StormMind
AI based Web-Service for Storm Damage Prediction

Bachelor Thesis

ZHAW School of Engineering
Institute of Computer Science

Submitted on:
06.06.2025

Authors:
Gämperli Nils
Ueltschi Damian

Supervisor:
Andreas Meier

Study Program:
Computer Science

Declaration of Authorship

We, Damian UELTSCHI, Nils GÄMPERLI, declare that this thesis titled, "StormMind - AI based Web-Service for Storm Damage Prediction" and the work presented in it are our own. We confirm that:

- All external help, aids and the adoption of texts, illustrations and programs from other sources are properly referenced in the work.
- This is also a binding declaration that the present work does not contain any plagiarism, i.e. no parts that have been taken over in part or in full from another's text or work under pretence of one's own authorship or without reference to the source.
- In the event of misconduct of any kind, Sections 39 and 40 (Dishonesty and Procedure in the Event of Dishonesty) of the ZHAW Examination Regulations and the provisions of the Disciplinary Measures of the University Regulations shall come into force.

I have no limitations.
— Thomas Shelby

To our parents...

Abstract

Extreme weather events cause considerable damage to infrastructure, the economy and the environment worldwide. Predicting such damage can help to optimize preventive measures and reduce costs. In this project, a neural network is being developed that analyzes historical weather data and damage reports to predict potential storm damage in Switzerland based on new weather forecasts. Using modern machine learning techniques, relevant patterns are recognized in order to train a predictive model. The model is tested for accuracy and optimized to enable reliable predictions.

Key words: Machine Learning, Neural Network, Storm Damages

Acknowledgements

We would like to thank Andreas Meier for his valuable support and guidance throughout this project. Additionally, we would like to thank to Dr. Katharina Liechti which has provided us with valuable insights about the sciences of weather and storm damages. For providing and collecting data to make this project possible, we also like to thank to the Swiss Federal Institute for Forest, Snow and Landscape Research WSL.

Preface

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Key words:

Contents

List of Abbreviations

LSTM	Long Short Term Memory Neural Network
NN	Neural Network
RNN	Recurrent Neural Network
VGP	Vanishing Gradient Problem

1 Introduction

1.1 Motivation

Why do we do this?

1.2 Work Outline

What do we want to predict etc.

2 Theoretical Background

2.1 Weather Research

2.1.1 Reason for Flooding

- Drought Followed by Heavy Reain
- Snowmelt in Spring

2.1.2 Reason for Landslide

- Slope, Water, Soil Condition

2.2 Machine Learning

Machine Learning has gained increasing popularity in recent years, particularly through advancements in Neural Networks (**NNs**). These developments have significantly expanded the capabilities of automated data-driven modeling across various domains. In this chapter, we focus primarily on **NNs** architectures, as they form the core modeling approach used in this project.

2.2.1 Neural Networks

(**NNs**) are a class of machine learning models inspired by the structure and function of the human brain. In biological systems, neurons are interconnected through synapses, and their strengths change in response to external stimuli—a process that underlies learning. **NNs** mimic this behavior by using computational units, also called neurons, connected by weighted links. These weights are adjusted during training to improve the model's predictions, analogous to synaptic strength adjustments in the brain.

Each artificial neuron receives inputs, scales them using learned weights, applies a non-linear activation function, and forwards the result to subsequent neurons. Through this architecture, an NNs models complex functions by propagating signals from input to output layers. Learning in NNs occurs via exposure to training data consisting of input–output pairs. The network adjusts its weights to reduce the difference between its predictions and the target outputs, thereby minimizing the prediction error.

While the biological analogy is imperfect, it has historically guided the development of neural architectures. More formally, NNss can also be viewed as compositions of simple mathematical units—such as logistic or linear regressors—structured into a computational graph. Their expressive power arises from stacking these units into deeper networks, enabling them to approximate highly non-linear relationships in data. This capacity to learn from examples and generalize to unseen inputs makes NNss a powerful tool in modern machine learning.

Architecture

An NNs trained with backpropagation, which is discussed in Section 2.2.2, can be illustrated as a directed acyclic Graph with inter-connections. It contains a set of neurons distributed in different layers.

- Each neuron has a activation function.
- The first layer, shown on the left side in Figure 2.1, is called the input layer and has no predacessors in the inter-connection graph. Additionally, is their input value the same as their output value.
- The last layer, shown on the right side in Figure 2.1, is called the output layer and have no successors in the inter-connection graph. Their value represents the output of the Network
- All other neurons are grouped in the so called hidden layers. In Figure 2.1 this is represented by the layer in the middle. A neural network can have an arbitrary amount of hidden layers.
- The edges in the inter-connection graph, are so called weights, which represent an arbitrary number in $\mathbb{R}$. These weights are updated during the trianing process.

