the widely researched approaches used to study fluid flow is setting up Computational Fluid Dynamic (CFD) simulations using software developed specifically for that purpose like Ansys Fluent[19]. It is unfeasible to model every particle of every fluid we are interested in modeling at every scale and every point in space due to computational and time limitations, so it is necessary to define a discretized mesh of finite elements of specific size as shown in Figure 1.1 below.



Bounding Box	
Length X	4. m
Length Y	2.2 m
Length Z	0. m

Element Size	2.5e-002 m

Statistics	
Nodes	14184
Elements	13886
Mesh Metric	Element Quality
Min	0.468792174173399
Max	0.999473684458787
Average	0.991226570466945
Standard Deviation	3.62318328170695E-02

Figure 1.1: Example of a discretized mesh of finite elements used for a CFD fluid flow simulation.

After deciding the scale which we are interested in modeling and studying, the next step is to define the relevant geometry, fluids, and boundary conditions for the simulation as show in Figures 1.2 and 1.3 below. Lastly, a solver is selected and the simulation is run for a specific period of time.

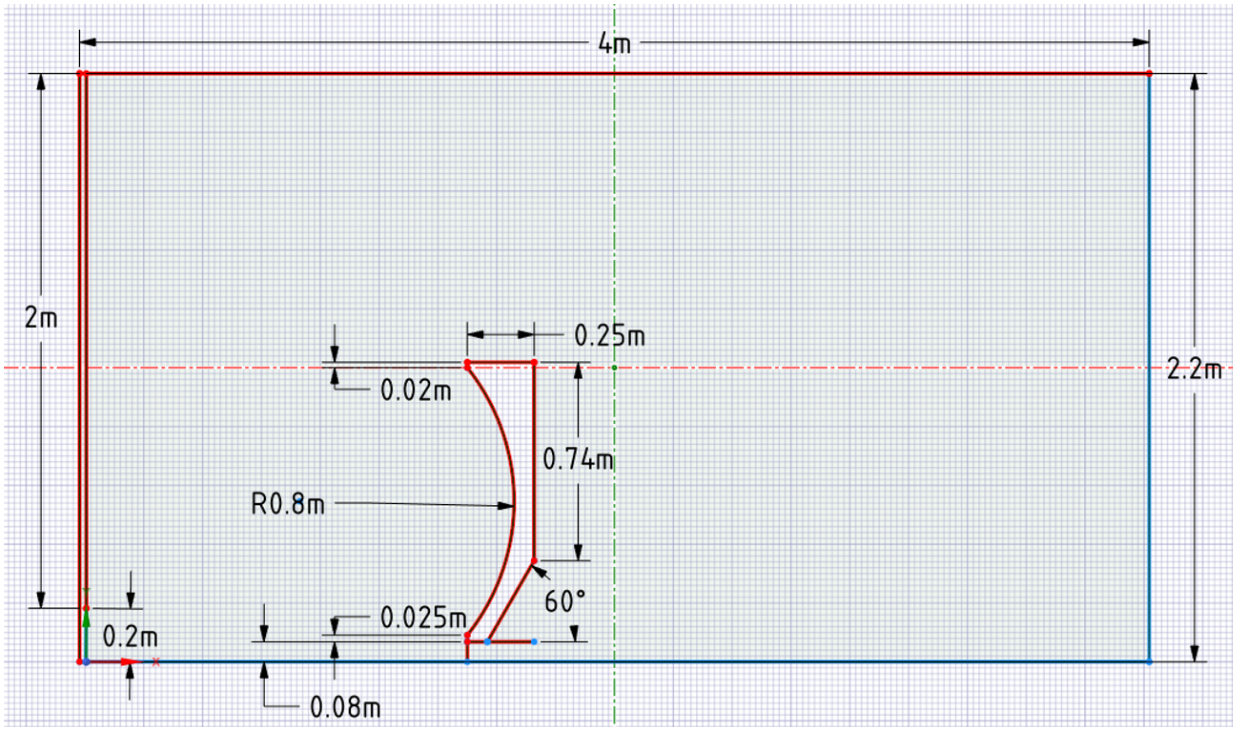


Figure 1.2: Example of geometry used for a CFD fluid flow simulation. This modeled geometry is for a proposed water-braking mechanism for a pusher sled system used in the Holloman Air Force Base for experiments.

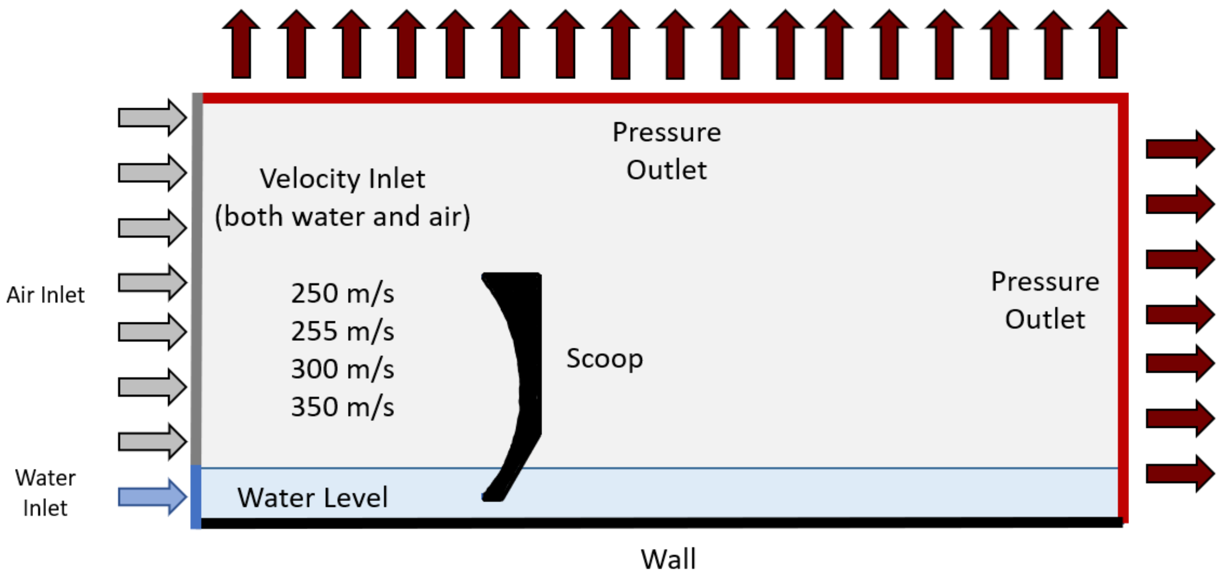


Figure 1.3: Example of boundary conditions used for a CFD fluid flow simulation. These boundary conditions model how the geometry (water-breaking scoop) will interact with the water as the pusher sled mechanism pushes the scoop at specific initial velocities.

One drawback of CFD simulations is that high-fidelity simulations can be very computationally expensive as having smaller finite elements, bigger meshes, bigger geometries, and bigger simulation environments increases the computations needed for the solvers to simulate each given time-step of the simulation. Although fluid flow datasets are available for some problems in fluid dynamics, many researchers are interested in fluid flows for specific problems with different geometries and boundary conditions than those in the available datasets. As fluid flows are a widely studied area in mechanical engineering in the field of fluid dynamics, we know that the Navier-Stokes equations can be used to describe some of these flows and that these equations are partial differential equations (PDEs) allowing us to leverage Physics-Informed Neural Networks [18] to tackle this limited data problem.

1.5.2 Glacial Ice Segmentation

Glaciers are a very important source of water for people and wildlife of many different regions of the world, as not only do they provide a source of drinking water but also water for watering crops and generating hydroelectric power [20]. As these regions rely heavily on glacial melt as a water source, it is important to monitor and keep track of changes that happen to these glaciers over time.

There exists multiple satellites such as NASA's Landsat-7, Landsat-8, and Sentinel-2 that have captured hyperspectral images of these glaciers over a long period of time, allowing glaciologists to take these images and use their expertise to determine what areas of the images are clean ice, debris covered ice (ice mixed with rocks), and regular rocks in a process called glacier mapping. This is one of the ways that these scientists monitor the glaciers over time, and in the area of computer vision is called image segmentation. An example of this is given in Figure 1.4 below.

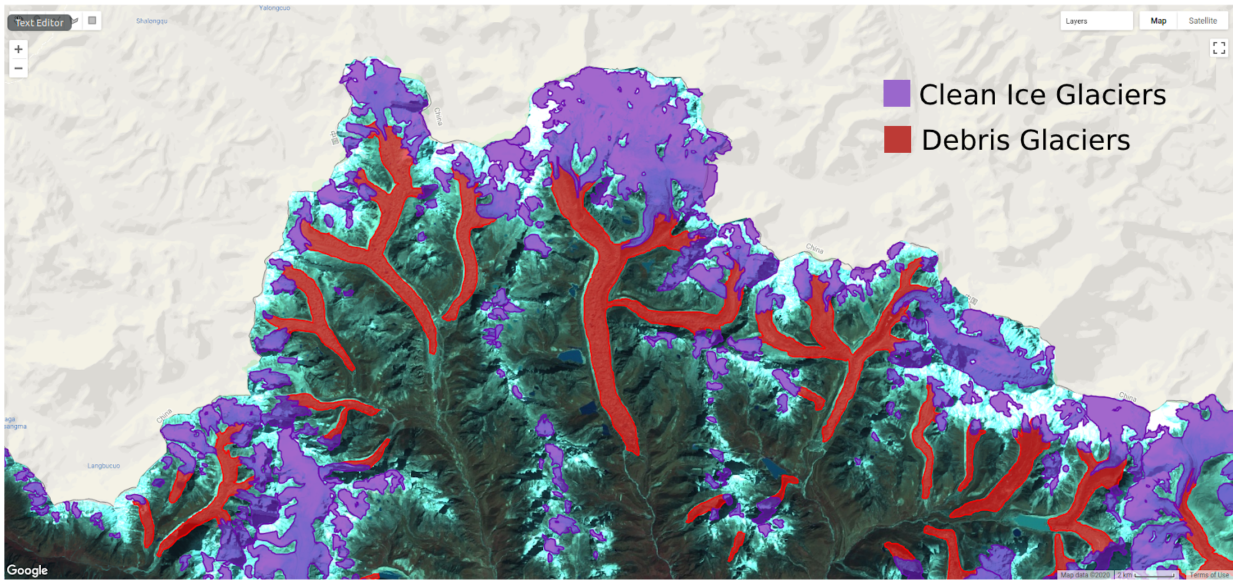


Figure 1.4: Example of glacier mapping or segmentation of a satellite image into clean ice and debris covered ice glaciers.

