A Convolutional VS a simple Neural Networks for mapping Glaciers using Multispectral Remote Sensing Data

Project 3, FYS-STK4155, UiO

Sergio Díaz, Varvara Bazilova

December 2020

Abstract

Understanding of the glacier extent change is crucial for the climate change assessment. Multispectral remote sensing images now have about 40 years of record and allow mapping of near global glacier extent with a very dense temporal resolution. However existing methods for glacier mapping only allow the semi-automatic way of delineation and still require a lot of manual corrections. In this report we present an attempt to apply Neural Networks of different kind (Fully Connected Feed Forward (FFNN) and Convolutional assimilating the U-Net architecture) to improve the automatic way of glacier mapping. We show how hyperparameters tunning on both models affect the classification results to obtain the final settings. We conclude that U-Net with ReLU activation function performed better than the FFNN model, since the architecture is designed under explicit assumption of working with image segmentation, whereas the FFNN model is still very sensible to band spectral reflectance and could not understand spatial patterns of the features as the U-Net does. Running Convolutional Neural Networks requires a lot of computational resources, so we also explore the way to get away of this problem (by dividing the data in smaller subsets etc.). We conclude, that result strongly depend on the activation functions. Also, that using of some infrastructure like Google Cloud/AI Platform/Earth Engine would allow to avoid data-size bottlenecks and improve the analysis.

Plain English Summary

We tried to make a good method for automatic mapping of glaciers from satellite images, since usually, when climate scientists need the information about glacier extent, it is not easy to get it: existing methods require people to correct and double-check everything manually to get decent results. That is a lot of not-so-smart-manual-work, especially if you are interested in the planetary scale information. So we used the U-Net architecture from Ronneberger et al. (2015) for image segmentation, and because is a fancier model than the simple convolutional network examples. We also tried to compare it with a feed forward fully connected neural network just to explain the obvious thing, that U-Net works better for these purposes of working with images. We make an attempt of computing new 3-band images as dataset to feed it to the models; each band being a operation between the Landsat 5 bands (based on our remote sensing knowledge). After waiting for a lot of hours, we came to see that the FFNN model does similar thing as state of art "classical" remote sensing methods, so... great! On the other hand, U-Net was able to go beyond this by actually associating principal patterns and geometries... wow! However, we were not completely happy with the results, since we expected them to be better. Because of time and computer power limitations, we didn't use for the U-Net the planned training dataset size. But after a valid analysis (and some pitty for our self esteems) we think we can blame this as a huge bottleneck to explain why U-Net didn't achieved its truly potential (or expected potential). We learned a lot at the end, including that fancy not-available-for-free GPUs (like Google Cloud/AI Platform) could be a great useful advantage for running processes to achieve the best, instead of just playing games.

1 Introduction

The cryosphere includes frozen components of the Earth system. Advances and retreats of mountain glaciers in the Arctic and Alpine environments are one of the most visible signs of climatic changes (Hock et al., 2019). At the same time, cryosphere and glaciers in particular are very vanurable to the current climatic changes and play an important role as the climate change indicators.

Despite a lot of effort is being put into in-situ glacier observations (e.g. WGMS - World Glacier Monitoring Service), it is often hard to maintain long time-series of observations. Field campaigns to the glacierized areas are hard, time-consuming and expensive (often dependent on short-term project funding). Furthermore some areas in the world can not be accessed by researchers due to political complications (e.g. Kashmir, Caucasus, etc.).

Satellite data are widely used for repeat mapping of glacier extent to asses the effect of climate change on the river runoff, mass balance changes, calving, local climatic conditions (Vaughan et al., 2013). Glacier outlines, derived from remotely sensed data are used as input for various Earth system models. Freely available satellite data at 10 to 30 m spatial resolution (e.g. NASA Landsat mission, ESA Copernicus programme etc.) with up to 2-3 days (depending on the latitude) allow semi-automated mapping of clean ice using specific spectral bands (Paul et al., 2016).

The most common method for glacier mapping utilizes the spectral properties of the snow and ice: the threshold is applied on a ratio or normalized difference of bands in different parts of the spectrum. However, these methods are commonly used for mapping of the clean ice and can not be applied in a completely automatic way for obtaining a glacier mask (a lot of manual corrections are required). The challenging issues for the latter are ice in shadow or under debris cover (e.g. medial moraines), calving/icebergs (for the moraine or freshwater terminating glaciers). Therefore, there is the demand for more sophisticated methods, that would not only take the reflectance of the pixels, but would allow to consider "glacier" as an object for image segmentation.

Therefore the aim of this report is to apply the Neural Network algorithms (Feed Forward and Convolutional) for mapping of marine terminating glaciers and discuss and evaluate the performance of these methods.

In this report we used multispectral Landsat 5 data (see "Data" section). The code for obtaining the data is available as a supplementary material (see "Code availability" section).

The report consists of: 9 major sections and supplementary material, including Introduction, Background of the study, description of Data and Methods, Results, Discussion and Conclusions, Code availability and the list of References. Supplementary material includes the list of satellite images, used for the study.

2 Background

2.1 FFNN

A neural network (NN) is a two-stage regression or classification model (Friedman et al., 2001). NN is typically represented by a NN architecture (the combination of input, output and hidden layers), number of nodes on each layer, vectors of biases [b] and weights [w] (usually initialized as normally distributed numbers between 0 and 1) and activation functions, connecting the layers.

Feedforward Neural Network (FFNN) is the simplest type of NN. The layers are connected with each other in one direction (forward). To train FFNN backpropagation is used. Backpropagation utilizes the derivatives of the cost function with respect to weights and biases and updating weights and biases after comparing the output with the targets.

The FFNN and backpropagation algorithm are described in the report 2 (Bazilova and Díaz, 2020) in detail.

2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is the type of NN, that does not only operates on pixel values, but takes neighbouring pixels into account (by specifying the kernel size)1. Applying the convolution process on the features, the CNN can be useful for image segmentation and delineating objects, that don't have clear spectral differences (pixel values). The trained CNN can recognise common spatial patterns, that are not reflected in pixel values. The main idea of the CNN is to reduce the images into a form which is easier to process, without losing features and patterns. CNN architecture makes explicit assumption, that input data are images and tries to "filter" the image, based on the feature properties.

As FFNN, CNN consists of NN architecture (the combination of input, output and hidden layers), number of nodes on each layer, vectors of biases $[b]$ and weights $[w]$ and activation functions. In addition, CNN has pooling layers, which together with activation functions connect hidden layers with each other.

The main underlying mathematics is similar to FFNN, however the procedure consists of some additional steps.

In general CNN has three basic stages: 1) several convolutions, that can run in parallel with weights and biases to train and set of linear activation 2) run through the activation functions 3) pooling stage.

Convolution and run through activation functions and called "detector stage".

Unlike, FFNN CNN has fewer parameters to optimize on each layer.

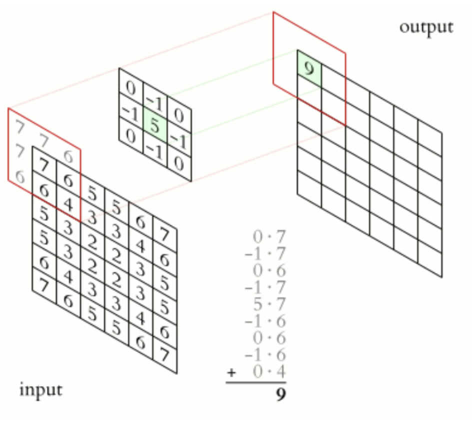


Figure 1: Principle of convolution (Sourse: Google Outreach)

The main advantage of the U-Net comparing with the simple CNN is the upwards part of the U-shape (2). Each final output of the convolutions layer in the downwards part of the U-shape are concatenated with the inputs of the upwards part. This allows that generalized features in the lower layers are computed aside with the more detailed features of the first convolutions in downwards motion of the U-shape, allowing to describe the specific features inside the generalized ones at the bottom, and thus, creating the pixel-wise classification which ends up being the image segmentation method.

3 Data and Methods

3.1 Data

The data, used for the analysis are multispectral Landsat 5 Thematic Mapper (TM) images. Landsat satellite mission is a joint program of USGS (United States Geological Survey) and NASA (National

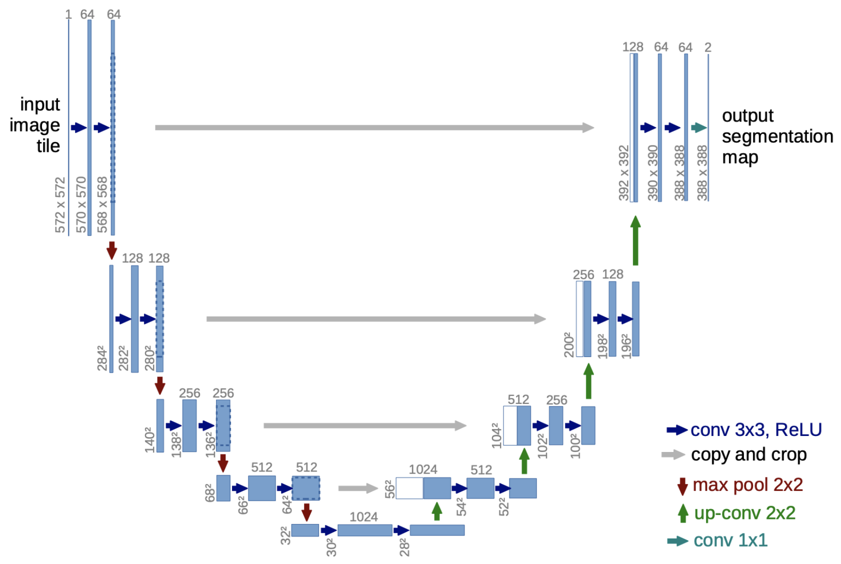


Figure 2: U-net architecture used by Ronneberger et al. (2015). "This example consists of (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations." (from Ronneberger et al. (2015)).

