Term paper – Numerical models in glaciology

Audun Sørheim

Introduction

The world's glaciers are in constant battle with both weather and climate. A glacier's mass will change due to snowfall and temperature, typically increasing during winter and decreasing during summer. To model and predict this change one can use deep learning networks or regular statistical models.

In this paper I will firstly go through Feed Forward neural networks (FNN), and OLS-models and explain shortly how they work. Then I will introduce my own FNN models of different varieties and compare this to the results from the OLS-model. This paper will demonstrate that Feedforward Neural Networks (FNN) outperform Ordinary Least Squares (OLS) models in predicting glacier mass balance due to their ability to capture complex nonlinear relationships in the data.

Feedforward Neural Network (FNN) Model

A Feedforward Neural Network (FNN) is a type of artificial neural network where connections between the nodes do not form cycles (DeepAI, 2020). In essence, data flows in one direction—from input to output—without looping back. This architecture is particularly effective for tasks like regression and classification.

In the context of the given dataset, the FNN model processes features such as "TMPP" (temperature) and "Annual_SF" (annual snowfall) to predict the label "MB_Year" (mass balance for a year). The model comprises multiple layers, each contributing to the learning process. The first layer normalizes the input data to standardize the feature values, ensuring that they are on a similar scale, which improves the convergence during training.

Following normalization, the network consists of several dense layers with ReLU (Rectified Linear Unit) or sigmoid activation functions. Each dense layer is a fully connected layer where each node receives input from all nodes in the previous layer. ReLU and sigmoid introduce non-linearity, enabling the model to learn complex patterns in the data. Batch normalization layers are applied after each dense layer to stabilize and accelerate training by normalizing the outputs of the previous layer.

Dropout layers are integrated to mitigate overfitting by randomly setting a fraction of the input units to zero during training. This ensures the model generalizes well to unseen data by preventing it from relying too heavily on any single feature.

The final dense layer outputs a single value, which is the predicted mass balance for the year. The model is trained using mean absolute error as the loss function, which measures the average magnitude of errors in predictions. Optimizers like Adam adjust the model weights to minimize this loss, leading to a more accurate predictive model.

Ordinary Least Squares (OLS) Model

Ordinary Least Squares (OLS) is a classical statistical method used for linear regression. It estimates the relationship between independent variables (features) and a dependent variable (label) by minimizing the sum of squared differences between observed and predicted values.

For the dataset in question, which includes features such as TMPP and Annual_SF, and a label MB_Year, the OLS model aims to find the best-fitting linear relationship. The model assumes that the relationship between the features and the mass balance can be described by a linear equation.

The goal of OLS is to estimate the coefficients of this equation that minimize the sum of squared residuals (the differences between the observed values and the values predicted by the linear model). This minimization is achieved using matrix algebra, which solves for the coefficient values that yield the smallest possible sum of squared errors.

OLS is advantageous for its simplicity and interpretability. The coefficients directly indicate the impact of each feature on the predicted mass balance, making it easy to understand how temperature and snowfall influence the mass balance. However, OLS assumes linear relationships and may not capture complex patterns that nonlinear models, such as FNNs, can uncover.

Preparing files and creating the AI-model

For the group part of the project, I was on the Europe team. On the fileshare website there were uploaded three python-files for the groups to use. Two file-preparation programs and one program which calculates the mean average error (MAE) for the OLS-model. To use these programs, I downloaded the *.csv-files for Central Europe, northern Scandinavia and Southern Scandinavia. I then tweaked the code to fit the filenames and directory structure on my computer. I used the first preparation file to prepare the data and turn them into *.pkl-files (pickles) and then used the second preparation file to split them into validation data, train data and test data with a 10%, 80% and 10% split respectively. This ensures the model can generalize well to unseen data, preventing overfitting. Then these pickles were uploaded back to the fileshare website, so the other students could use them to train their AI-models and compare to the OLS-model for Europe. The regions have different number of datasets, EU ~100, NA ~50, HMA ~35 and SA ~10. This will show later in the paper.

I initially prepared the EU data split into the three regions, but I ended up not running these through either the OLS-model or my AI-model. Lastly, I ran the Europe pickles through the OLS-model which gave a MAE equal to 697.554. And uploaded this to the fileshare-website.