However, manually labelling hyperspectral satellite images into clean ice and debris covered ice glaciers is a very time consuming task. This is due to two main reasons:

- Satellite images have more channels than the regular 3 (RGB) in digital cameras, so they are hyperspectral images and require the usage of specialized tools (like QGIS) and knowledge about what channels capture which bands of the electromagnetic spectrum to properly visualize and analyze.

- High resolution images are difficult to segment as there are too many individual pixels, and although glaciologists use tools such as QGIS to label the images using geometric shapes instead of pixel by pixel to speed up the process it is still time consuming and difficult to label the borders/boundaries between the clean ice and debris covered ice specifically due to the amount of fine detail and care needed.

Labeling a single patch of a glacier using Sentinel-2 satellite imagery can take an expert 1 to 4 weeks depending on the complexity of the image, and the Hindu-Kush Himalayas (HKH) glaciers are made up of at least 256 of such patches.

A labeled dataset exists for the glaciers in Hindu-Kush Himalayas (HKH) region created by the International Centre for Integrated Mountain Development (ICIMOD) for images taken with NASA's Landsat-7 satellite. These images contain 7 channels including RGB, Near-Infrared, and Digital Elevation Map data.

There also exists a dataset of glacier ice velocities created by the National Snow and Ice Data Center containing the surface velocities of major glacier-covered regions in the world including the Hindu-Kush Himalayas (HKH) spanning from 1985 to 2018 and compiled from multiple of NASA's Landsat satellites called the "MEaSUREs ITS_LIVE Regional Glacier and Ice Sheet Surface Velocities" dataset. As glacial ice velocity prediction is a specialized form of fluid velocity prediction where the fluid being analyzed is ice from the glaciers, we can use the Navier-Stokes equations once again to leverage Physics-Informed Neural Networks [18] to help us tackle this limited data problem.

1.6 Thesis Statement

Incorporating physics in neural network models through modification of data and loss functions will allow for better performance and faster convergence for the problems of fluid flow velocity prediction and glacial ice segmentation.

1.7 Expected Contributions

The proposed contributions of this research are as follows:

1. **A physics-informed neural network model for the task of fluid flow velocity prediction** - We will develop a model that uses Physics-Informed Neural Networks (PINNs) [18] for the task of fluid flow velocity prediction and investigate ways to combine PINNs with networks built for sequential data (such as [21]) to take advantage of how fluid flows over time sequentially allowing for improvements on network convergence speed.

2. **A neural network model with physics-informed data augmentation for the task of glacier mapping** - As there already exists a network for glacier mapping by image segmentation [22], we will investigate ways to augment the current available labeled data for glacier mapping based on physics, improving the performance of the pre-existing segmentation model.

3. **A physics-informed neural network model for the task of glacial ice velocity predictions** - Glacial ice velocity prediction is a subset of the general fluid flow velocity prediction problem, allowing us to use Physics-Informed Neural Networks (PINNs) [18] combined with networks built for sequential data (such as [21] and [23]) to leverage the fact that glacial ice flows over time (sequentially) for the task of glacial ice velocity prediction.

4. **A physics-informed neural network model for the task of glacier mapping** - We will develop a new model based on a combination of Physics-Informed Neural Networks (PINNs) [18] and a pre-existing segmentation model [22] for the task of glacier mapping by segmenting glacier ice in satellite images. We will investigate ways

to combine datasets containing hyperspectral satellite images that have no velocity information and datasets of ice glacier velocities to leverage the knowledge we have about the physics of ice glacier velocity flows and achieve better performance on the segmentation of glacier ice.

Chapter 2

Physics-Informed LSTM Network For Velocity Prediction of Fluid Flow Simulations

2.1 The Importance of Fluid Flow Velocity Prediction

In fluid flow velocity prediction you model a system that describes some physical geometry and specific fluids with some initial boundary conditions and predict how these fluids will flow and interact in the system and what the fluid velocities will be at every point in the system after a certain amount of time has passed given the initial boundary conditions. This problem is important in mechanical engineering as the mechanical designs of airplane wings, turbines, and other important machinery are based on how they will interact with the flow of air, water, and other particles in the atmosphere or environment where they are used.

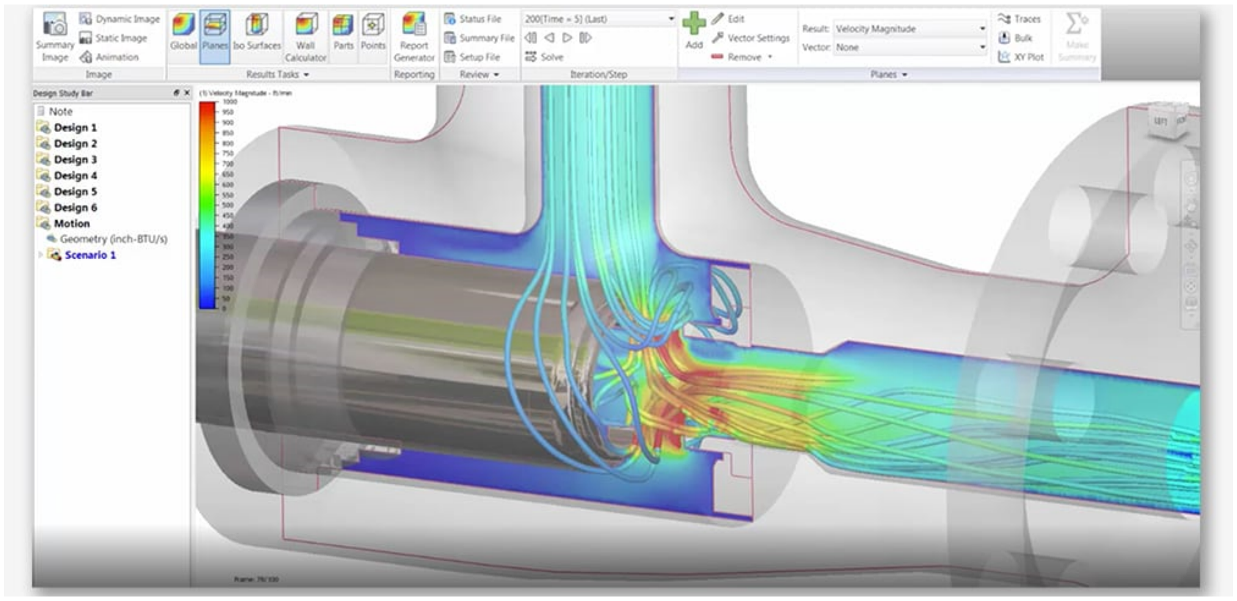


Figure 2.1: An example of a Computational Fluid Dynamics (CFD) simulation predicting the fluid flow velocities in an engine.

2.2 Long Short-Term Memory Networks (LSTMs) For Time Series Data

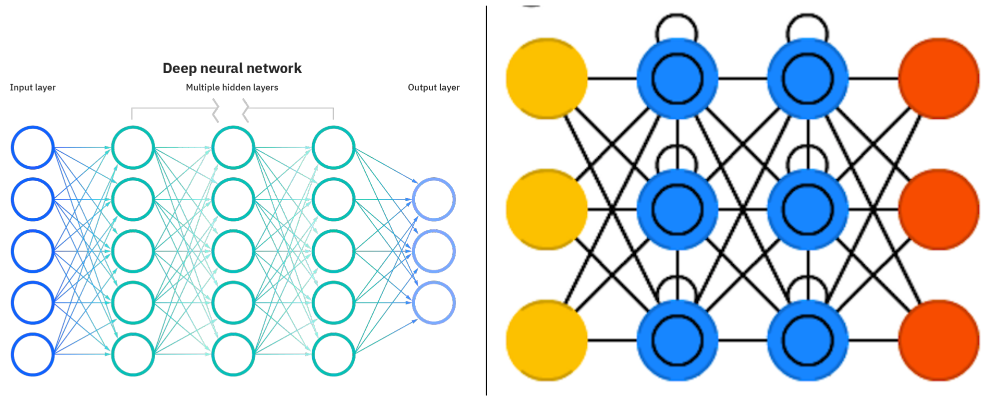


Figure 2.2: A regular 3-layer dense neural network on the left, an LSTM on the right. On the LSTM the blue dots with circles are the specialized LSTM cells, and note how self-connections from RNNs are present in LSTMs as well.

Long Short-Term Memory Networks (LSTMs) [21] are neural network models designed to learn long term dependencies in sequential datasets and are a variation of the Recurrent Neural Network (RNN). LSTMs selectively pick and store short-term information that might be useful to know for later using a unique cell structure that includes an input, forget, and output gate. These gates determine what past information to forget and what new information to keep track of and it is through this mechanism of storing short memories for a long period of time where "long" short-term memory derives the name from.