Aeronautics and Space Administration). Landsat 5 has been launched in 1984. Depending of the region, the earliest data us available since March 1984 (or slightly later, depending on the location) until 2012. Landsat 5 images consist of 7 bands from visible blue to shortwave infrared part of the spectrum. The Landsat 5 images, used are calibrated at the calibrated top-of-atmosphere (TOA) reflectance. The algorithm to convert calibrated Digital Numbers (DN) to physical units (TOA reflectance) is described in (lan, 2009). The use of TOA reflectance allows to standardized the data and reduce the biases when comparing the images acquired on different dates or by different sensors.

The bands, used for the reports are: blue (B1), green (B2), red (B3), near infrared - NIR (B4), shortwave infrared - SWIR (B5). Since ice and snow exhibit strong differences in spectral reflectance between SWIR and the visible part of the spectrum, the common method to map snow and ice is to use a "band ratio" image. Therefore, in addition to the TOA reflactance Landst 5 bands, the band ratio between SWIR and visible Red was computed.

As well as this, the Normalized Difference Snow Index (NDSI), which is another common way to map snow and ice was computed using the visible green and SWIR bands: (**??**) (Hall et al., 2001). NDSI utilizes the significant reflectance difference of snow and ice in visible and infrared parts of the spectrum.

$$NDWI = \frac{(Xgreen - Xswir)}{(Xgreen + Xswir)} \tag{1}$$

Another "product" which is used in this study is modified Normalized Difference Water Index (mNDWI), which uses green and NIR bands of multispectral optical image (2. The variations of Water Index are widely used for water classification (McFeeters, 1996):

$$NDWI = \frac{(Xgreen - Xnir)}{(Xgreen + Xnir)} \tag{2}$$

The band ratios is more sensitive to the noise, then the Normalized Difference indexes.

The ”ground truth” data, that are used as ”target values” for training the Neural Networks are glacier outlines, provided by Randolph Glacier Inventory and Global Land Ice Measurements from Space international initiative (Pfeffer et al., 2014) (hereafter GLIMS dataset). The initiative aims to repeatedly survey the world’s estimated 200,000 glaciers.

The data has been obtained using Google Earth Engine - the cloud based platform for planetary-scale geospatial analysis, that brings together remote sensing datasets (collections) and Google Cloud infrastructure and computational resources (Gorelick et al., 2017). The images for the analysis were downloaded to the local machine as spatial subsets of original Landsat 5 scenes 3. The list of Landsat 5 scenes used is available in supplementary material.

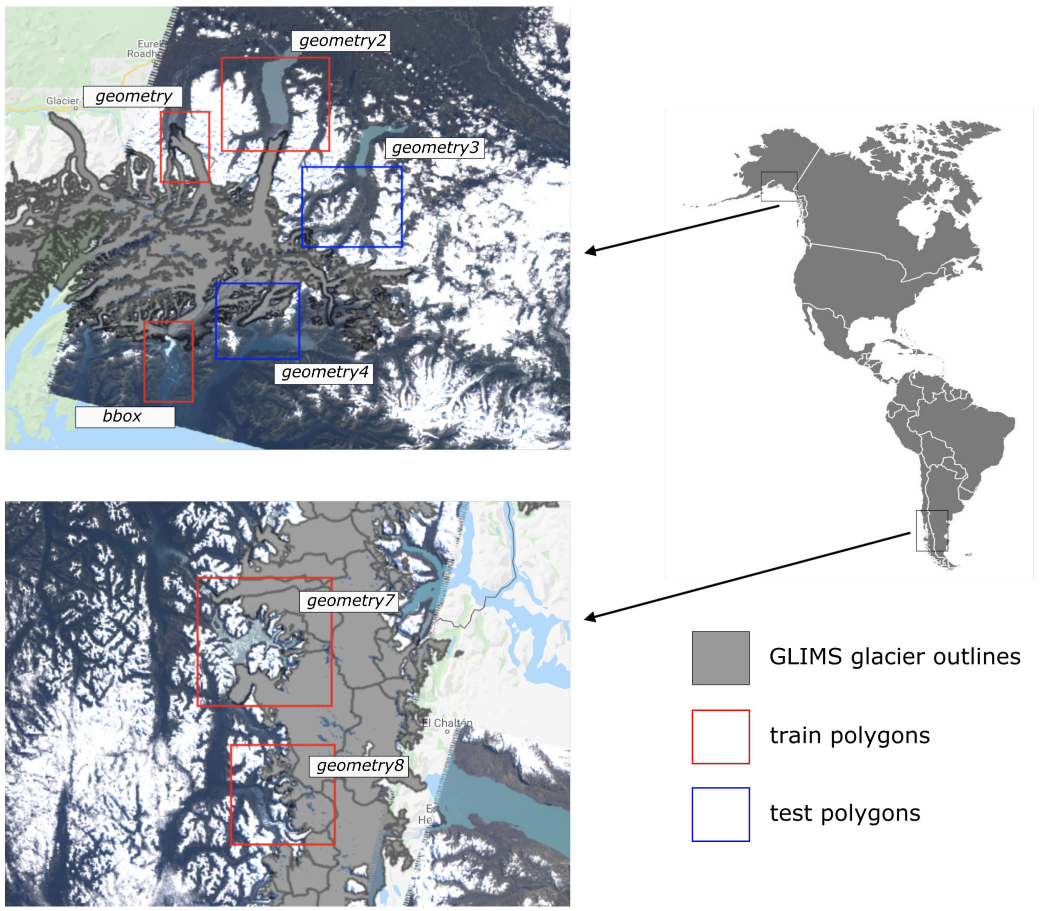


Figure 3: The study area: train and test polygons as subsets of Landsat 5 scenes (background: True Color Composite Landsat 5 scenes

3.2 Methods

3.2.1 Features

The methods used were originally designed to use a regular RGB-composite image. However, since multispectral satellite images consist of more then 3 bands, we had to select the three bands that would describe

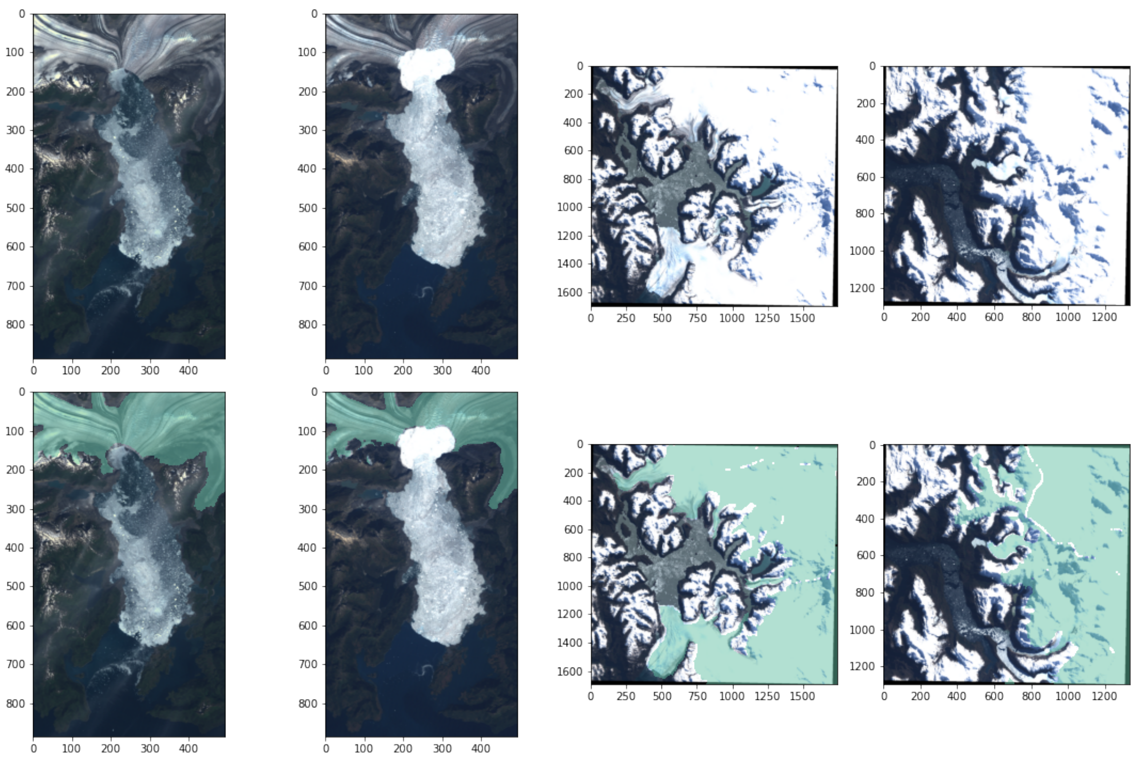


Figure 4: The four images used for the training dataset. Original images in RGB color are on top and with the GLIMS mask (delineated glaciers) on the bottom (transparent green).

the properties of the objects of interest and delineate the glaciers in the best way. In this case, instead of taking it as visible colors bands, that make the natural color composite image, we decided that we should use three important features, that we consider the most important for recognizing glaciers.

The true color composite is an important feature to map the glacier extent and can highlight important boundaries, as normally human would be able to do manually. However, for manual mapping of the ice false color composite is used more often (with SWIR, NIR and Red bands instead of RGB, respectfully) (Paul et al., 2016).

We considered the following feature combinations: 1) Blue band (snow and ice have very high reflectance in this part of the spectrum), 2) band ratio (SWIR/Red), 3) mNDWI (that would help to exclude water for the marine/lake terminating glaciers). This combination can be observed in Figure 5

3.2.2 Data Augmentation

We were aware that there was a computational power limitations. Figure 3 shows that only 4 original images were used for training and 2 images used for test (with two subsets treated as an "unseen data"), and all their sizes are non equal between them. The data problem was concentrated on the image sizes and the number of available data. In order to assess these problems, then the data augmentation was made.