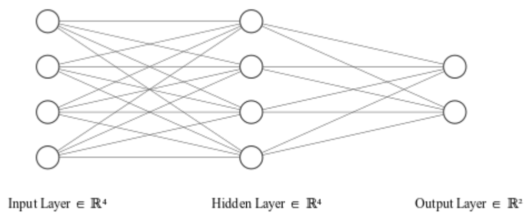

Figure 2.1 – Illustration of a Neural Network with 3 layers. Illustrated with [1]

Computation of the Output

The calculation of the output of the NNs is also called a forward pass. To do so, the value of each neuron needs to be calculated. This is done by summing all the inputs and then put this value into a given activation function. Mathematically, this process can be represented as: $y = f\left(\sum_i^n \xi_i\right)$ wher f represents the activation function of the neuron and ξ_i the input or also called the potential of a neuron. To compute a full forward pass, this is done for each neuron from the input layer towards the output layer. When a value is passed throughtouh a weight to a successor neuron, the value is multiplied by the value of the weight. This process can then be summarized to:

$$\begin{aligned} y_1 &= f\left(\sum_i^n w_{ij} x_i\right) \text{ [input to hidden layer]} \\ y_{j+1} &= f\left(\sum_i^n w_{ij} y_j\right) \forall j \in \{1...k-1\}\text{[hidden to hidden layer]} \\ o &= f\left(w_{ij+1} y_j\right) \text{ [hidden to output layer]} \end{aligned} \tag{2.1}$$

[2] where j denotes the layer, ascending from input layer to output layer, f activation function, w_{ij} weight at index i and layer j, x as input at index i. The state of the neuron in the output layer o can then by denoted as the output vector.

2.2.2 The Backpropagation Training Algorithm

Objective: To identify a set of weights that guarantees that for every input vector, the output vector generated by the network is identical to (or sufficiently close to) the desired output vector.

Note: The actual or desired output values of the hidden neurons are not explicitly specified by the task.

For a fixed and finite training set: The objective function represents the total error between the desired and actual outputs of all the output neurons for all the training patterns.

Error Function

$$E = \frac{1}{2} \sum_p^P \sum_i^N \left(y_{ip} - d_{ip}\right)^2 \tag{2.2}$$

[3]

where P is the number of training patterns, N the number of output neurons, d_{ip} is the desired output for pattern p, y_{ip} the actual output of the neuron i and output neuron i.

Procedure

1. Compute the actual output for the presented pattern
2. Compare the actual output with the desired output
3. Adjustment of weights and thresholds against the gradient of the error function (Equation (2.2)) for each layer from the output layer towards the input layer

Figure 2.2 – Training Procedure of the backpropagation algorithm

Adjustment Rules

$$\begin{aligned} w_{ij}(t+1) &= w_{ij} + \Delta_E w_{ij}(t) \\ \Delta_E w_{ij} &= -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} \end{aligned} \tag{2.3}$$

where $\Delta_E w_{ij}$ denotes the change of the Error Function with respect to w_{ij}, E the Error Function, y_j the output of the output neuron j, ξ_j the potential of the neuron j and w_{ij} the weight with index i at layer j.

2.2.3 Recurrent Neural Networks

The feedforward **NNs** discussed in Section 2.2.1 are inherently limited to fixed-size, unordered input representations. This makes them unsuitable for sequential data, where both the order and length of the input can vary. To address this limitation, we introduce a class of models specifically designed to process variable-length sequences: Recurrent Neural Networks (**RNNs**)

Architecture A **RNNs** consits of the following components:

- Input signal: The external data which is fed into the network at a timestep n and represent the current information which the network is processing.
- State signal: Also known as the hidden state, represents the memory of the **RNNs** for a given neuron. It contains information about the past inputs in the sequence and is updated at each time step based on the current input and the previous state. The hidden state is updated with the following formula: $h_t = f(h_{t-1}, x_t)$. After the update, the hidden state of neuron i serves as input into the neuron $i + 1$
- Weights: The weights of the **RNNs** neurons are shared among all different states.
- Output: Each neuron has a output, which is denoted as y_1 - y_4 in Figure 2.4. This output can serve as the output for the current state or as input into the next neuron.