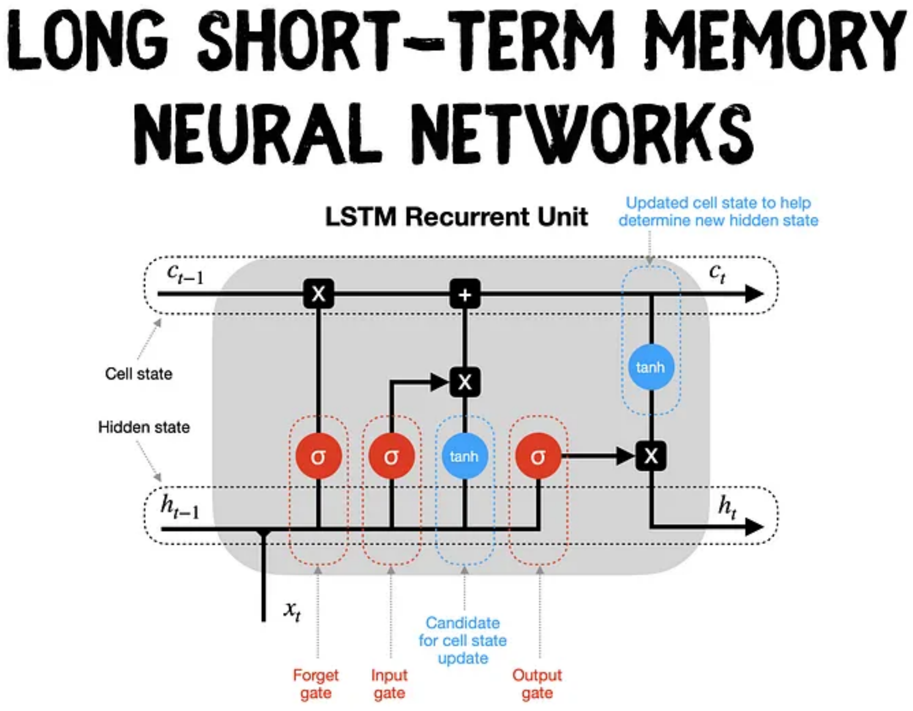

LONG SHORT-TERM MEMORY NEURAL NETWORKS
LSTM Recurrent Unit
Updated cell state to help determine new hidden state
c_{t-1}
c_t
X
+
X
tanh
Cell state
Hidden state
σ
σ
tanh
σ
X
h_{t-1}
h_t
x_t
Candidate for cell state update
Forget gate
Input gate
Output gate

Figure 2.3: An LSTM cell or unit showing all the gates and operations.

2.3 Adding Physics to LSTMs

I began to tackle the problem of fluid flow predictions by creating a new neural network architecture based on a combination of Physics-Informed Neural Networks (PINNs) and

Long-Short Term Memory Networks (LSTMs) [24]. Specifically, we created a two-branch network architecture that takes as inputs the measurements taken from a Computational Fluid Dynamic (CFD) simulation and predicts what the velocity components (in 2D) will be for a given timestep.

2.3.1 Fluid Flow Velocity Data From Simulations

The data was generated using Ansys by creating a simulation in which we modeled a water-braking scoop mechanism in 2D space, constraining the domain of the model to a 2.2 meter by 4 meter box and running the simulation with different inlet velocity profiles. The simulation geometry, mesh, and boundary conditions can be seen in Figures 1.2, 1.1, and 1.3 in the Introduction section.

The following figures demonstrate the simulation results for different boundary conditions at specific times in the simulation. For the figures with two pictures, the left picture represents the pressure and velocity contours and the right picture the velocity and water volume fraction.

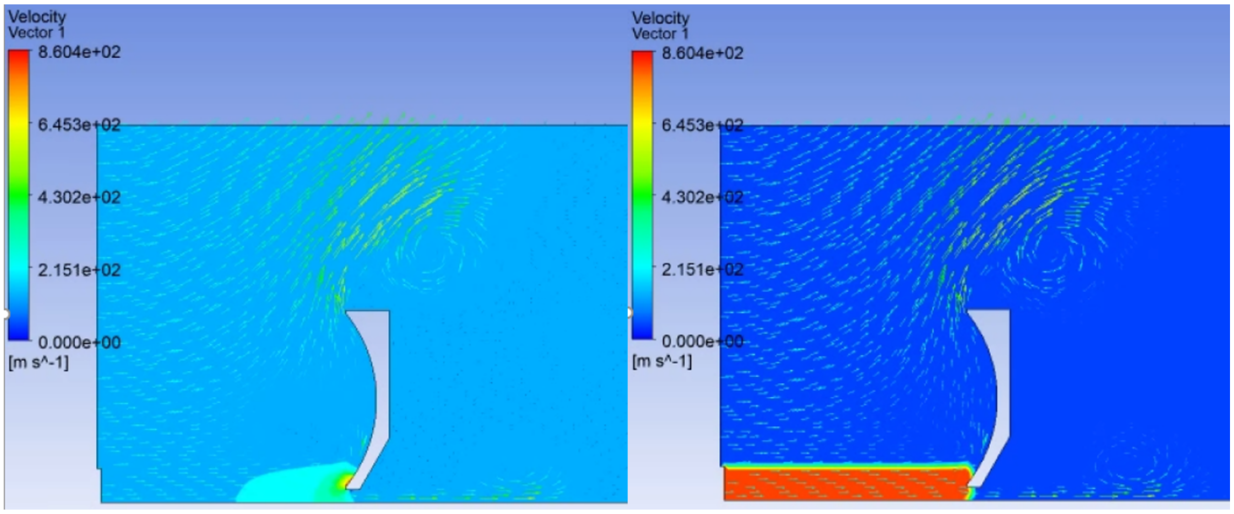


Figure 2.4: Velocity Inlet = $250 m/s$, Time = $0.011 sec$

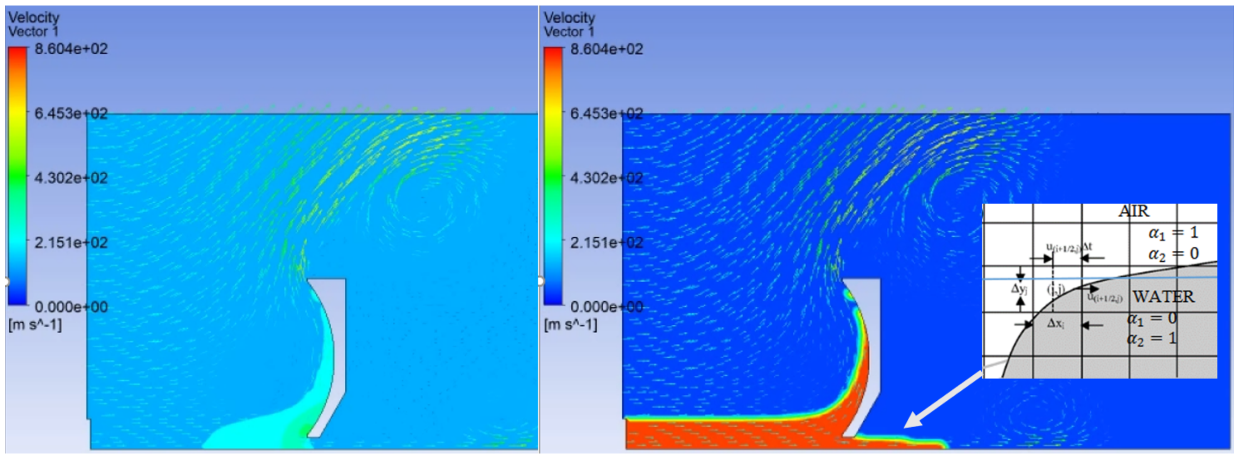


Figure 2.5: Velocity Inlet = $250m/s$, Time = $0.016sec$.

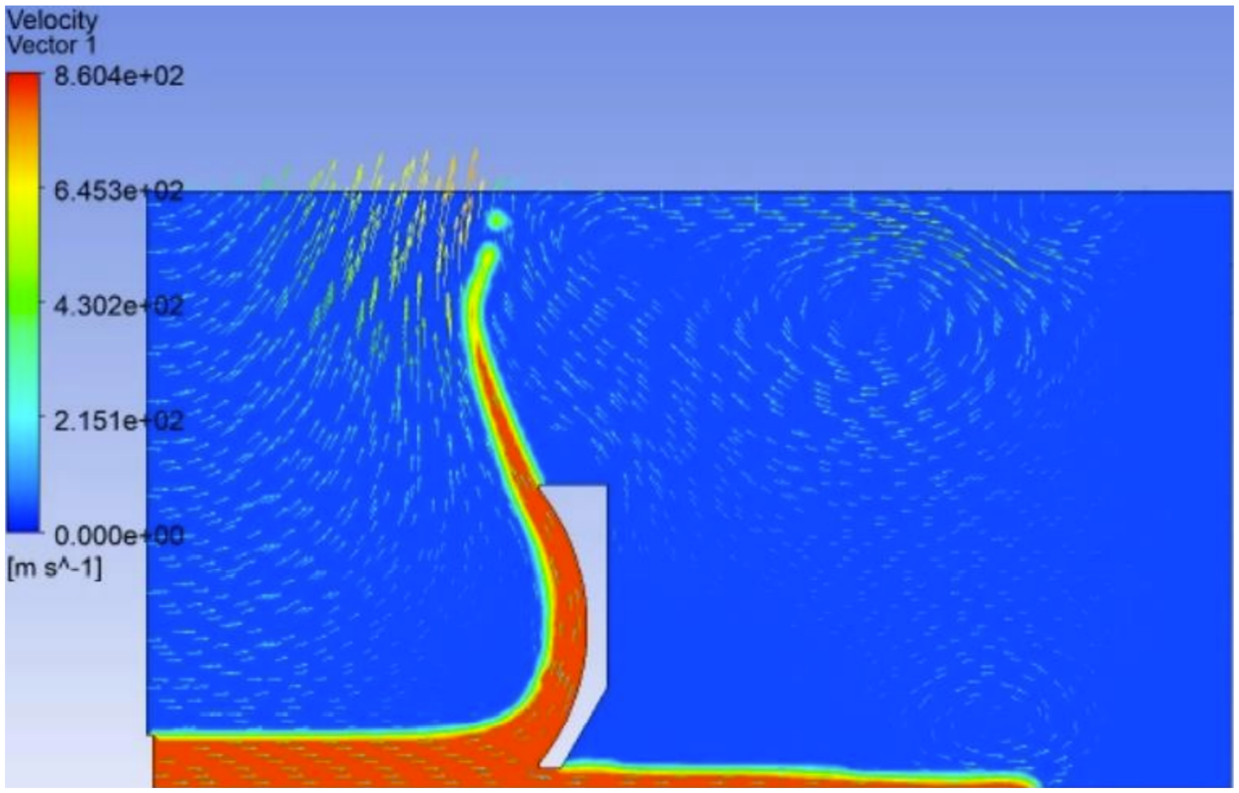


Figure 2.6: Velocity Inlet = $250m/s$, Time = $0.025sec$

385 The 7 measurements stored for each data sample in the simulation were:

386 • The x-coordinate of the node, x

- The y-coordinate of the node, y
- The timestep, t
- The pressure (in atmospheres), p
- The u-velocity representing the x-component of the velocity field, u
- The v-velocity representing the y-component of the velocity field, v
- The water volume fraction or percentage of water from 0 to 1, $w.vf$

From now on, we will refer to data samples with the letter x, so for example x^p is the pressure of the sample x and x_t^p is the pressure of the sample x at a specific point in time t.