The data augmentation consisted on dividing all the images into several pieces (mini-images) of a defined pixel size. If the pixel value is 128, then the image would be divided into pieces of sizes 128x128 by going in row and column iteration. If the double of the pixel input is bigger than one of the original spatial dimensions of the image, then it would only crop one mini-image in one of the dimensions.

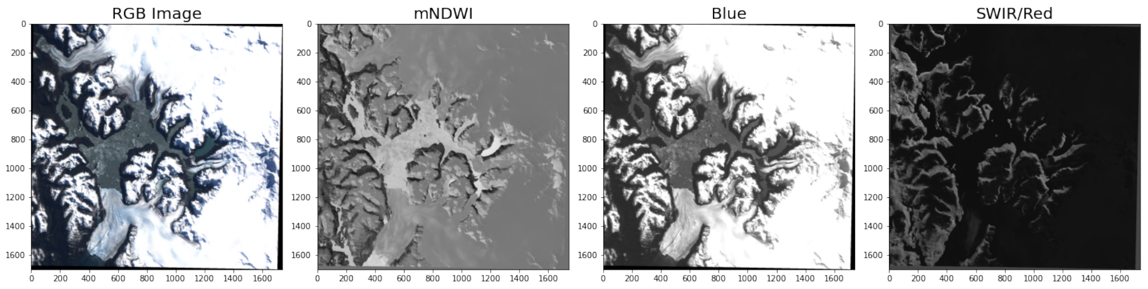


Figure 5: RGB Image from *geometry7* aside with the used feature combinations: mNDWI, Blue and SWIR/Red computed bands.

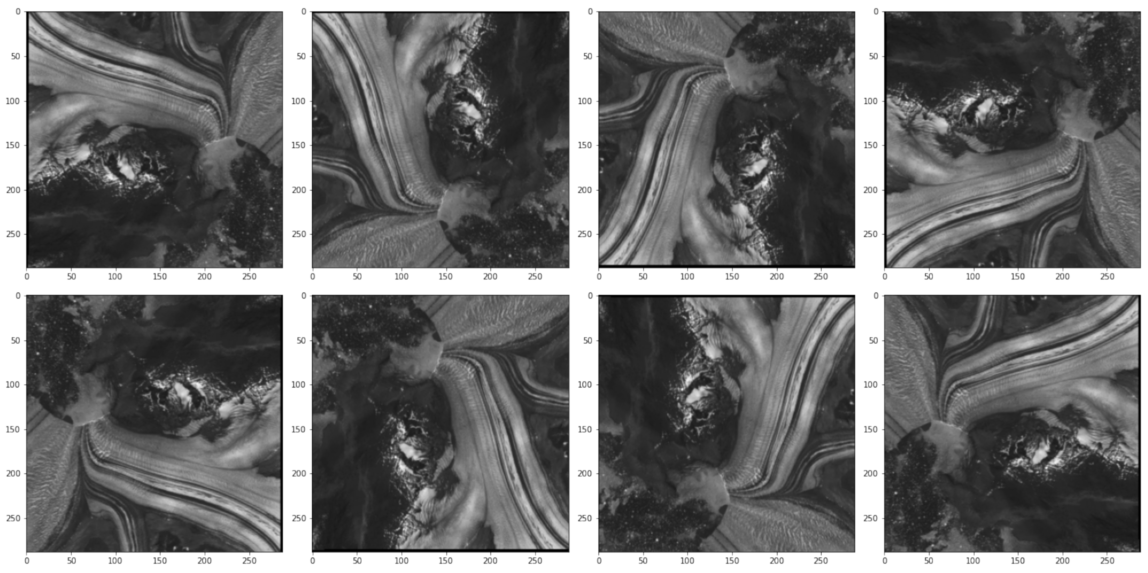


Figure 6: Data augmentation example. One of the mini-images can be observed, all its possible 90 degree rotations and same wise with the reflected image.

The next step was to create more artificial samples by rotating each mini-image and their reflections. Figure 6 shows an example of this by using a mini-image with size of 288x288, and we managed to increase in this example the amount of data from 1 to 8 mini-images.

3.2.3 Neural Network Models

In order to be sure that the performances for both methods are okay, both of them were made using Keras API under Tensorflow library (Abadi et al., 2015; Chollet et al., 2015). We decided this under the conditions established under the options and instructions of the task. Our Neural Network knowledge is based on the previous report, we have made: (Bazilova and Díaz, 2020), where the methods, Neural Network mathematical background are described in details.

The FCNN (or NN) was made on the last project (2) (Bazilova and Díaz, 2020) using Keras, and also made ourselves, which we named Beta NN. This last one can be found in the *nn.py* files under the class NeuralNetwork (Bazilova and Díaz, 2020). We decided to use only the Keras model because of our trust on Tensorflow library, even though, to our surprise, the Beta NN seemed to perform better than the Keras NN. However, the main advantage here is that by using the Keras NN we do not need to worry too much about any normalization or standarization of the data. Each way of pre-processing the data proved to affect the performance significantly in the last study (Bazilova and Díaz, 2020), so by taking this out and letting Keras assess this work, we would use the available time more efficiently.

Since this is a binary problem, the classification must also be binary. We set an output of two neurons to predict probabilities for each class, however, we are aware that this could have been done with one neuron and a determined probability threshold.

For the CNN, we replicate the U-Net architecture from (Ronneberger et al., 2015) using the libraries of Keras. The code for this can be found at *nn.py* file named under the class U_Net. We constructed exactly the same architecture but with different (smaller) feature sizes since we were planning to divide the images to a smaller shape than the ones used by the original article (Ronneberger et al., 2015) and because it would be more computationaly expensive, a luxury we can not afford right now due to computer and time limitations. The feature sizes varies according to the image input shape, but once the model is run with a certain image size (one fixed) then all the data that want to be classified must also be of the same shape.

Also, in this U-Net model we established a Early Stopping method from Keras (Chollet et al., 2015; Abadi et al., 2015). This method monitors the variations of the performance of the train set during epochs, and decide to stop the training if the variations are low (meaning a possible convergence) during a certain iteration number. In this case, the iteration number established to monitor the accuracy value of the training set was of 15. As an addition, this time we used the inside functions of Keras to split the dataset into train and test. The function inside the training function first shuffles the dataset and then splits it in 20% and 80% for the train and test, respectively (Chollet et al., 2015).

As in previous report, in order to find the best hyperparameters for the FFNN, we also tried different configurations by exploring different learning rates, activation functions and penalty parameters. As for the CNN, due to computer limitations, we explored the performance between the Keras "Adam" optimizer and the SGD method with different learning rates as in the FFNN, different activation functions, and most important, the image sizes since this would tell how important is to take into account the "big picture" of the glacier distributions. Nevertheless, we assigned as a default optimizers the Adam and the SGD to the U-Net and the FFNN models, respectively.

Finally, based on the hyperparameters tunning, we set a for both methods the best models of each one for both to be finally be compared in accordance to the ultimate goal, the classification. We made the two final models to classify the images used on train to validate the performance, and other two images not used on the train set with unknown solutions.

4 Results

4.1 U-Net (CNN) hyperparameters tunning

As mentioned before, we have done several hyperparameter tunnings. Figure 7 shows different performances of the U-Net with dependence on the input shape, meaning the image sizes. Size 288x288 presents the best test accuracy score, reaching near 0.9, aside with the test performance of size 384x384. The other sizes can just reach to 0.8 to 0.85 of accuracy score on test sets. For the training set, all sizes reach to an accuracy score of near 1.0, however 288x288 shows less variations and more stability on the performance than any other size while iteration umber increases, both for train and test sets. Tanh seems to perform similar to ReLU on the test sets but ReLU shows to be best in the train set.

Regarding the activation functions variability, Figure 8 illustrates the behavior of all the tested activation functions for the U-Net architecture. Sigmoid function shows the worst performance for train and test datasets, even if it shows to be the more stable one. On the other hand, Leaky ReLU shows to be the most efficient because of their high performance scores on train and test datasets, with a huge stability on

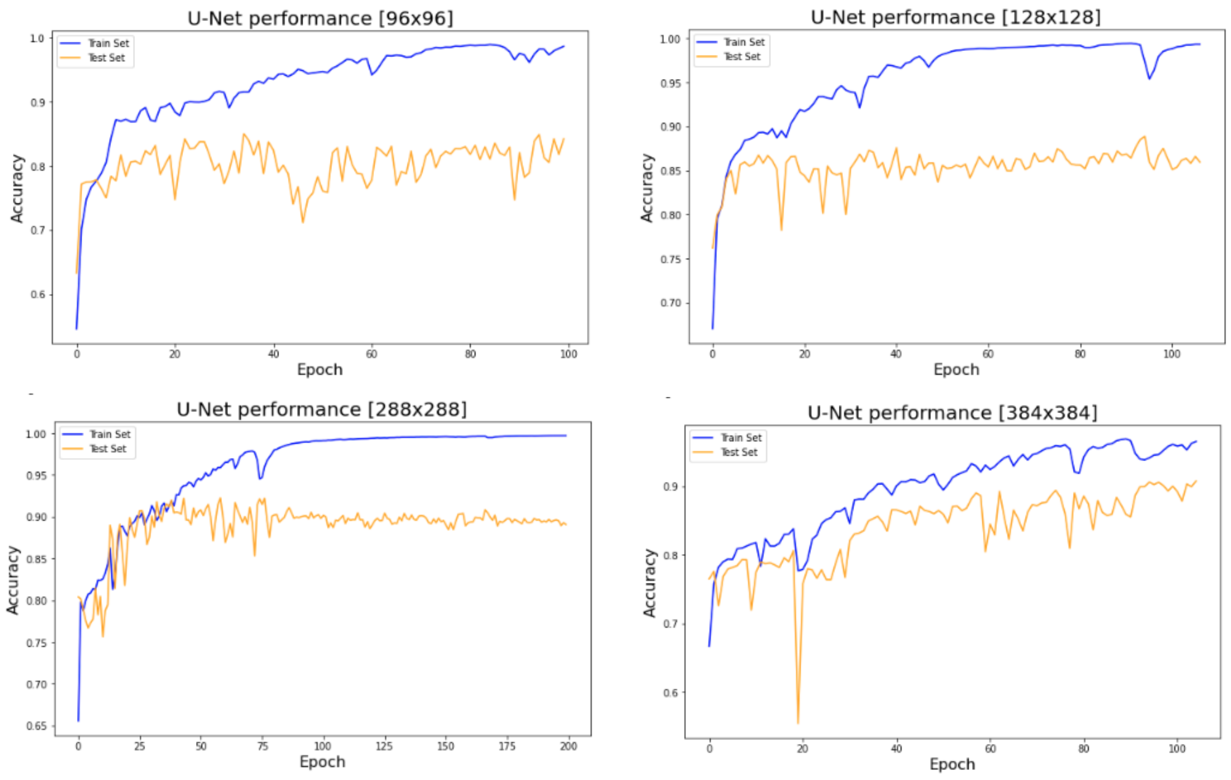


Figure 7: Performance of the U-Net in relation to the input size. Images fed where of sizes: 96x96, 128x128, 288x288 and 384x384. Fixed parameters: Optimizer = Adam, Activation function = ReLU, Epochs = 200, Penalty = 0.0.

the train set, alongside with ReLU.