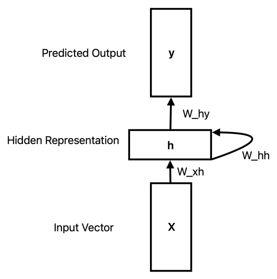

Figure 2.3 – **RNNs**

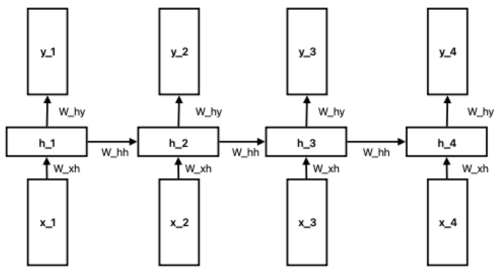

Figure 2.4 – 4 times unrolled **RNNs**

Vanishing Gradient Problem

The Vanishing Gradient Problems (**VGPs**) is a challenge encountered during the training of of **RNNs**, particullary dealing with deep **RNNs** and long input sequences. It arieses from the way how gradients are updated during the backpropagation algorithm (discussed in Section 2.2.2), which updates the network's paramertes / weights by propagating gradients backward through each time step. In backpropagation, gradients are computed via the chain rule, resulting in repeated multiplication of weight matrices and derivatives of activation functions across time steps. When these values are consistently smaller than one, the gradients exponentially decrease as they traverse earlier layers or time steps. Consequently, the gradients become vanishingly small, leading to negligible updates for earlier parameters and impairing the network's ability to learn long-range dependencies. The same principle aries, when the gradients become too large. In this case, the problem is called the exploding gradient problem.

2.2.4 Long Short Term Memory Neural Networks

Long Short Term Memory Neural Networks (**LSTMs**) are a special form of **RNNs** designed to address the problem of Vanishing Gradients while having a more fine-grained control over the previous input data. **LSTMs** are an enhanement of **RNNs**, because the recurrence conditaions of how the hidden state h_t is processed. To achieve this aim, we introduce a new hidden state of the same dimesion as h_t, which is called the cell state and is denoted as c_t. The key innovation of the **LSTMs** lies in its ability to control the flow of information using a set of gating mechanisms. These gates regulate how information is added to, removed from, or exposed from the cell state. Each gate is implemented as a sigmoid-activated neural layer and serves a distinct role in the update process.

Architecture

At each time step t with a given input vector x_t, previous hidden state h_{t-1} and previous cell state c_{t-1}, the **LSTMs** performs the following computations:

- Input Gate: Decides which new information will be added to the cell state and is calculated as: $i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$
- Forget Gate: This gate decides which parts of the previous cell state should be forgotten. The value of the forget gate is calculated as: $f_t = \sigma\big(w_f[h_{t-1}, x_t]\big) + b_f)$
- Candidate Cell State: Computes possible candidates $\tilde{c_t}$ which can be added to the cell state, computed as: $\tilde{c_t} = \tanh(w_c[h_{t-1}, x_t] + b_c)$
- Cell state update: Given the candidates, the cell state can be updated as: $c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c_t}$
- Output Gate: Determines which part of the cell state influences the hidden state and therefore the output. It is computed with: $o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$
- Hidden state update: The final hidden state is computed by applying the output gate to the activated cell state. It is computed with: $h_t = o_t \cdot \tanh(c_t)$

Note: w_i represents a complete weight matrix, b_x denotes the bias for the corresponding gates, and σ denotes the sigmoid function.

[2], [4], [5], [6]

2.3 Software Engineering

3 Methodology

3.1 AI Engineering

3.1.1 Data

3.1.2 Data Cleaning

3.1.3 Deep Learning Model

3.2 Software Engineering

3.2.1 Backend

3.2.1.1 Technologies

3.2.1.2 Architecture

3.2.2 Fronttend

3.2.2.1 Technologies

3.2.3 Test Concept

4 Results

4.1 Prediction Results

4.2 Software Results

5 Discussion and Outlook

Bibliography

[1] "NN SVG." Accessed: May 03, 2025. [Online]. Available: http://alexlenail.me/NN-SVG/°

[2] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-29642-0°.

[3] I. Mrázová, "Multi-Layered Neural Networks."

[4] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306°.

[5] F.-P. Schilling and Z. Cai, "Lecture 05: Sequential Models," Mar. 19, 2025.

[6] D. Thakur, "LSTM and Its Equations." Accessed: May 05, 2025. [Online]. Available: https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af°

List of Figures

List of Tables

A Appendix

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.