2.3.2 Measuring Network Performance With The Mean Squared Error

To measure the performance of our neural network we used a metric called **Mean Squared Error** which computes how erroneous our predictions were on average from the real values. The metric is defined in Equation 2.1 below.

$$\frac{1}{n}\sum_{i=1}^{n}(y_{pred} - y_{true})^2 \tag{2.1}$$

where y_{true} was the real value, y_{pred} was our predicted value, and n was the number of values or samples we were comparing.

For perfect predictions the value of $MSE = 0$. Therefore, the closer to 0 the better a model is performing.

2.3.3 Proposed Network Architecture

The proposed two-branch neural network took a range of time of measurements as input called the "lookback" and outputted the velocity components u and v that were predicted for the same position in space for the next timestep. The LSTM branch incorporated knowledge about the sequence and the Physics-Informed branch incorporated knowledge about the Partial Differential Equations (PDEs) describing in-compressible 2D discrete Navier-Stokes fluid flows. The architecture looked as follows:

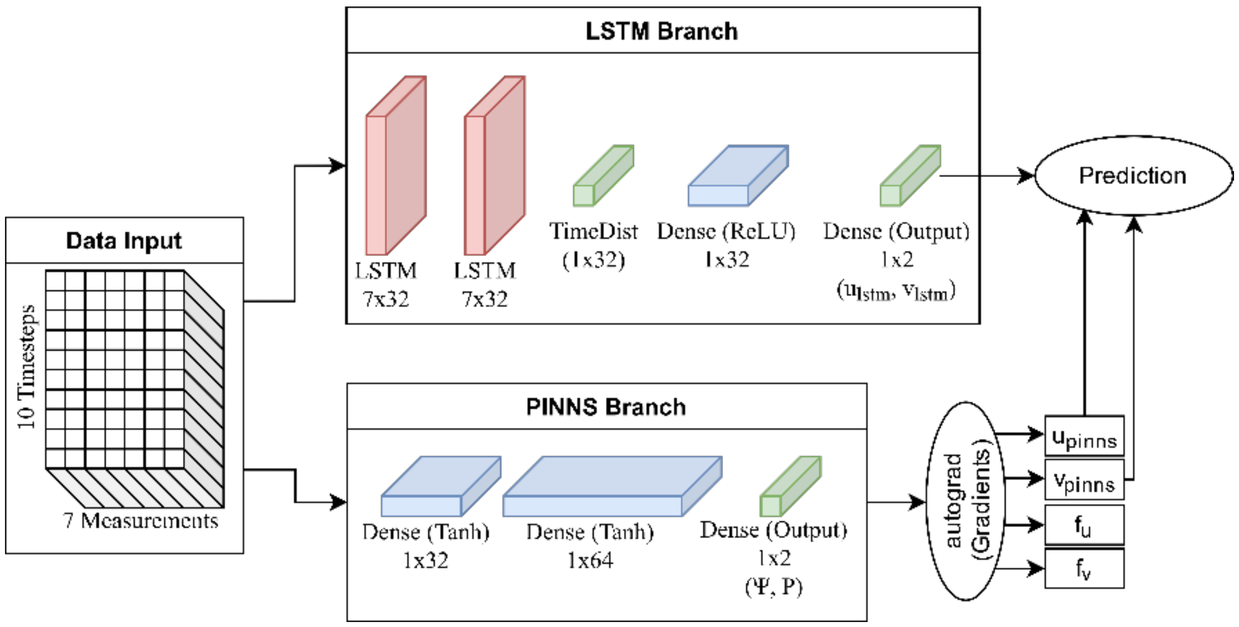


Figure 2.7: Physics-Informed LSTM Architecture showing both the LSTM and Physics-Informed branches and their connection to the final output prediction.

LSTM Branch

The LSTM branch had two stacked bi-directional LSTM layers with 32 activations each, followed by a Time Distributed layer that used a fully connected layer of 32 activations with a linear activation function. The Time Distributed layer applied the same weights to the previous LSTM outputs one timestep at a time. The LSTM layers were followed by a fully connected layer with 32 activations using the ReLU [25] non-linear activation function which was then followed by a fully connected output layer. The output layer consisted of two outputs, the x-component of the velocity field called u which was represented as u_{lstm} and the y-component of the velocity field called v which was represented as v_{lstm}. The LSTM layers were connected to the fully connected part of the network by concatenating the two directional hidden states outputted by the last LSTM layer.

Physics-Informed Branch

As described in [18], the general 2D Navier-Stokes equations:

$$\begin{aligned} u_t + p_x + \lambda_1(uu_x + vu_y) - \lambda_2(u_{xx} + u_{yy}) &= 0 \\ v_t + p_y + \lambda_1(uv_x + vv_y) - \lambda_2(v_{xx} + v_{yy}) &= 0 \\ u_x + v_y &= 0 \end{aligned} \tag{2.2}$$

We assumed that for some latent function $\psi(x, y, t)$

$$\begin{aligned} u &= \psi_y \\ v &= -\psi_x \end{aligned} \tag{2.3}$$

Then we approximated $\psi(x, y, t)$ using a dense neural network f with three inputs (x, y, t), two outputs (u, v), and two learnable parameters (λ_1, λ_2).

$$\begin{aligned} f_u(x, y, t) &:= u_t + p_x + \lambda_1(uu_x + vu_y) - \lambda_2(u_{xx} + u_{yy}) \\ f_v(x, y, t) &:= v_t + p_y + \lambda_1(uv_x + vv_y) - \lambda_2(v_{xx} + v_{yy}) \end{aligned} \tag{2.4}$$

The combined physics loss we needed to minimize was then computed from the following four losses:

$$\begin{aligned} MSE_u &= \frac{1}{n}\sum_{i=1}^{n}(\psi_y - u^i)^2 \\ MSE_v &= \frac{1}{n}\sum_{i=1}^{n}(-\psi_x - v^i)^2 \\ MSE_{f_u} &= \frac{1}{n}\sum_{i=1}^{n}(f_u(x^i, y^i, t^i))^2 \\ MSE_{f_v} &= \frac{1}{n}\sum_{i=1}^{n}(f_v(x^i, y^i, t^i))^2 \end{aligned} \tag{2.5}$$

Such that the total loss was:

$$MSE_{total} = MSE_u + MSE_v + MSE_{f_u} + MSE_{f_v} \tag{2.6}$$

The architecture consisted of a simple dense network with 3 layers. The first layer had 32 activations and used the hyperbolic tangent non-linear activation function (tanh), followed by another fully connected layer with 64 activations using tanh once again, followed by the output layer with two outputs. The automatic differentiation system included in PyTorch named **autograd** was used to compute the partial derivatives required for the computation of the final physics loss.

Two-Branch Combined Model

The two outputs of each branch were combined by using a weight α that was set between 0 and 1. For our experiments, $\alpha = 0.5$ (but in the future it could be set as a learnable parameter)

$$\begin{aligned} u_{pred} &= \alpha * u_{lstm} + (1 - \alpha) * u_{pinns} \\ v_{pred} &= \alpha * v_{lstm} + (1 - \alpha) * v_{pinns} \end{aligned} \tag{2.7}$$

where (u_{pred}, v_{pred}) were the final predictions, (u_{lstm}, v_{lstm}) were the two outputs of the LSTM branch, and (u_{pinns}, v_{pinns}) were the two outputs of the Physics-Informed branch.

2.3.4 Baseline Used To Compare Against Our Proposed Network

The baseline used for this model was to simply assume that the velocities at the next timestep will be the same as those in the previous timestep for each data point.

$$\begin{aligned} prev_u &= x_t^u \\ prev_v &= x_t^v \\ y_{t+1}^{pred} &= (prev_u, prev_t) \end{aligned} \tag{2.8}$$

where t was the current timestep, x_t was the current data sample, x_t^u was the u-velocity component of x_t, y_t^v was the v-velocity component of x_t, and y_{t+1}^{pred} was our predicted velocity.

2.3.5 Experimental Results

The following table summarizes the results of the experiments and provides the mean squared error for the velocity components **U** and **V** for each model along with the total training time after 200 epochs. The experiments were performed using an NVIDIA RTX 3090 and every epoch took around 27 seconds for non-informed LSTM models and 30 seconds for physics-informed LSTM models.

Model	**U**	**V**	**Time (min)**
Baseline	5.455	2.561	0.061
PINNS Only	5709	12038	80
LSTM Only	1.7811	8.6962	117
PINNS+LSTM	**0.4677**	**1.2794**	142

Table 2.1: Comparison of the performance (MSE) of different models when using the 250m/s simulation for training and the 300m/s simulation for testing.