4.2 FFNN hyperparameters tunning

For the FFNN model, Figure **??** shows the variations of learning rates and penalties against the epochs, and results are more than decisive. For the train set, the accuracy increases from learning rate ranges of near 0.0009 to near 1.71, with a peak of performance near 0.3. In the test sets we see the same behavior but with more noise. This, as saw in other projects (Bazilova and Díaz, 2020), is normal for the test set to present more variations, but the tendency of the learning rates ranges for high performance is mostly the same, including the defined peak.

As for the penalty parameters, both train and test sets show consistency on the behavior. Figure 9 shows that both of them increase the performance each time that the penalty parameter decreases. The test set shows the noise in performance in accordance to what is normal to see. However, in the last one is more noticeable the change from bad performance to a good one at a penalty of 10^{-6}.

For activation function variations, Figure 10 shows the performances of four different types of activation functions. The ReLU on both sets performs as the best, and the Tanh function as the second one which is quite close to it The Sigmoid function seems to also be good in the training set, but on the test set it doesn't perform well, but rather close as the Linear function, which stopped to increase the performance in the very beginning. The Sigmoid and the Tanh functions experimented a decrease in the early iteration

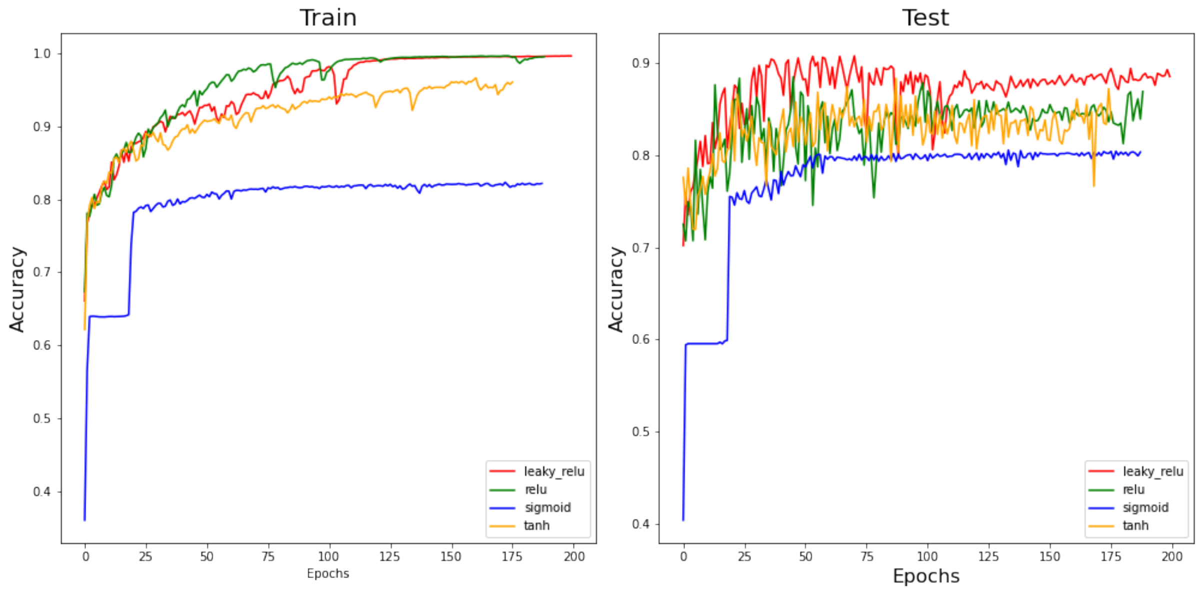


Figure 8: Different activation function performances on train and test datasets for the U-Net model. The different activation functions presented are Leaky ReLU, ReLU, Sigmoid and Tanh with red, green, blue and orange color, respectively. Fixed parameters: Optimizer = Adam, Epochs = 200, Penalty = 0.0, Image size = 288x288.

stages before rising again in the last stages. On the other side, the ReLU never showed this behavior, but rather a continuous increase until reaching an accuracy limit, the best one between all the activation functions.

For the hidden layers and the nodes variability, the train and test sets didn't performed the same (see Figure 11). Train set shows higher performance between a number of 3 and 5 hidden layers, with an amount of 80 to 140 nodes, showing a high accuracy score region inside this range. The test set on the other hand shows noise in such a way that no tendency can be determined.

4.3 U-Net test on Glacier classification

After changes in hyperparameters, the best model regarding the U-Net shows outputs in Figure 13 and 14. The Figures shows the performance for glacier classification with the final defined U-Net chosen model. Also, we used two extra unknown images with unknown solution to it as a final test (see Figure 3, blue-line boxes). The best model chosen hyperparameters are showed in Table 1. Another solution is presented just to validation the case in which Leaky ReLU was chosen (see Figure 12).

4.4 FFNN test on Glacier classification

For the FFNN model we also choose the best model. Figure 15 and 16 shows the performance for glacier classification with the final FFNN chosen model for the same images in the same procedure as with the U-Net. The hyperparameters chosen for the final model can be seen at Table 1.

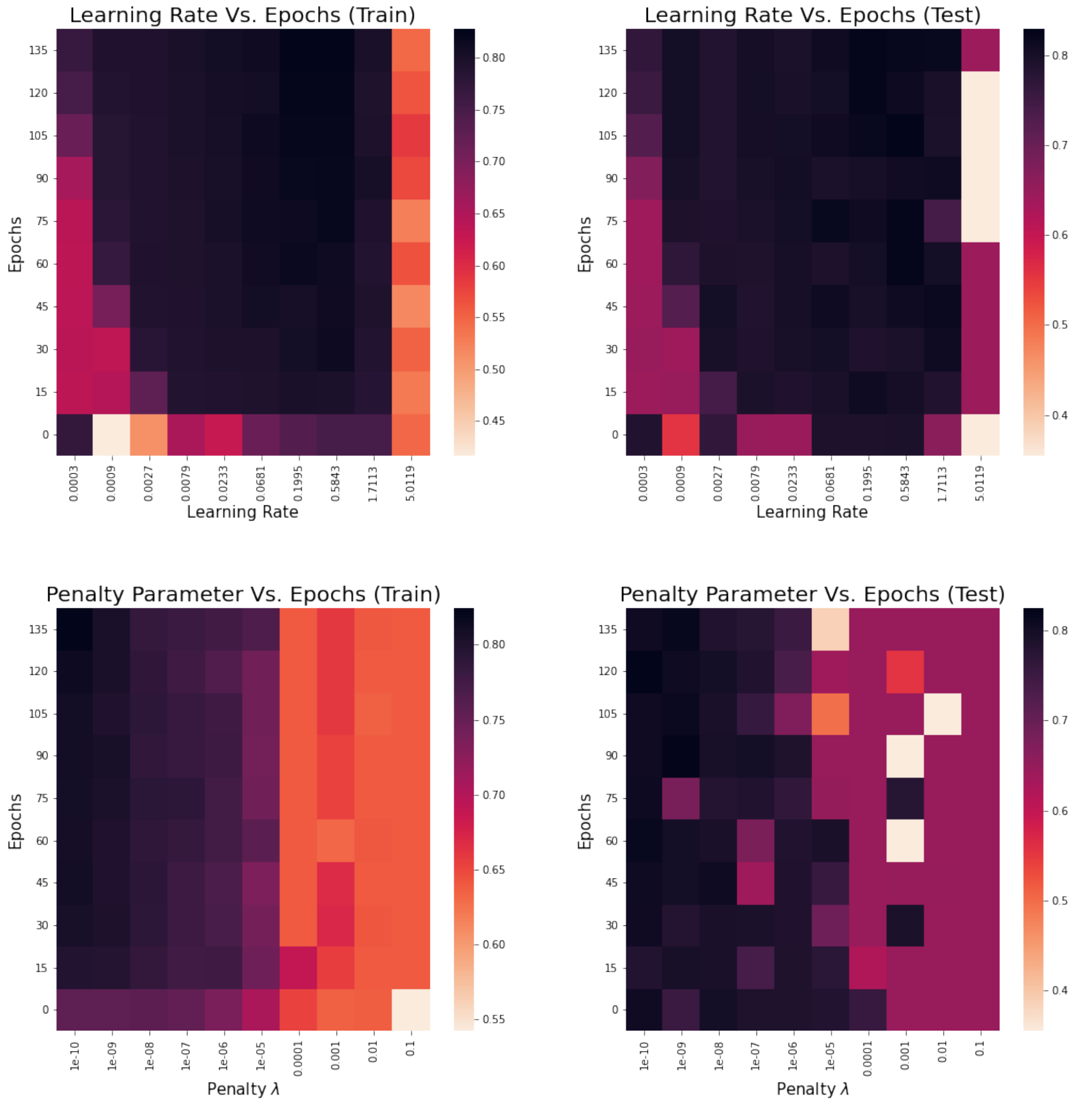


Figure 9: Learning rate and Penalty values variations for the FFNN against Epochs. The values report Accuracy score. Fixed parameters: Optimizer = SGD, learning rate = 0.3 (for penalty tuning), penalty = 0.0 (for learning rate tuning), activation function = ReLU, nodes = 120, layers = 1.