To create the AI-models I used the Tensorflow package for Python. I used two Python files, one with a 2-input model and one with a 24-input model. Which means that I used the same AI-model for all the geographic target regions (GTR), which are Europe (EU) (divided into Central Europe and North and South Scandinavia), North America (NA) (mainly the Rockies), South America (SA) (mainly Patagonia) and High Mountain Asia (HMA) (The Himalayas and the Karakoram mountain range). To develop my AI-model I tried different number of layers with different number of units, activation functions and dropout rates. I ended up with the structure shown in table 1. This structure is used both for the 2-input model and the 24-input model.

-	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6 (output)
Number of units	256	128	64	32	16	1
Dropout rate	0.2	0.2	0.2	0.2	0.2	0.2
Activation function	ReLu	ReLu	ReLu	ReLu	ReLu	ReLu
Activation function	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid

Table 1: A table overview of the layer structure of the AI-model. Tensorflow's BatchNormalization function is applied to all the layers with a batch size of 32. ReLu is short for Rectified Linear Unit.

I used the same learning rate for both models and for all GTRs, which was 0.001. However, I used different epochs for the different GTRs, so that when I plotted the MAE and loss as functions of epoch. One could see the training and validation losses flatten out and oscillate up to 2-4 percent around some local average. One can see the behaviors for the training and validation loss in figure 1 and 2 for the 2-input and 24-input model respectively. By trying out different epochs, one can cut off after the losses don’t change that much and start to oscillate around a local average, as can be seen from figure 1 and figure 2.

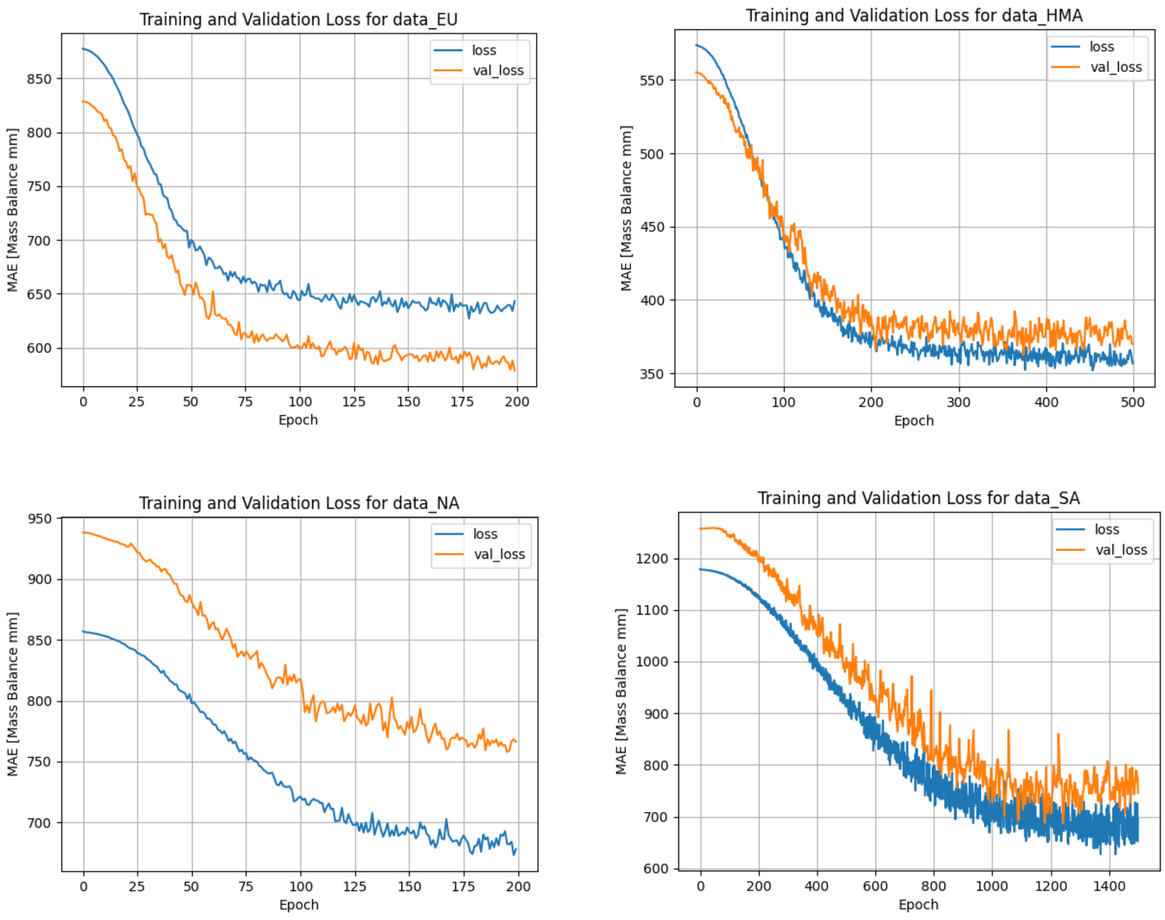

Figure 1: The training and validation loss of the 2-input ReLu-based AI-model for all four GTRs. For EU and NA epoch was 200, it was 500 for HMA and 1500 for SA.