2.3.6 Conclusion

Our experimental results in Table 2.1 show that the Physics-Informed LSTM outperforms the non-informed LSTM approach and leads us to our conclusion that informing the LSTM about the governing physics leads to better performance than just using LSTMs or PINNs by themselves.

Chapter 3

Glacial Ice Segmentation of the HKH Region With Physics-Informed Data Augmentation

3.1 The Importance of Glacial Ice Segmentation

Glacial ice segmentation refers to the problem of gathering hyperspectral images taken by satellites of different glaciers and segmenting or delineating which areas are glacial ice, which areas are debris covered ice (ice mixed with rocks), and which areas are just rocks. This problem is important in the field of geology as monitoring the amount and location of ice from glaciers such as the ones in the Hindu Kush Himalayas (HKH) is critical as these glaciers provide a source of freshwater to a big population of the world.

3.2 Image Semantic Segmentation: Grouping Similar Pixels Together

The task of semantic segmentation refers to grouping similar or related pixels of an image together, like those of a specific object. For example, grouping all the pixels of people, roads, buildings, or trees as seen in the figure below:

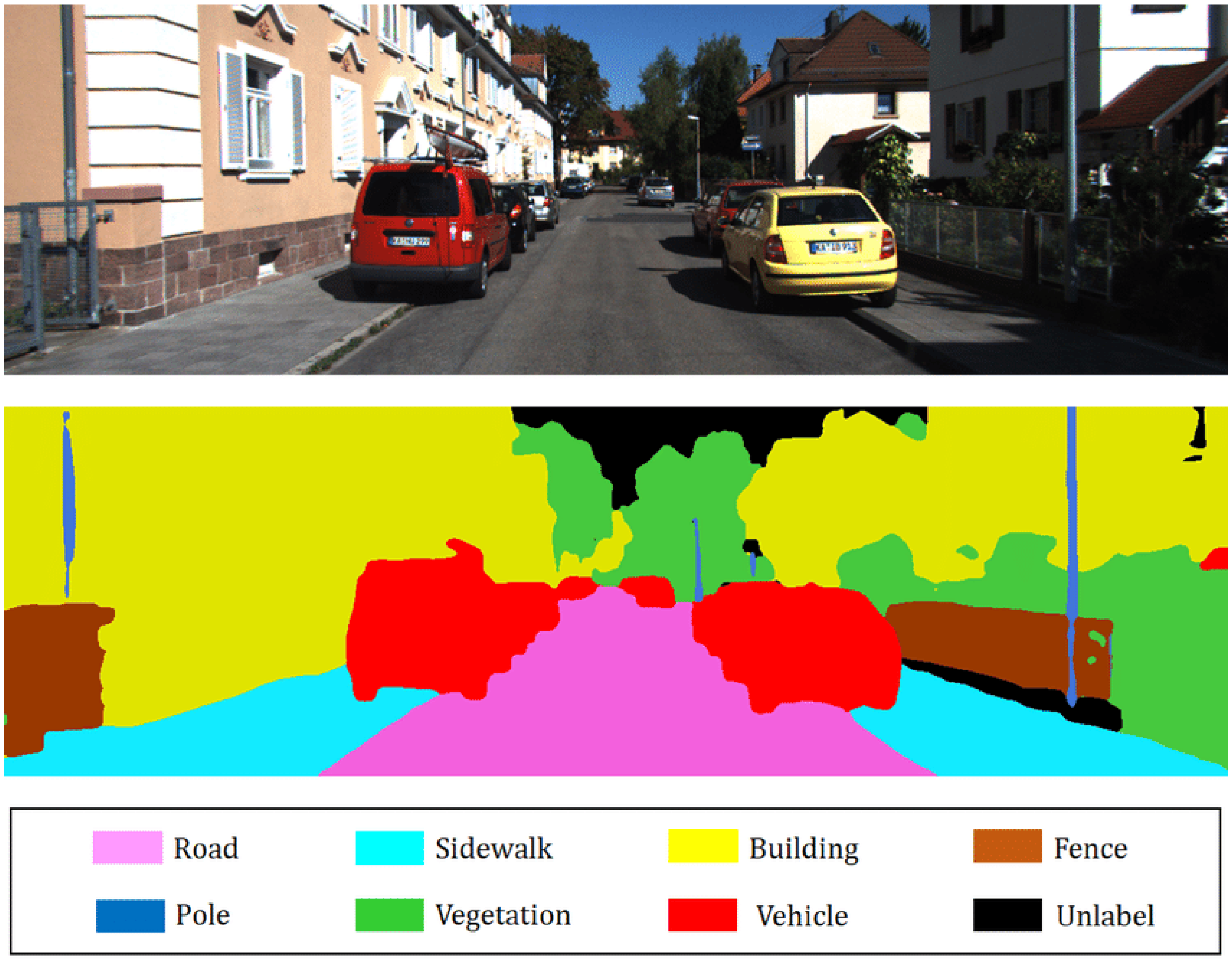


Figure 3.1: An example of semantic segmentation of an image. The pixels are grouped together by different categories.

3.3 Glacier Mapping Through Segmentation of Ice in Hyperspectral Images

Glacial ice segmentation or glacier mapping is simply the task of semantic segmentation applied to hyperspectral satellite images of glaciers. The goal is to determine for each pixel in the image whether the pixel is an area of glacial ice, debris covered ice, or background (regular rocks) as shown in the figure below:

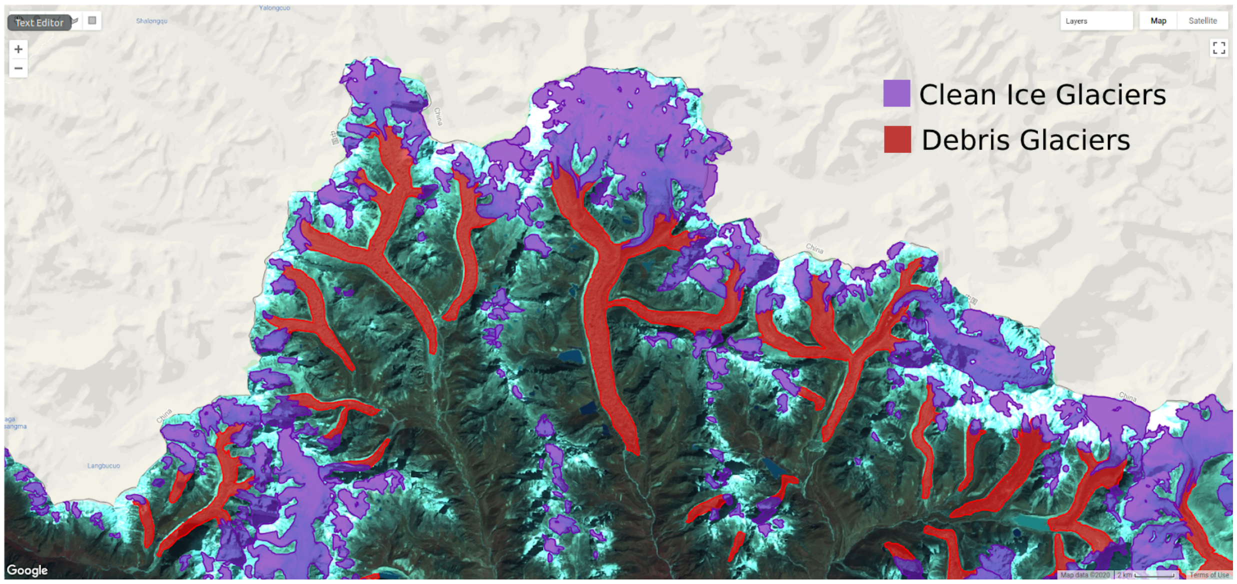


Figure 3.2: Semantic segmentation of a glacier satellite image by a human annotator.

3.4 Convolutional Neural Networks: The Backbone of Almost All Networks That Use Images

Convolutional Neural Networks [26] are one of the most important and fundamental neural network models when dealing with datasets that contain images and their role in state-of-the-art architectures in image classification competitions such as ImageNet and other image tasks cannot be understated. CNNs operate by assuming the inputs are images and performing operations on the pixels of these images called convolutions. The purpose of these convolution operations is to apply a ‘filter‘ to extract important features out of these images as feature maps, and because these CNNs are neural networks these ‘filters‘ or ‘kernels‘ automatically learn to filter out information which might be useful for the specific task we are trying to solve. After the convolution operations a pooling operation is applied to the extracted feature maps which reduces size of these maps and the computational complexity of the network. A common pooling operation is ‘2x2 max-pooling‘ where the image is split into 2x2 patches and the maximum value at each of these patches is used to create the new reduced size feature map, the idea being that the most prominent features or those with high values are the most important for solving image problems. After extracting important features of images through convolutions and pooling layers the next step in CNNs is to convert the 2D feature maps into 1D vectors by flattening them out and then feeding these feature vectors as inputs to a regular dense network to get a final output. The layers of this dense network are typically called ‘fully-connected layers‘ to differentiate them from the ‘convolutional layers‘.