5 Discussion

5.1 U-Net

The image size variation was great to monitor how important is to get the big picture. 384x384 pixel images works best because it proves to be able to keep increasing performance so the train set reaches an accuracy of near 1.0, test set reaches near 0.9, which is higher than the rest. Nevertheless we have a huge limitation which is computer power, and time. Because of this, we decided that the 288x288 pixel size image is the best to use as the best result because it accomplish the highest trade-off between computation time, and

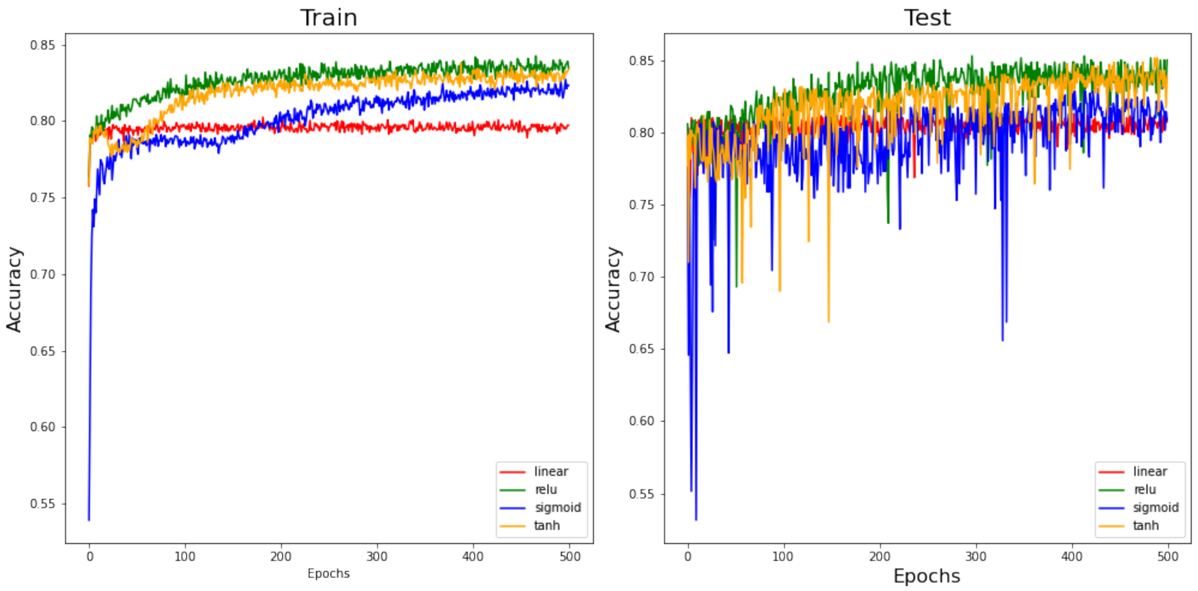


Figure 10: Different activation function performances on train and test datasets for the FFNN. The different activation functions presented are Linear, ReLU, Sigmoid and Tanh with red, green, blue and orange color, respectively. Fixed parameters: Optimizer = SGD, learning rate = 0.3, penalty = 0.0, nodes = 120, layers = 1.

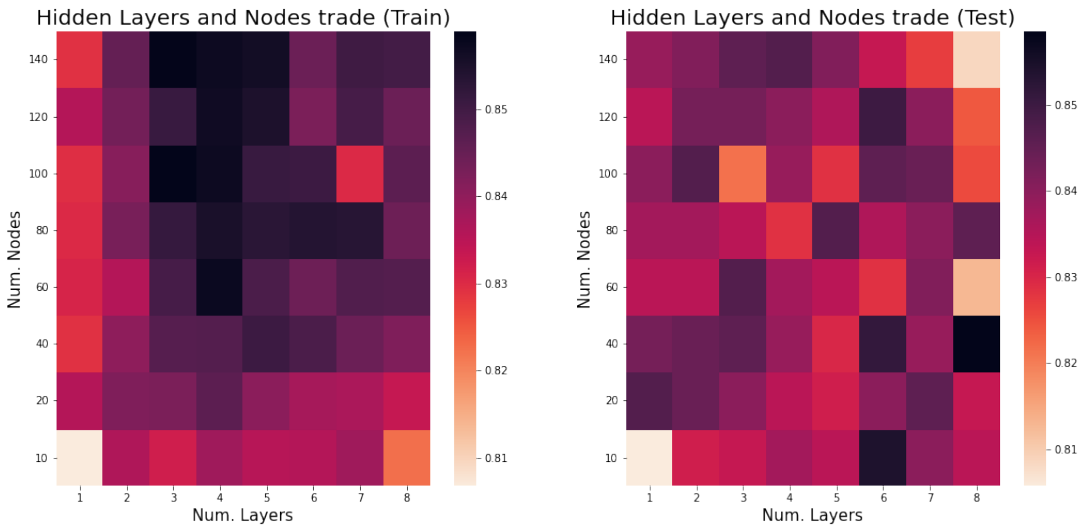
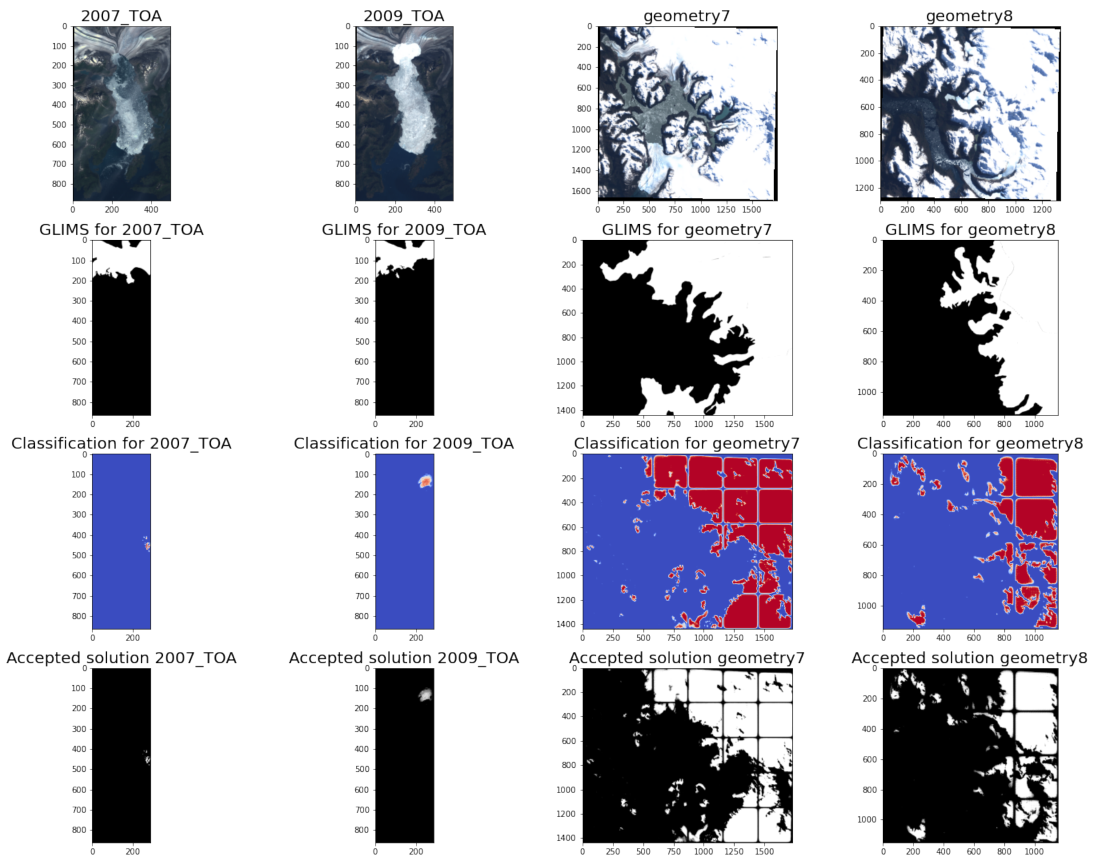


Figure 11: Performance of the FFNN towards variations in node numbers and layer numbers for train and test datasets. Fixed parameters: Optimizer = SGD, learning rate = 0.3, penalty = 0.0, activation function = ReLU.

geometry capture. This last concept is important, and this is what explains the good performance of the size 384x384. When there is a bigger image, the CNN can visualize the bigger picture, and allows it to



2007_TOA
2009_TOA
geometry7
geometry8
GLIMS for 2007_TOA
GLIMS for 2009_TOA
GLIMS for geometry7
GLIMS for geometry8
Classification for 2007_TOA
Classification for 2009_TOA
Classification for geometry7
Classification for geometry8
Accepted solution 2007_TOA
Accepted solution 2009_TOA
Accepted solution geometry7
Accepted solution geometry8

Figure 12: Test performance of the final U-Net model with hyperparameters chosen from the best results of the tunning process, but using Leaky ReLU as activation function.

Parameter	**FFNN**	**U-Net**
Activation Function	ReLU	ReLU
Optimizer / Learning Rate	SGD / 0.3	Adam
Penalty (Regularization)	0.0	0.0
Epochs	400	120
Layers (Architecture)	4	5 (MaxPooling)
Nodes	100	2 (Conv. Processess)
Image Sizes	NA	288x288

Table 1: Summary of chosen hyperparameters for the U-Net and the FFNN models. NA = Not Applicable. Conv. = Convolutional, SGD = Standard Gradient Descent.

recognize detailed patterns like the outlet parts of the glaciers, the terminus positions etc. So basically a more generalized picture means a more recognition of the class geometries.