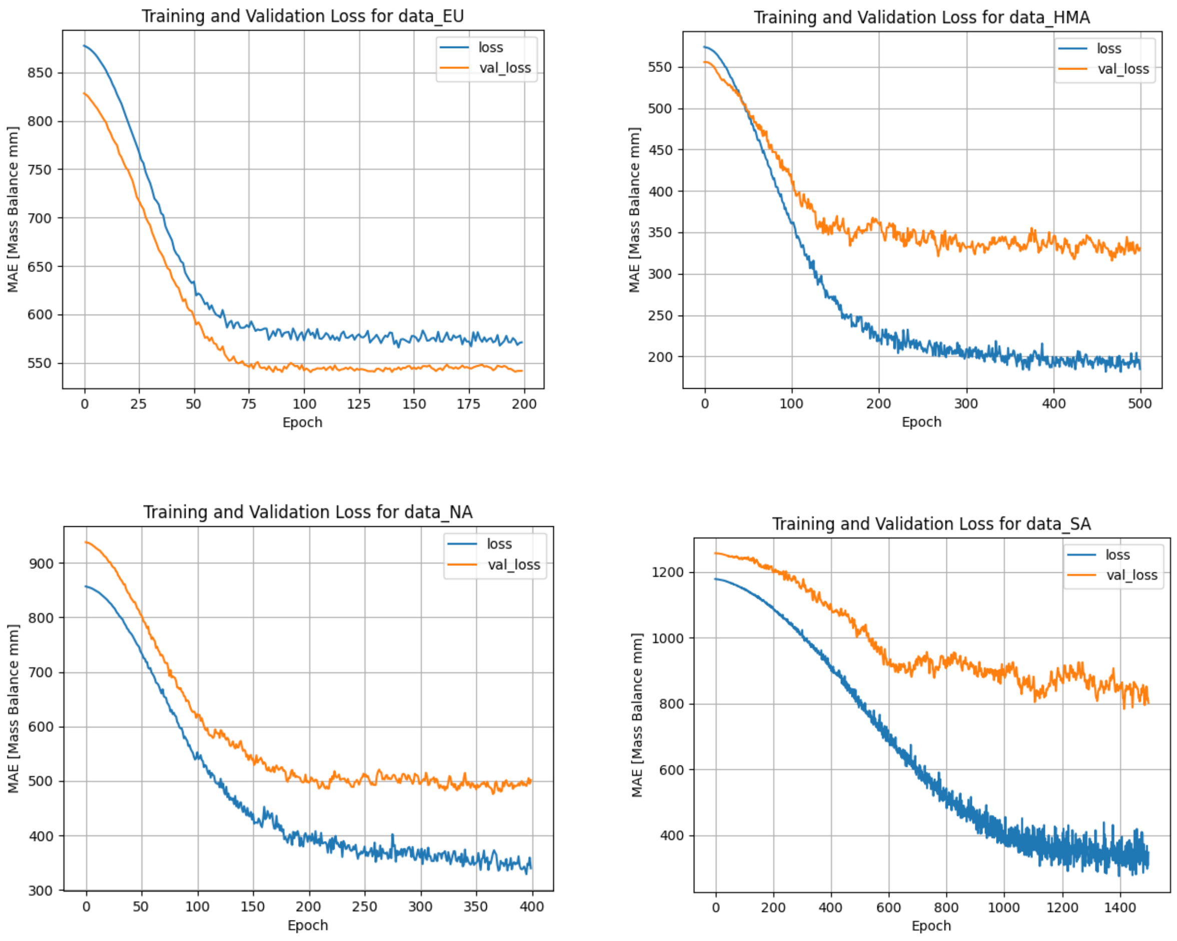

Figure 2: The training and validation loss of the 24-input ReLu-based AI-model for all four GTRs. For EU epoch was 200, it was 400 for NA, 500 for HMA and 1500 for SA.

I have not included figures showing the sigmoid-based AI-models, as the graphs have similar shapes, which is the important factor the figures above show. I did use the same epoch, learning rate and batch size for the sigmoid-based models, so the only thing that has changed is the activation function of each layer.