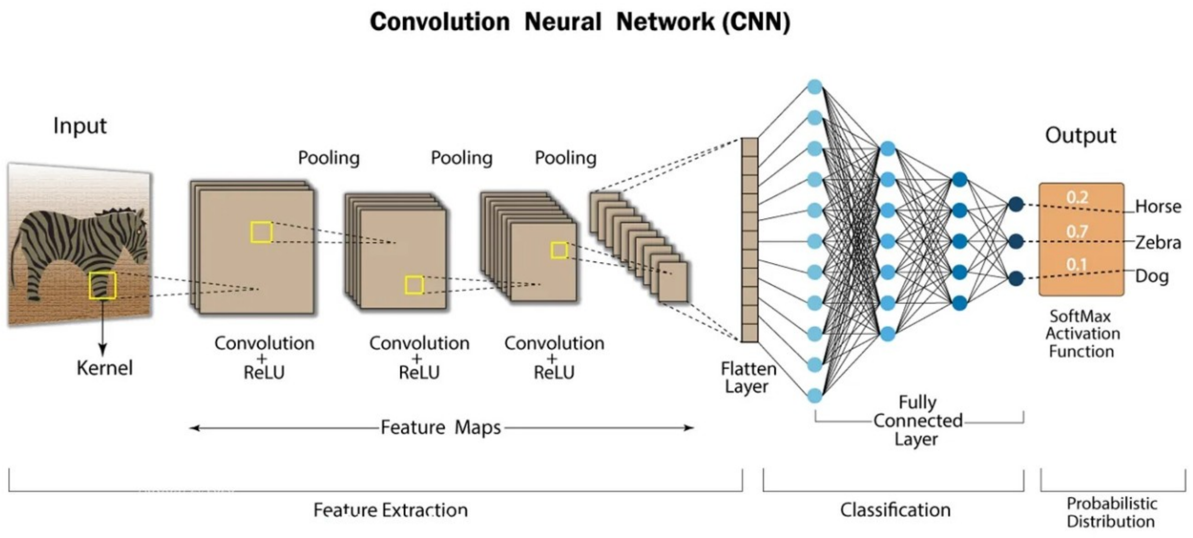


Figure 3.3: An example of a CNN model for an animal classification problem. Features are extracted with convolutions and pooling, and then used in a fully connected network for a final classification.

3.5 UNet: The Standard Network For Image Segmentation

U-Net [27] is a variation of a CNN and was derived from the improvements achieved by Fully Connected Convolutional Networks (FCNs) for semantic segmentation [28]. FCNs transform the fully-connected layers of a CNN into convolutional layers and through an upsampling strategy called ‘deconvolution‘ or ‘transposed convolution‘ output an image that is the same size as the original image. U-Net is split into two components, and encoder in the left half of the U-shape and a decoder in the right half.

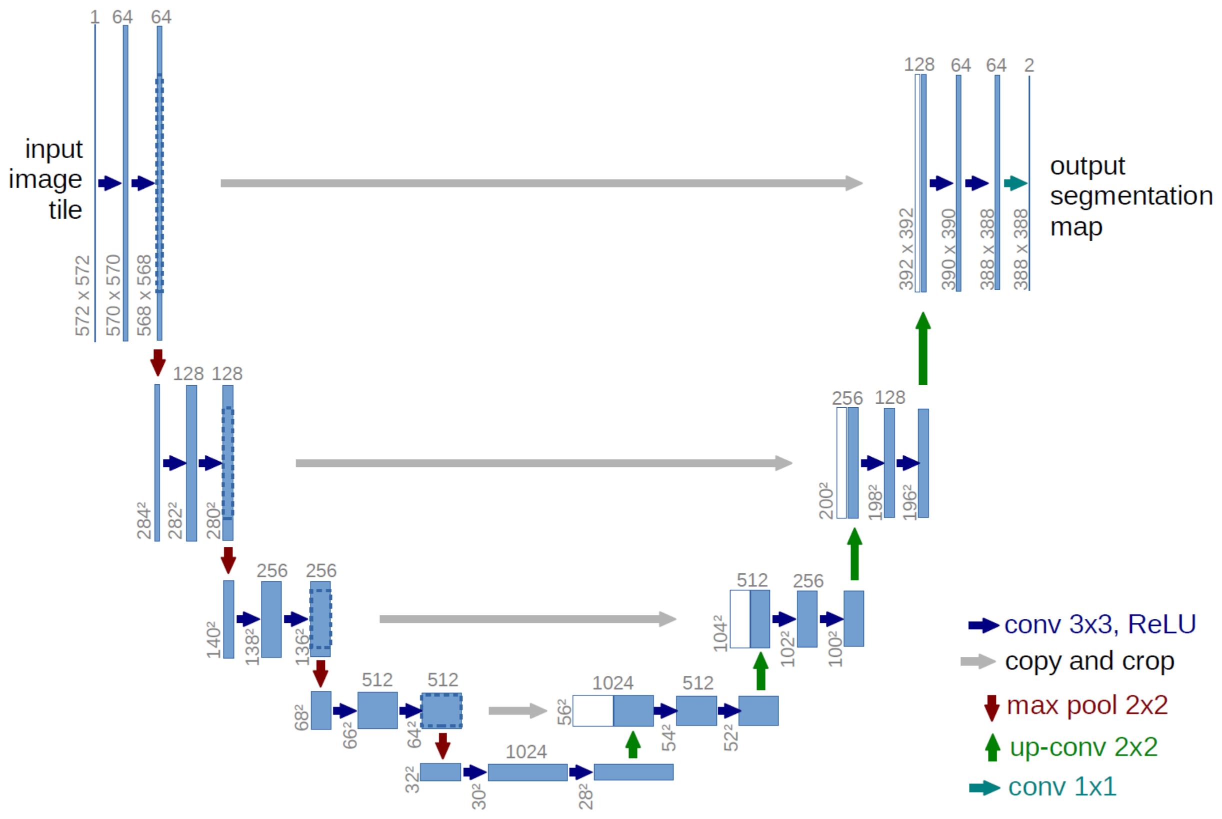


Figure 3.4: U-Net architecture showing both components and how they are connected to output a final segmentation map. Note the straight connections feeding previous inputs into later outputs going straight across.

The encoder also known as the contracting path is the left half of the U-shape that takes the original input image, applies regular convolutions and max-pooling, and outputs a reduced size feature map at the bottom. This is the feature extraction component that aims to capture the important context of the image.

The decoder also known as the expanding path is the right half of the U-shape that takes the feature map from the encoder and through ‘transposed convolutions‘ or ‘up-convolutions‘ upsamples and expands the feature map all the way until we get an output segmentation map which is the same size as the original input image.

Lastly, to help the decoder maintain some of the location information that is lost while encoding there are ‘skip-connections‘ going from the encoder to the decoder straight across the U-shape.

U-Net achieved state-of-the-art results when introduced in 2015 for biomedical image segmentation and these types of networks are still used to this day for many segmentation problems as a starting baseline.

3.5.1 Measuring Network Performance With IoU

To measure the performance of the segmentation model, the main metric we used was the **Intersection over Union (IoU)** which is defined in Figure 3.5 below and has numbers in the range from 0 to 1 with 1 being a perfect score.

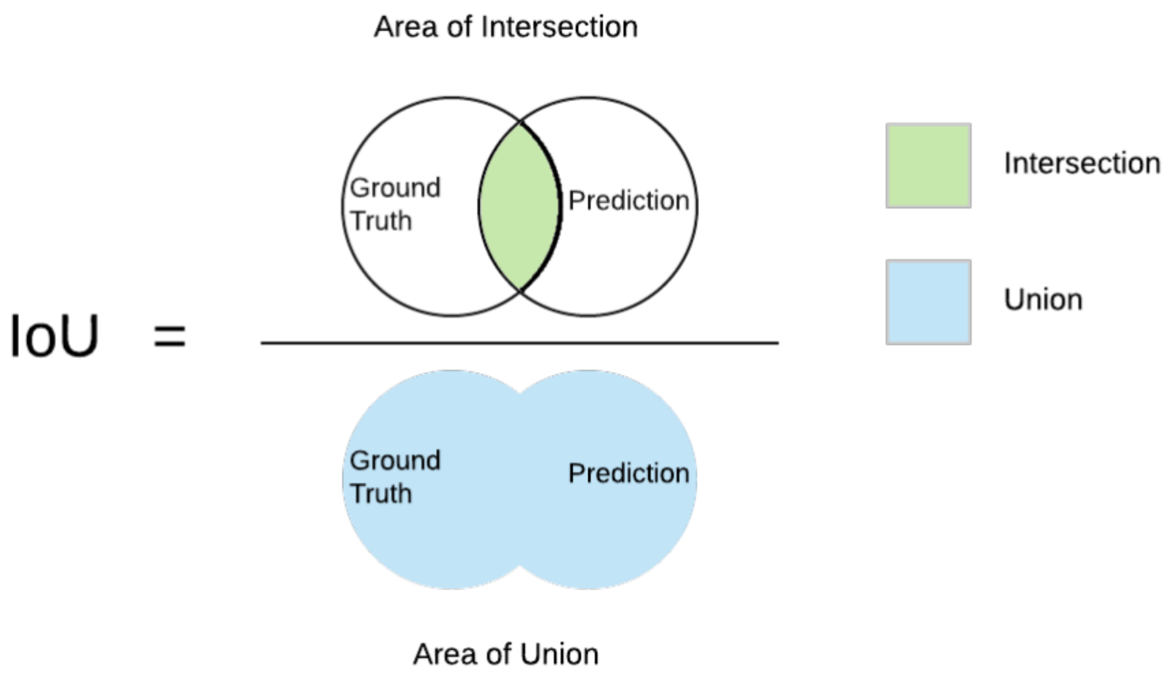


Figure 3.5: Definition of Intersection over Union.

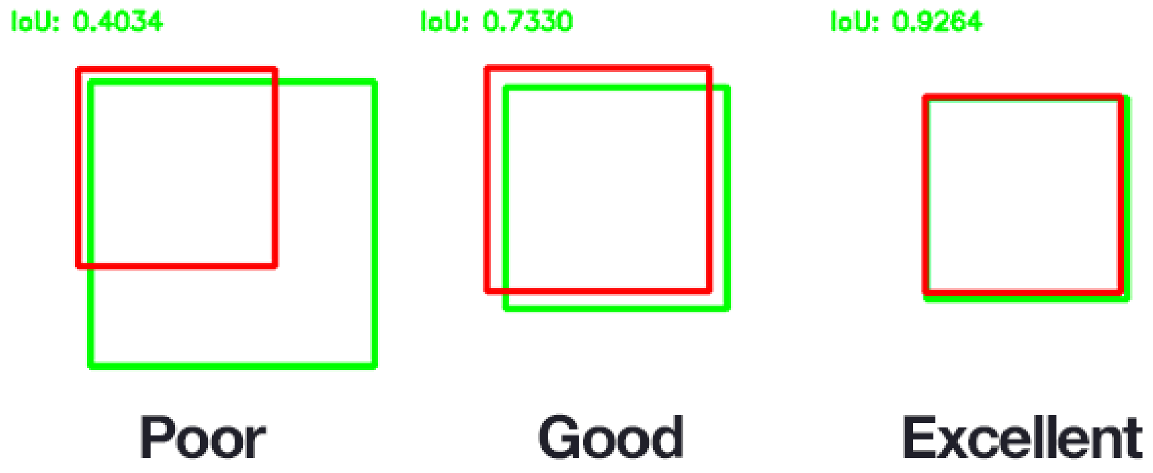


Figure 3.6: Three examples of different IoU values and what type of performance they represent.