Nevertheless, one possible disadvantage of this decision was that, if we see the plot of 288x288 carefully (see Figure 7), the test performance after 50 epochs began to decrease slowly while the train set keeps increasing. This could be an indicator that the model is beginning had entered in the overfitting phenomenon, but since the test decrease is very low, we decided to take the chance of choosing this input shape.

In terms of activation functions, Leaky ReLU seemed to perform the best. However, the way it performed on the test set was strange (see Figure 8). It begins by increasing performance, and suddenly for extended epochs it began to decrease and increase again. This made us doubt of the reliability of Leaky ReLU. Besides this, image classification and image segmentation, the most used activation function because of fast convergence speed, simple calculation and good results is ReLU (Lin and Shen, 2018). So at the end we decided to go for ReLU as the final activation function.

From the rare Sigmoid flat-curve behavior (see Figure 8), we could think that this is because of saturation in 0 or 1. Due to the relation of the continuous increase on the other functions, we believe that the neurons outputs are just values saturated above 1.

We wanted to test the performance between the Adam optimizer and the SGD with different combinations of learning rates. However, the training of the U-Net model takes too much time, and we wanted to be loyal to keep the other hyperparameters as fixed as possible so we could have an ideal comparison field. We estimated that for the training with different learning rates it would have took more than 24 hours.

We planned to use Keras functions to plot some visualizations of the activation layers, which is considered a good way to see what is going inside the convolutional layers and the where in the image the weights are giving more importance. This would have give us more light about what was going inside the U-Net and we could have compare it with the saturation of Linear function in the FFNN hyperparameters (see next subsection). Nevertheless, time was a limited resource and with the current computer power we couldn't afford it.

Other thoughts for efficiency programming would have been to use the *ImageDataGenerator* function of the Keras API code (Chollet et al., 2015). This is a method for data augmentation on images, and probably would have done more efficient modifications on the original images and at the same time saving time. However we realized this in the latest steps of this study. This made us think of also be able to add some stochastic noise on the pixels, but we were not sure how much this could have affected the real data because of possible high sensitivity of the features captured int he bands.

Aside of this, we thought of being able to change the number of "down-layers" in the U-Net MaxPooling processes, in order to also variate the "number of layers". Nevertheless, lately we realized that the coding of the U-Net was highly inefficient because every layer was added manually, instead of using the *Sequential* operator form Keras (Chollet et al., 2015). Another thing could have been to try even to assign at the end of the convolutions (down and up of the U shape) a fully connected hidden layers (maybe the same FFNN) to make a "CNN - FFNN hybrid awesome monster model". But still the main problem was the computational power, and thus, somehow we are satisfied to have been not able to do that.

5.2 FFNN

For the FFNN model, it was quite easy the interpretation of hyperparameetrs to use since all of the tunning was pointing towards clear tendencies, which meant reliable values to choose.

Figure 9 shows a clear and reliable result on what it concerns to learning rates and penalty values. It basically shows an accepted range of learning rates mentioned in the results section, but when we got a closer look on the most high performance regions, we established a learning rate value of 0.3, considering it a near the highest peak of performance (see Figure 9). Regarding the penalty value, it kept increasing performance every time the penalty value was reduced, cleanly indicating that even no penalty value was needed for this problem. Based on this, we choose to use no regularization methods, which means a penalty value of 0.0.

On the activation functions (see Figure 10), its clear that ReLU works as the best one because of the maximum score achieved between the activation functions and because of the low variability in the performance (more stable values). This was another reason for us to take into consideration when we choose ReLU as the activation function for the U-Net final model. The curious behavior of the Linear function seems to be the same as the one encountered for the Sigmoid function in the U-Net hyperparameters tunning (activation functions, see Figures 10 and 8). Again we see this saturation that seems to not let the model to improve the training. This might be due to saturation on the image on every iteration. With a pixel saturated value, it is possible that the convolutional layers don't get too much differences between one epoch and another, and at the end they could have keep training "white images" (saturated).

Regarding hidden layers and nodes, Figure 11 shows very clear the best combination between nodes and hidden layers for the best performance. Train set shows maximum peaks while using 4 hidden layers, with number of nodes ranging form 60 to 140. Since there is not noticeable difference between the performances ranging from 100 to 140, we decided to choose 100 as the number of nodes, also because we thought is the better trade-off between computational efficiency and performance. This gave a final architecture of 4 hidden layers for the FFNN model, each one consisting of 100 fully connected nodes.

5.3 U-Net VS FFNN

The final chosen hyperparameters to create the final models for the U-Net and the FFNN can be seen in the Table 1, and based on that we got the performance test with known and unknown data (see Figures 15 and 16) and in the same order for the U-Net (see Figures 13 and 14).

The classification models showed that some areas are classified as false positives, which is correct in terms of spectral signatures of the snow and ice. However, since not every bright pixel belongs to a glacier, it is important to consider spatial properties. The sea ice and ice melange was misclassified as glacier ice (false positive). The problem is similar: the spatial properties of the glaciers used as target for model training were not considered in the best possible way. From a geoscientific prospective, that it is not easy for any kind of algorithm to use the spatial properties of the glaciers, because unlike some other natural phenomenons (e.g. rivers, agricultural fields etc.) they don't have a pronounced unified signal in their shapes.

The use of blue band, band ratio between SWIR and Red bands and mNDWI unfortunately didn't show great results 12. With the probability of the pixel being a glacier of 0.7 only big spatial features were captured. Perhaps, we should have used a lower probability threshold. However, some snow patches are classified as glaciers. One of the reasons for that is that in Alaska, the images without any seasonal snow were used (images from September, when all seasonal snow is melted). The images used for South America (polygons *geometry7*, *geometry8*) also were acquised in September (beginning of Southern Hemisphere spring), so the fresh and seasonal snow might be present. These images were used as the only cloud-free images available during the considered time-period (corresponding to the GLIMS glacier outlines source dates), so the use of different images or the use of a synthetic cloud-free composite (e.g. minimum reflectance over the stack of images over a few years) might improve the result a lot. Use of the minimum reflectance will also allow to avoid icebergs in the fjords, seasonal snow and other unwanted noise (at the same time, using this kind of data simplifies the original goal of the project too).

Unexpectedly, the FNNN performed better than the U-Net with the Leaky-ReLU activation function 15 and 16, despite overestimating the glacier extent a lot. The probability threshold used for the binary classification problem is 0.5, so adjusting this parameter will affect the result in a better way (probably to lower it). The "unseen data" (polygons *geometry3* and *geometry4*) are located in North America, but still have the same problems (seasonal snow, cloud cover etc), as polygons *geometry7* and *geometry8*. However, the medial moraine is still not considered to be a glacier ice, which is, again, completely correct from a spectral signature prospective. Polygon *geometry3* has a lot of seasonal snow and located downstream from a glacier, but has a very little part of the glacier ice (in the South East corner) on it. The FFNN classified the glacier ice as glacier. The river (water pixels) was also misclassified, that is a known problem of remote sensing of water bodies, snow and ice. This might be avoided by considering spatial features (glaciers are usually wider, then rivers in proglacial environments).

Nevertheless, U-Net with ReLU shows a great advantage and future potential because is able to group subfeatures like the debris cover on the ice with the glacier itself, in contrast to the FFNN model (see Figure 13 and 15). The architecture allows to group (concatenate) specific features in the first convolution layers of the U-Net with more generalized fields in deepest layers of the same, associating the specific features to the generalized ones, and thus, giving the image segmentation. This allows to address the problem of feature grouping by neighbors, which is something that the FFNN model cannot handle since just care about individual pixel values rather than their neighbors relations and distributions.

Since the training of the U-Net was limited due to time (reduction from 3008 samples to 200) and the data was shuffled, the probability that the U-Net was trained with few data containing the most non-glacier features was high. A complete training with all the train planned dataset would have improved significantly the performance and the test of the model, overcoming the huge mistakes of the U-Net during test with "unknown" new data [see Figure 14]. Also due to the time and computational resources we did not manage to test, how different set of features, would have performed in our out task. We believe, that the chosen features (or band, band ratios, indexes) might not be the optimal combination, but that can only be tested duruing an experiment.

6 Conclusions

U-Net, or Convolutional Neural Networks, in general, is a method that could be quite effective for image segmentation and image classification because it can be able to extract geometrical patterns and features from the image, which brings a clear advantage in opposition to the simple Feed Forward (Fully Connected) Neural Networks. However, the computational time that it takes is quite high.

In most of the training it took more than 12 hours with a normal PC (16GB RAM and 8 CPU cores of 3.4GHz) and a little bit less or equal in a sever called *wessel* (256GB RAM and 32 CPU cores with unknown speed for us). PyTorch (Paszke et al., 2019) has functions to activate GPU usage for processing, which makes the CNN training more faster as the graphics card can host processes and memory usage. However, we didn't had access to independent GPU card for computational speed improvement. Time and deficit in computational power were the main reasons for not considering to do more things or even to process more data. Our Data Augmentation methods were capable of significantly increased the number of available data for training. Nevertheless, due to computational power we were not able to dispose of it in the plenty way that we wanted, and thus, affecting the performance of the models during the training. More available material resources would have improved the models accuracy significantly.

For the final tests of the U-Net we were planning to use 3008 images after the data augmentation process. However, due to time and computational power, we ended up using just 200, which is significantly less than what was planned originally. We must mention again that some of the original images contain more non-glacier features than glacier ones, and thus, increasing the non-glacier objects in the dataset in contrast to interesting features. Also, final dataset was shuffled multiple times and then the first 200 were selected. This could have meant a substantial feeding of non-glacier images to the U-Net as training dataset, and in that way affecting the further performance of the model. This is why we think that a plenty computational power would have help us to increase the performance of the model until perceiving an astonish classification.

The Leaky ReLU final test for the U-Net showed very bad solutions, which just justifies even more the trust on ReLU as a default activation function for image analysis, and specially for image segmentation problems.

Due to computational time and mistakes in the code, we didn't managed to produce a proper U-Net classification using the ReLU method, instead of the Leaky ReLU. We were fooled by the Leaky ReLU performance from Figure 8 and we took it as the best results, but somehow during the test classification it didn't performed well. This just outstands the mentioned ReLU reliability on image classification and its preference on image classification problems as discussed before.