Results

Models	EU	NA	SA	HMA
OLS-model	697.554	783.777	961.176	424.356
2-input model, testing MAE	626.273	683.556	721.635	363.587
2-input model, validation MAE	578.749	766.140	766.052	369.934
24-input model, testing MAE	563.501	318.054	312.588	185.829
24-input model, validation MAE	541.584	501.475	802.015	330.745

Table 2: Mean average error for the different models with ReLu activation function, for the AI-models I included the MAE for both the testing dataset and the validation dataset.

Models	EU	NA	SA	HMA
2-input model, testing MAE	626.239	679.293	905.105	365.313
2-input model, validation MAE	649.931	750.986	880.180	378.448
24-input model, testing MAE	567.138	386.873	354.062	215.393
24-input model, validation MAE	549.859	515.347	700.974	331.678

Table 3: Mean average error for the different models with sigmoid activation function, for the AI-models I included the MAE for both the testing dataset and the validation dataset.

Discussion

Ordinary Least Squares (OLS) Model

The OLS model, which relies on annual snowfall and temperature data, exhibited the highest MAE values across all regions. The higher MAE values for SA can be explained by the amount of data, which was considerably lower than the other regions. It is essential to recognize that both AI models consistently outperformed the OLS model, no matter the input nodes or activation functions. This, including the runtime of the AI-models (which never were longer than one minute), the OLS-model is clearly least suitable to analyze glacier's mass balance through the years.

2-Input AI Model

The 2-input AI model with ReLU activation shows improved performance over the OLS model in all regions and for both activation functions. These results indicate a significant reduction in error compared to the OLS model, particularly in SA with ReLu activation function, where the MAE decreases by approximately 25%. The MAE for the models is approximately equal for both activation functions except for SA, where the sigmoid-based model is closer to the OLS result than the ReLu-based model. There is also a significant increase in validation MAE from the ReLu-based model to the sigmoid model for EU. So, for this 2-input AI-model, one would rather use the ReLu activation function than the sigmoid. To avoid the artifacts and bad results for EU and SA.

24-Input AI Model

When compared to the ReLU-based 24-input model, there is a slight increase in both testing and validation MAE in EU, indicating a minor reduction in performance. In NA, the testing MAE increases significantly, while the validation MAE shows a slight increase, suggesting a performance decline. The testing MAE in SA increases, but the validation MAE decreases, indicating improved generalization despite worse testing performance. In HMA, both testing and validation MAE values increase, reflecting a decrease in model performance. The significant decrease in the validation MAE for SA can again be explained by the low amount of data. Given this, and some of the weird drop-offs and increases in the performance of the AI-models in the SA region. It is best to give more weight to the results of the other regions. Doing this shows that use of the sigmoid activation function in the 24-input model results in higher MAE values, suggesting that the ReLU activation function might be more suitable for capturing the complex patterns in the glacier mass balance data.

Regional Analysis

In EU, the 2-input model with sigmoid activation performs similarly to the ReLU-based model, with minor variations. The 24-input model shows slightly higher MAE values with sigmoid activation, indicating a marginal decrease in performance. In NA, the 2-input model with sigmoid activation performs slightly better in both testing and validation compared to the ReLU-based model. However, the 24-input model shows a significant increase in testing MAE and a slight increase in validation MAE, suggesting that ReLU might be more effective for this region. In HMA, the 2-input model's performance remains stable with minor increases in MAE values. The 24-input model shows increased MAE values, indicating a reduction in accuracy compared to the ReLU-based model.

South America presents a unique challenge, with the 2-input model with sigmoid activation showing a substantial increase in both testing and validation MAE, indicating a marked decline in performance. The 24-input model, however, exhibits improved validation MAE despite worse testing performance, suggesting better generalization. As explained above the amount of data is probably the cause of this MAE decrease in SA for the sigmoid-based 24-input model. As one of the datasets in the validation set might just fit better than for the ReLu-based model, and thus significantly decrease the MAE, as this one dataset has a lot of weight when there are so few.

Conclusion

The comparison between sigmoid and ReLU activation functions in AI models for glacier mass balance prediction reveals nuanced differences in performance. While sigmoid activation offers slight improvements in some regions, it generally leads to higher MAE values compared to ReLU, especially in the 24-input model. The ReLU activation function appears to capture the complex, nonlinear relationships in the data more effectively, particularly for the 24-input model.

This paper highlights the superiority of FNN models over OLS models in predicting glacier mass balance, emphasizing the importance of capturing nonlinear patterns in the data.

References

(1) DeepAI. (2020, June 25). Feed Forward neural network. DeepAI. https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network