3.6 Adding Physics To Image Segmentation Through Data Augmentation

As a first step towards tackling the problem of glacial ice segmentation I began by taking a previously proposed architecture for segmentation of the Hindu Kush Himalayas (HKH) region glaciers [22] and improving its performance by adding physics to the model through data augmentation.

The main idea was to take the elevation map from the satellite images and encode an abstract representation of a "precipitation model" from that elevation map as a new channel of the image. Although the actual physics equations of ice flow were not explicitly encoded in this new data channel, this new data was created from an abstract representation of the physics and we therefore called it a physics-informed data augmentation. In this channel we simulated precipitation and encoded paths where water or ice might flow from the top to the valley of the glaciers.

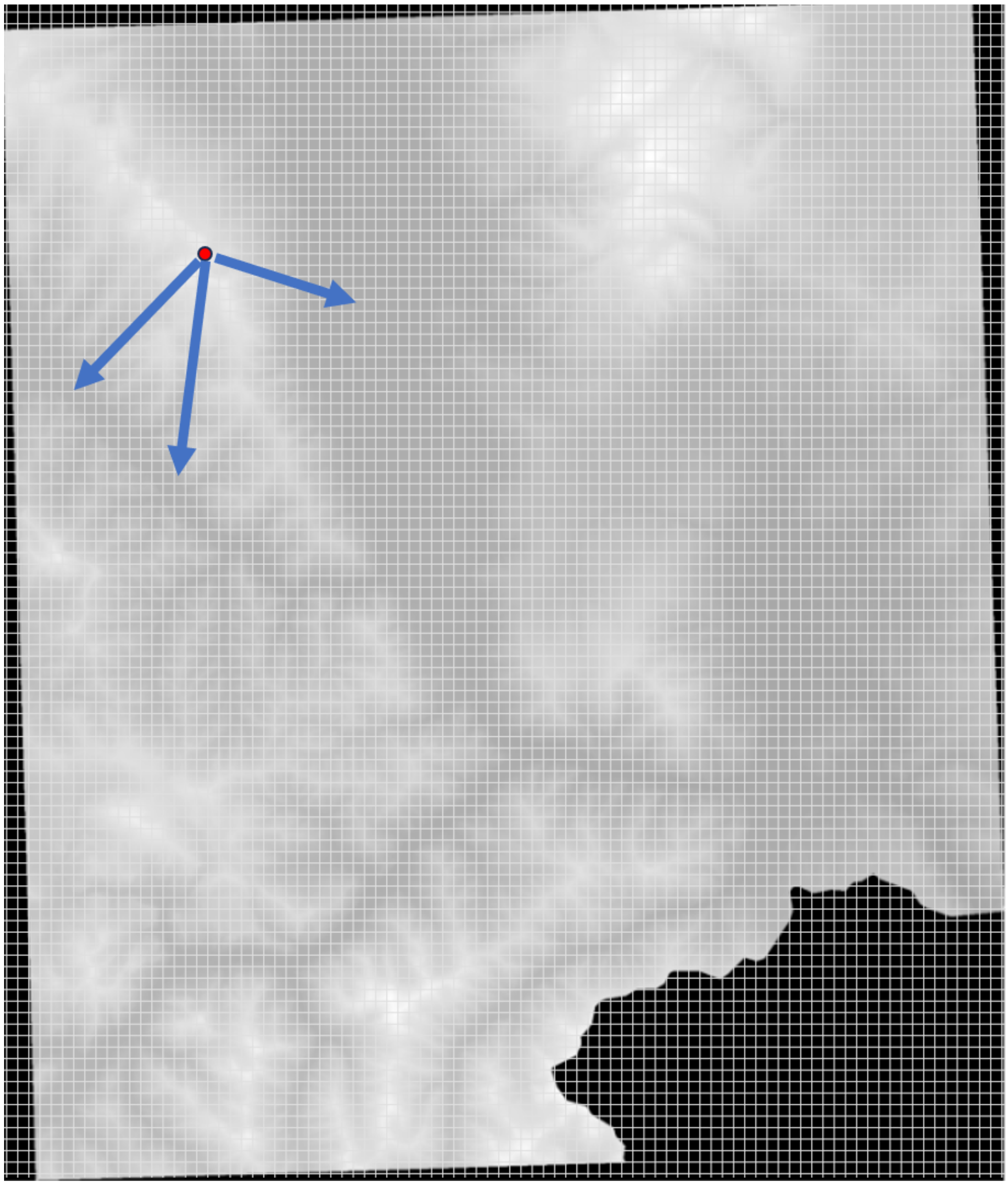


Figure 3.7: For each pixel in the image we simulated precipitation going down the glacier and accumulating, then we encoded such information as an additional physics-informed channel.

Specifically, the algorithm as follows:

1. Take the elevation map as the input image *elevation*
2. Create an empty image of the same size as the input called *output*
3. For each pixel in the image, perform Breadth-First Search (BFS) with a special limitation where you can only visit pixels you haven't visited before AND that are lower elevation than the current pixel popped from the queue as water/ice can only flow down.
4. Each time a pixel is visited, the *output* at that pixel's position increases by 1 as 1 drop of water/ice has flowed down to it.
5. At the end, normalize the image between 0 and 1 by subtracting the mean and dividing by the standard deviation.

In pseudocode, the data augmentation algorithm would look like:

Algorithm 1 Physics-Informed Data Augmentation Algorithm

```
$im \leftarrow$ elevation map from satellite image
$output \leftarrow$ image full of zeros the same size as $im$
$im.shape[0] \leftarrow$ number of rows in $im$
$im.shape[1] \leftarrow$ number of columns in $im$
for $u := 0 \rightarrow im.shape[0]$ do
    for $v := 0 \rightarrow im.shape[1]$ do
        Breadth_First_Search(im, output, u, v)
    end for
end for
$im = (im - im.mean())/im.std()$
```

With the modified BFS algorithm being:

Algorithm 2 Physics-Informed Breadth-First Search

```
$im \leftarrow$ elevation map from satellite image
$output \leftarrow$ previous accumulated output from other pixels
$u \leftarrow$ row of source pixel
$v \leftarrow$ column of source pixel
$source \leftarrow (u, v)$
$visited = source$
$Q =$ Queue with $source$ appended to it
**while** $Q \neq Empty$ **do**
    $x = Q.pop()$
    $curr\_elevation \leftarrow im[x]$
    **if** $x \neq source$ **then**
        $output[x] += 1$
    **end if**
    **for** $n$ in $get\_neighbors(im, x)$ **do**
        $n\_elevation \leftarrow im[n]$
        **if** $n$ not visited & $n\_elevation < curr\_elevation$ **then**
            Add $n$ to $visited$
            Append $n$ to $Q$
        **end if**
    **end for**
**end while**
```

The following figure presents an example of the results after performing the augmentation.

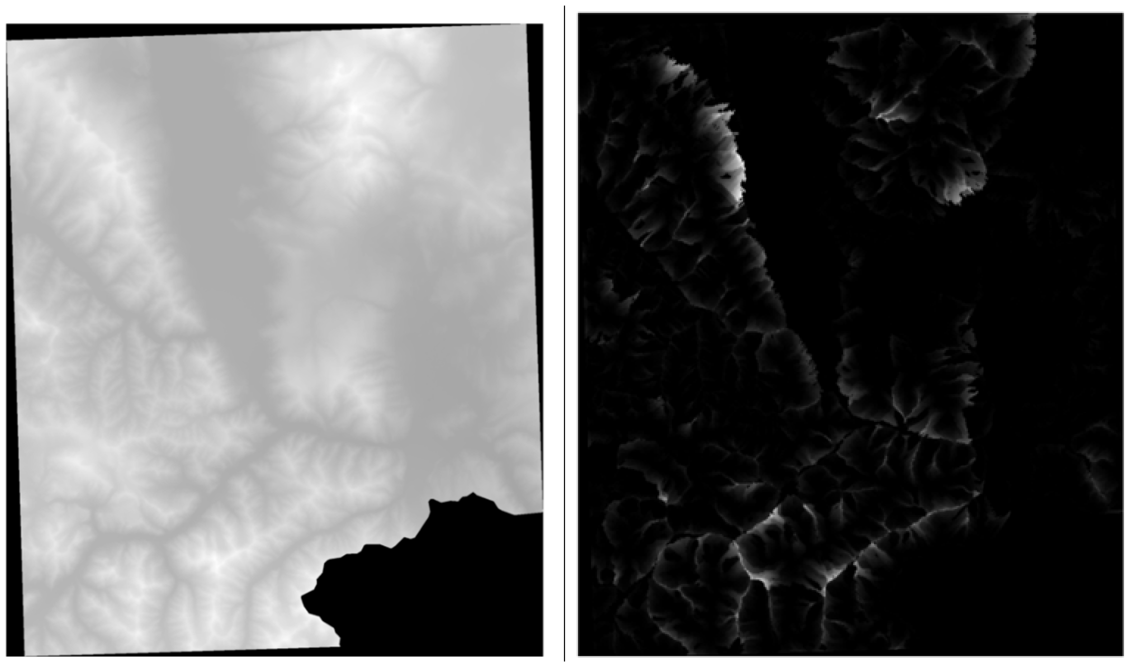


Figure 3.8: Left is the elevation map from one of the glacier satellite images where white pixels are peaks and darker pixels are valleys. Right is the resulting physics-informed data augmentation channel.