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Figure 17: Conclusions? (Sourse: xkcd.com)

As we learned working on two previous projects, machine learning algorithms have a huge potential, when they are applied in a smart way. There are some bottlenecks, such as data size and computational time expenses. However, the general principal of all modeling also applies to Neural Networks of any kind: when the input data is not good enough, the result can not be any better (also known as "crap in - crap out"). Machine Learning algorithms allow to avoid this problem a little bit (because of the complexity of the training procedure), but the one should be very careful with that 17.

We should conclude, that it is relatively "easy" (at least, easier) to make a decent model with sufficient results, when working with so-called "vanilla datasets", that are huge, accurate and perfectly organized. It is not like that with "the real" data, that requires a lot of organizing, pre-processing, structuring etc.

From a geoscientific prospective, any kind of machine learning methods should be used carefully, when describing the processes of a natural kind, since they are considered to be a "black box" and do not describe physical processes, just the numerical connections and correlations. So just stearing the pile of equations and cracking numbers is not enough. Despite these algorithms work relatively well and are really powerful, the one should be careful interpreting the results, When trying to understand the causations and natural phenomenons.

7 Code Availability

The code for this report can be found: `https://github.com/Doctus5/FYS-STK4155/tree/main/project3`, project3 folder (last access last access 18.12.2020). The folder includes the test runs and the 'readme' file.

The code to obtain the data can be found at
`https://code.earthengine.google.com/ba7916acba15c18d38c633319da51b92` (last access 17.12.2020). The code to obtain the GLIMS glacier outlines can be found at
`https://code.earthengine.google.com/b07204f19023f099527e9b38d12d4266` (last access 17.12.2020).

References

(2009). Summary of current radiometric calibration coefficients for landsat mss, tm, etm+, and eo-1 ali sensors. *Remote Sensing of Environment*, 113(5):893 – 903.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bazilova, V. and Díaz, S. (2020). Classification and regression? insights in linear regression, logistic regression and neural networks.

Chollet, F. et al. (2015). Keras.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., and Moore, R. (2017). Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202:18 – 27. Big Remotely Sensed Data: tools, applications and experiences.

Hall, D. K., Riggs, G. A., Salomonson, V. V., Barton, J., Casey, K., Chien, J., DiGirolamo, N., Klein, A., Powell, H., and Tait, A. (2001). Algorithm theoretical basis document (atbd) for the modis snow and sea ice-mapping algorithms. *Nasa Gsfc*, 45.

Hock, R., Rasul, G., Adler, C., Cáceres, B., Gruber, S., Hirabayashi, Y., Jackson, M., Kääb, A., Kang, S., Kutuzov, S., Milner, A., Molau, U., Morin, S., Orlove, B., Steltzer, H., Allen, S., Arenson, L., Baneerjee, S., Barr, I., Bórquez, R., Brown, L., Cao, B., Carey, M., Cogley, G., Fischlin, A., Alex, d. S., Eckert, N., Geertsema, M., Hagenstad, M., Honsberg, M., Hood, E., Huss, M., Zamora, E. J., Kotlarski, S., Lefeuvre, P.-M., Juan Ignacio, L. M., Lundquist, J., McDowell, G., Mills, S., Mou, C., Nepal, S., Noetzli, J., Palazzi, E., Pepin, N., Rixen, C., Shahgedanova, M., McKenzie, S. S., Vincent, C., Viviroli, D., Weyhenmeyer, G., Sherpa, P. Y., Weyer, N., Wouters, B., Yasunari, T., You, Q., and Zhang, Y. (2019). *High Mountain Areas. In: IPCC Special Report on the Ocean and Cryosphere in a Changing Climate*. The Intergovernmental Panel on Climate Change (IPCC).

Lin, G. and Shen, W. (2018). Research on convolutional neural network based on improved relu piecewise activation function. *Procedia computer science*, 131:977–984.

McFeeters, S. K. (1996). The use of the normalized difference water index (ndwi) in the delineation of open water features. *International journal of remote sensing*, 17(7):1425–1432.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Paul, F., Winsvold, S. H., Kääb, A., Nagler, T., and Schwaizer, G. (2016). Glacier remote sensing using sentinel-2. part ii: Mapping glacier extents and surface facies, and comparison to landsat 8. *Remote Sensing*, 8(7):575.

Pfeffer, W. T., Arendt, A. A., Bliss, A., Bolch, T., Cogley, J. G., Gardner, A. S., Hagen, J.-O., Hock, R., Kaser, G., Kienholz, C., et al. (2014). The randolph glacier inventory: a globally complete inventory of glaciers. *Journal of Glaciology*, 60(221):537–552.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Vaughan, D., Comiso, J., Allison, I., Carrasco, J., Kaser, G., Kwok, R., Mote, P., Murray, T., Paul, F., Ren, J., Rignot, E., Solomina, O., Steffen, K., and Zhang, T. (2013). *Observations: Cryosphere*, book section 4, page 317–382. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

8 Supplementary material

North America (Alaska):

1. LANDSAT/LT05/C01/T1_TOA/LT05_066017_20070917
2. LANDSAT/LT05/C01/T1_TOA/LT05_066017_20090906

South America (Chilie):

1. LANDSAT/LT05/C01/T1_TOA/LT05_230095_20070914
2. LANDSAT/LT05/C01/T1_TOA/LT05_232095_20070912
3. LANDSAT/LT05/C01/T1_TOA/LT05_232095_20070912
4. LANDSAT/LT05/C01/T1_TOA/LT05_232094_20070912
5. LANDSAT/LT05/C01/T1_TOA/LT05_232095_20070912

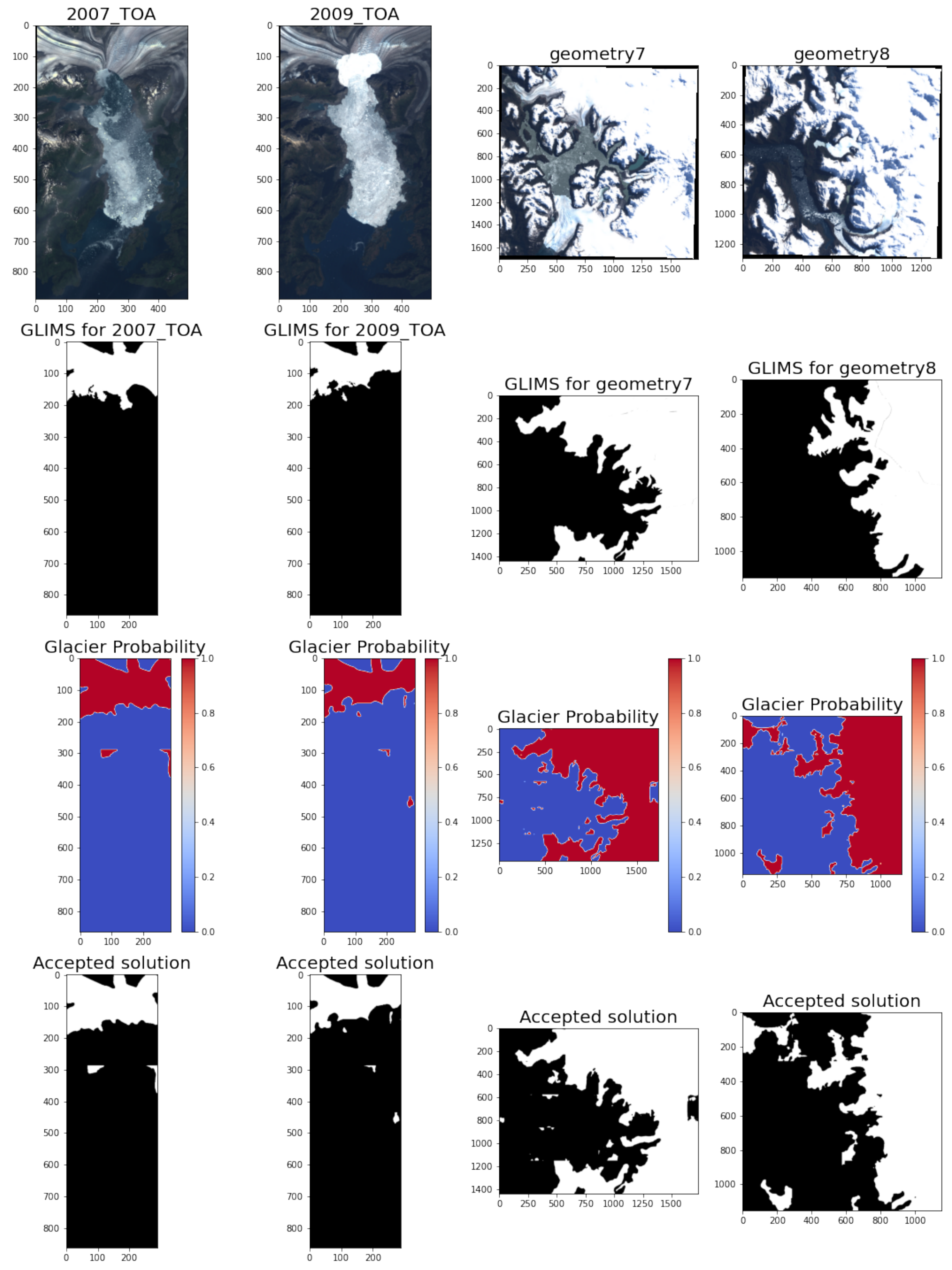


Figure 13: Test performance of the final Unet model with hyperparameters chosen from the best results of the tunning process. The data tested was the same trained data.

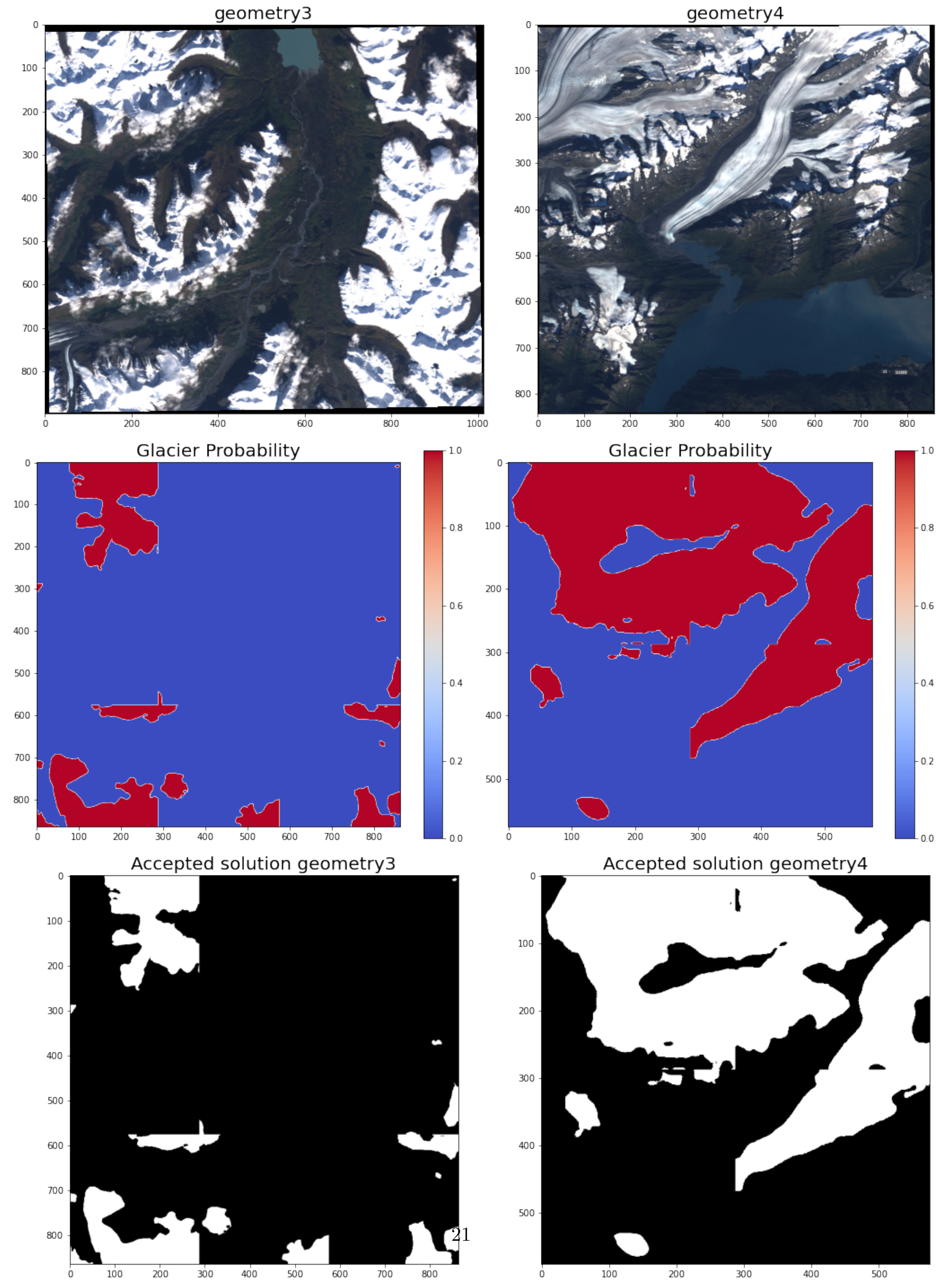


Figure 14: Test performance of the final U-Net model with hyperparameters chosen from the best results of the tunning process. The data tested was new unknown data with no solution downloaded.

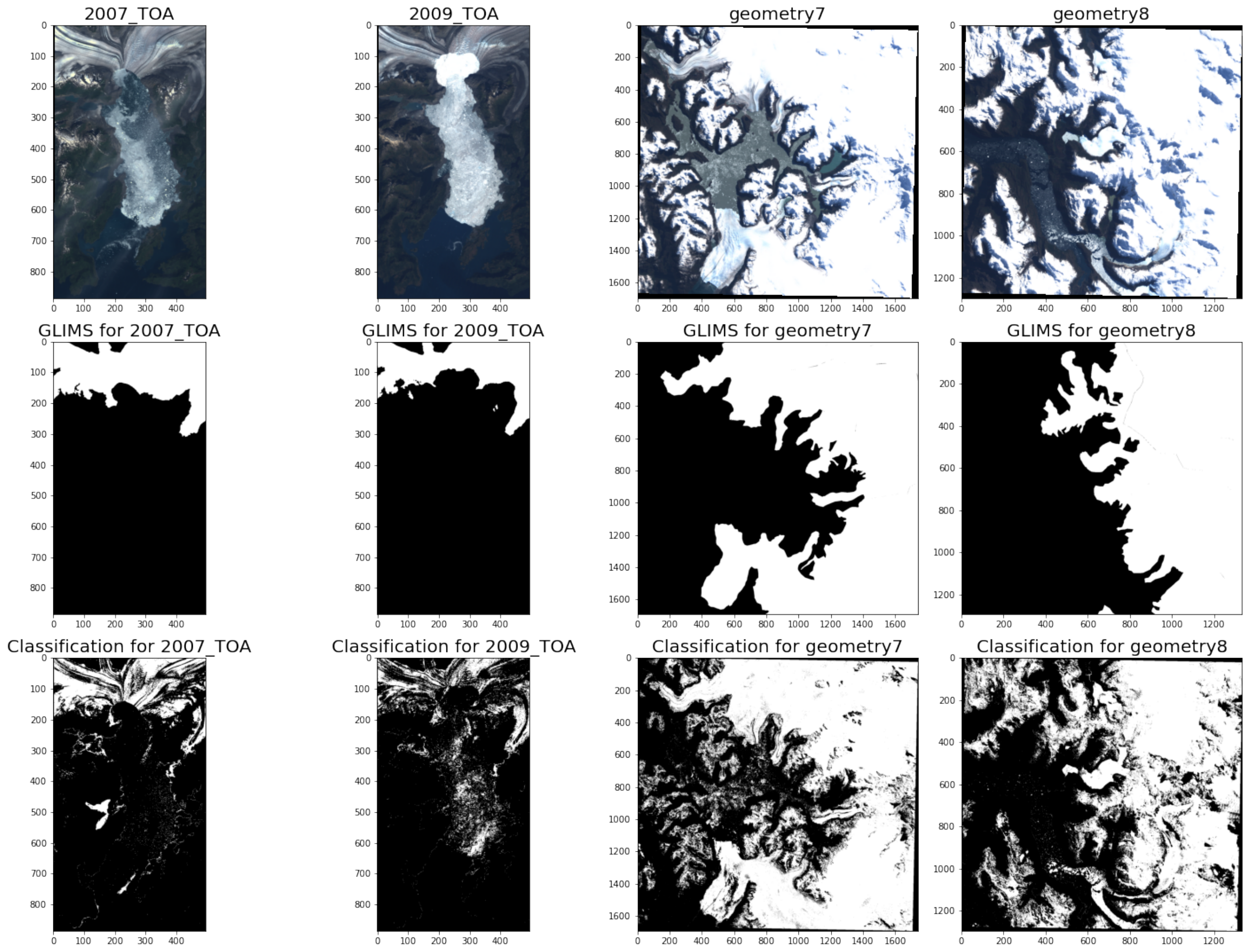

2007_TOA
2009_TOA
geometry7
geometry8
GLIMS for 2007_TOA
GLIMS for 2009_TOA
GLIMS for geometry7
GLIMS for geometry8
Classification for 2007_TOA
Classification for 2009_TOA
Classification for geometry7
Classification for geometry8

Figure 15: Test performance of the final FFNN model with hyperparameters chosen from the best results of the tunning process. The data tested was the same trained data.

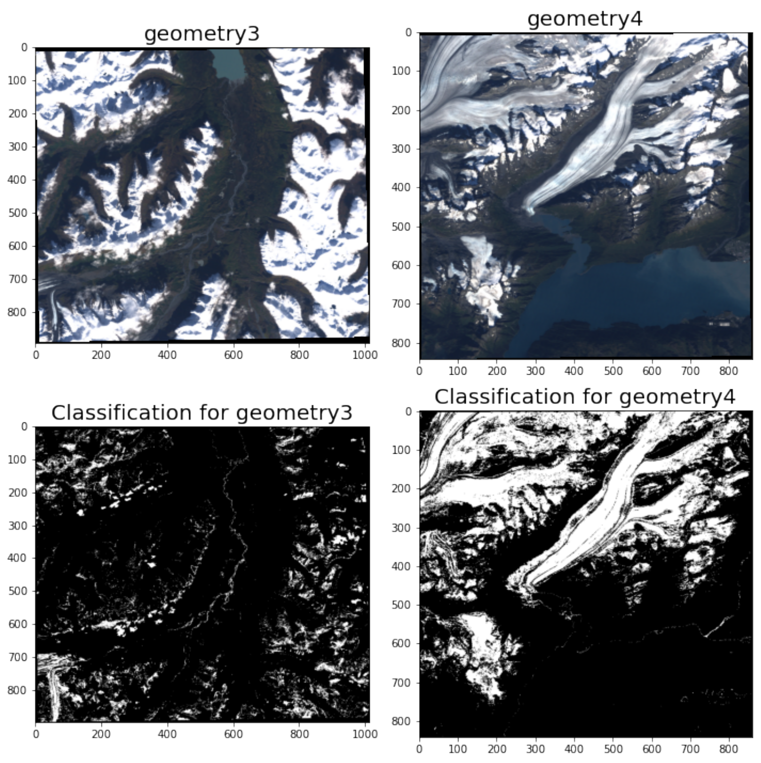


Figure 16: Test performance of the final FFNN model with hyperparameters chosen from the best results of the tunning process. The data tested was new ”unseen data”