3.6.1 Glacier Mapping Data From ICIMOD And Landsat7 Satellite

The data used to train and evaluate the models were gathered and labelled by experts at the International Centre for Integrated Mountain Development (ICIMOD) from multispectral imagery from NASA's Landsat 7 satellite for the glaciers of the Hindu Kush Himalayas region from 2002 to 2008.

3.6.2 Baseline Network

The baseline model used to evaluate our proposed data augmentation technique was the Boundary-Aware U-Net for Glacier Segmentation model developed and published in 2023 by Bibek et. al [22]. No modifications to the model were made apart from slight changes to the hyperparameter configuration and the addition of the physics-informed data augmentation technique.

3.6.3 Experimental Results

Model	Background IoU	CleanIce IoU	DebrisIce IoU
Regular U-Net	N/A	N/A	0.2850
Regular U-Net	N/A	0.6560	N/A
Boundary-Aware U-Net (Baseline)	N/A	N/A	0.3594
Boundary-Aware U-Net (Baseline)	N/A	**0.6817**	N/A
Physics & Boundary-Aware U-Net (mine)	0.8640	0.6350	**0.3850**

Table 3.1: Comparison of the performance of different neural network models by the Intersection Over the Union (IoU) between the predicted labels and the true labels for each class. IoU is a metric between 0 and 1, with 1 being a perfect score.

3.6.4 Conclusion

The current experimental results demonstrate that my proposed physics-informed data augmentation leads to improved performance for the segmentation of Debris-covered Ice from the baseline model. This is significant as classification of Debris-covered Ice is a more challenging task than that of Clean Ice for both the baseline model and expert glaciologists. With some hyper-parameter tuning I hypothesize that we can reach up to 0.40 IoU, and by switching U-Net to a newer state-of-the-art segmentation model such as MANet [29] along with using newer loss functions such as the Unified Focal Loss [30] I believe we might be able to reach above 0.50 IoU. Although my proposed approach does not outperform the baseline for Clean Ice segmentation, I believe this is due to the fact that the original baseline uses two separate binary models for segmentation of each class instead of one unified multi-class model. I will train two separate binary models for a more fair comparison in the final version of my full dissertation.

Chapter 4

Physics-Informed Network For Glacial Ice Velocity Predictions

As my next step, I propose making a new Physics-Informed architecture based on what I learned from the Physics-Informed LSTM Network [24] which I previously applied to fluid flow simulations and adapting it to the problem of glacial ice velocity prediction. Glacial ice velocity prediction is simply the problem of fluid flow velocity prediction where the fluid is ice from a glacier. The aim of this step is to prepare a network architecture and dataset than can be later incorporated into the glacier segmentation problem as I hypothesize that including velocity information based on the physics laws of fluid flow will improve the performance of the glacier segmentation models proposed by [22]. There already exists a dataset of glacier ice velocities [31] for satellite images created by the National Snow and Ice Data Center which can be easily adapted and incorporated with my previous methodology for velocity predictions. However, since then I have learned about newer architectures that perform better than LSTMs [21] such as GRUs [32] and Transformers [33] which I would like to try as well.

The main challenge with this step is determining how to set-up the Physics-Informed part of the network as my previous work focused on air and water flows and not ice flows. The architecture will not require many changes, it is the loss function which was based on the incompressible 2D Navier-Stokes equations which might need to be modified to accommodate the new type of data.

Chapter 5

Physics-Informed MANet for Glacial Ice Segmentation of the HKH Region

As my final step, I propose combining the Physics-Informed Network used for velocity prediction and the U-Net used for image segmentation to further improve the performance on the segmentation of ice in the HKH region. I will do this by trying two different methodologies. In the first methodology, I will simply create a two branch network the same way I did with the Physics-Informed + LSTM network where we just have to carefully connect the inputs and outputs of their respective branches correctly. In the second methodology, I will use a self-learning loss that will combine the losses of both networks into one and optimize them both at the same time. The self-learning combined loss will be as follows:

$$L_{Combined} = \alpha * L_{UNET} + (1 - \alpha) * L_{PINNS} \tag{5.1}$$

where $L_{Combined}$ is the new combined loss from both networks, L_{UNET} is the loss of the UNet ice segmentation network and L_{PINNS} is the loss of the Physics-Informed velocity prediction network. This loss is based on the self-learning boundary aware loss specified in [22].

The main challenge will be combining the datasets for the combined network as they were not taken at the same exact time and so the satellite images used for segmentation and the velocity images are slightly different. I do not believe this will present a major problem, as I have the script used to get both datasets so I will be able to control how far apart these datasets are in real time to minimize the error introduced by this problem.

I will also try a newer segmentation architecture that has been shown to be better than UNet called MANet [29] which I hypothesize will give me the best performance in the end.

Lastly, I will develop a simple WebUI that allows glaciologists to feed their own hyperspectral satellite images to our trained models to get labeled masks of glacial ice in a format that can be loaded into QGIS to help them with their glacier mapping efforts.

Bibliography

1. Rombach R, Blattmann A, Lorenz D, Esser P, and Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021:10674–85.

2. Redmon J, Divvala SK, Girshick RB, and Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015:779–88.

3. Tan M and Le QV. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv 2019;abs/1905.11946.

4. Devlin J, Chang MW, Lee K, and Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *North American Chapter of the Association for Computational Linguistics.* 2019. URL: https://api.semanticscholar.org/CorpusID:52967399.

5. Brown TB, Mann B, Ryder N, et al. Language Models are Few-Shot Learners. ArXiv 2020;abs/2005.14165.

6. Amankwa G. Modeling the Spatiotemporal Dynamics of Active Regions on the Sun Using Deep Neural Networks. PhD thesis. University of Texas, El Paso, 2021.

7. Mao J, Shi S, Wang X, and Li H. 3D Object Detection for Autonomous Driving: A Comprehensive Survey. International Journal of Computer Vision 2022;131:1909–63.

8. Tolstikhin IO, Houlsby N, Kolesnikov A, et al. MLP-Mixer: An all-MLP Architecture for Vision. ArXiv 2021;abs/2105.01601.

9. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, and Aila T. Analyzing and Improving the Image Quality of StyleGAN. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2019:8107–16.

10. Deng J, Dong W, Socher R, Li LJ, Li K, and Fei-Fei L. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition 2009:248–55.

11. Lin TY, Maire M, Belongie SJ, et al. Microsoft COCO: Common Objects in Context. In: *European Conference on Computer Vision*. 2014. URL: https://api.semanticscholar.org/CorpusID:14113767.

12. Karras T, Laine S, and Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2018:4396–405.

13. Deng L. The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine 2012;29:141–2.

14. Finn C, Abbeel P, and Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: *International Conference on Machine Learning*. 2017. URL: https://api.semanticscholar.org/CorpusID:6719686.

15. Koch GR. Siamese Neural Networks for One-Shot Image Recognition. In: 2015. URL: https://api.semanticscholar.org/CorpusID:13874643.

16. Snell J, Swersky K, and Zemel RS. Prototypical Networks for Few-shot Learning. In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:309759.

17. Sung F, Yang Y, Zhang L, Xiang T, Torr PHS, and Hospedales TM. Learning to Compare: Relation Network for Few-Shot Learning. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition 2017:1199–208.

18. Raissi M, Perdikaris P, and Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 2019;378:686–707.

19. ANSYS Fluent. Canonsburg, PA.

20. National Snow and Ice Data Center. Why Glaciers Matter — National Snow and Ice Data Center. 2023. URL: `https://nsidc.org/learn/parts-cryosphere/glaciers/why-glaciers-matter`.

21. Hochreiter S and Schmidhuber J. Long Short-Term Memory. Neural Computation 1997;9:1735–80.

22. Aryal B, Miles K, Vargas Zesati S, and Fuentes O. Boundary Aware U-Net for Glacier Segmentation. In: *Proceedings of the Northern Lights Deep Learning Workshop*. Vol. 4. 2023. DOI: `10.7557/18.6789`.

23. Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need. In: *Advances in Neural Information Processing Systems*. Ed. by Guyon I, Luxburg UV, Bengio S, et al. Vol. 30. Curran Associates, Inc., 2017. URL: `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

24. Pérez J, Baez R, Terrazas J, et al. Physics-Informed Long-Short Term Memory Neural Network Performance on Holloman High-Speed Test Track Sled Study. American Society of Mechanical Engineers, Fluids Engineering Division (Publication) FEDSM 2022;2.

25. Fukushima K. Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements. IEEE Trans. Syst. Sci. Cybern. 1969;5:322–33.

26. Krizhevsky A, Sutskever I, and Hinton GE. ImageNet classification with deep convolutional neural networks. Communications of the ACM 2012;60:84–90.

27. Ronneberger O, Fischer P, and Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv 2015;abs/1505.04597.

28. Shelhamer E, Long J, and Darrell T. Fully convolutional networks for semantic segmentation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2014:3431–40.

29. Li R, Zheng S, Duan C, and Su J. Multiattention Network for Semantic Segmentation of Fine-Resolution Remote Sensing Images. IEEE Transactions on Geoscience and Remote Sensing 2020;60:1–13.

30. Yeung M, Sala E, Schönlieb CB, and Rundo L. Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation. Computerized Medical Imaging and Graphics 2022;95:102026.

31. Fahnestock M, Scambos T, Moon T, Gardner A, Haran T, and Klinger M. Rapid large-area mapping of ice flow using Landsat 8. Remote Sensing of Environment 2016;185:84–94.

32. Cho K, Merrienboer B van, Gülçehre Ç, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Conference on Empirical Methods in Natural Language Processing*. 2014. URL: https://api.semanticscholar.org/CorpusID:5590763.

33. Dosovitskiy A, Beyer L, Kolesnikov A, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ArXiv 2020;abs/2010.11929.