A Large Scale Analysis of Mountain Glacier Shrinkage Using Convolutional Neural Networks

Matthew Waismann, Somansh Budhwar, and Armin Schwartzman

***Abstract*—Mountain glaciers are undergoing widespread shrinkage, with consequences for fresh water supplies and contributions to sea level rise. While large inventories of glacier surface areas exist, these datasets are temporally sparse, and traditional methods of collecting glacier data, such as ground-based measurements or manual analysis of satellite imagery, are not scalable for continuous global monitoring. With recent advancements in computer vision techniques and the growing availability of remote sensing data, it is now feasible to programmatically analyze the extensive archives of satellite images from missions such as Landsat. In this work, we develop a scalable, programmatic framework for globally estimating the area of mountain glaciers from satellite images over time. After applying our novel data pre-processing and filtering methods, the main engine is a U-net deep Convolutional Neural Network (CNN) with a modified Resnet encoder. The CNN is trained on 7174 glacier contours and their corresponding images, each at only one time point, and then applied to time series of satellite images of 1083 glaciers, comprising a total of 505835 images, to obtain time series of estimated surface area for those glaciers over the 40-year period 1984-2023. Our global analysis confirms a consistent trend of glacial surface area shrinkage in all five major glacierized regions: Asia, the Caucasus, Europe, North America and South America. We estimate an overall glacier surface area decline of $0.2\% \pm 0.01\%$ per year.**

***Index Terms*—Remote sensing, Landsat, digital elevation models, image segmentation.**

I. Introduction

MOUNTAIN glaciers play a crucial role in the cryosphere, providing fresh drinking water to a significant portion of the global population, particularly in regions near the Himalayas, Andes, Rockies, Alps and Karakoram mountain ranges [8]. However, glaciers around the world are shrinking and the consequences extend beyond the local environments and populations. Glacier shrinkage is a major contributor to sea level rise, with studies estimating that 18–24% of the global increase is attributable to glacier mass loss. These doing so largely as a result of anthropogenic climate change. Between 1991 and 2010, 45–93% of global glacier mass loss has been attributed to human activity [7]. Furthermore[5].

Historical efforts to globally quantify glacier mass have faced significant challenges. Glacier inventories such as the Randolph Glacier Inventory (RGI) and the World Glacier Inventory (WGI) are good starting points, providing valuable, unified datasets that include point-in-time glacier surface area measurements and glacier outlines [1]. However, these datasets are temporally sparse, glacier outlines are prone to human annotator error, and the quality of these annotations vary significantly. In addition, traditional ground-based or manual methods for collecting glacier measurements are not scalable for continuous global monitoring. As a result, comprehensive global analyses of glacier decline remain limited [6].

With recent advancements in computer vision techniques, including Convolutional Neural Networks (CNNs), and the growing availability of distributed computing resources, it is now feasible to programmatically analyze the extensive archives of remote sensing data from missions such as Landsat. In this paper, we leverage these technologies to analyze (be more specific, detailed, action words, about what we've developed (e.g. code, data, ML architecture) and why its novel) glacier surface area changes at scale, producing high-resolution temporal estimates of glacier extent worldwide.

Our global analysis reveals a consistent trend of glacial surface area decline across all five major glacierized regions: Asia, the Caucasus, Europe, North America and South America. We estimate an overall glacier surface area decline of $0.2\% \pm 0.01\%$ per year.

II. Data Description

A. Overview

In order to effectively train a machine learning model for the task of glacier segmentation and make meaningful inferences on glaciers, across the array of available satellite and other remote sensing data sources, time points, and worldwide locations, we built an extensive and highly curated data product across various data sources. We leveraged the Global Land Ice Measurements from Space (GLIMS) database for identifying glaciers and obtaining human-annotated glacier outlines, United States Geological Survey (USGS) Landsat satellite image data and metadata vended through Google Earth Engine for glacier imagery with high spatiotemporal resolution, NASA-provided digital elevation models (DEMs) from the Shuttle Radar Topography Mission (SRTM) and our own custom-derived image features. These datasets were used to support exploratory data analysis, machine learning model training, inference, and our analysis and impact

B. GLIMS Database

The GLIMS database provides a comprehensive list of glacier outlines for over 328,115 glaciers worldwide. GLIMS has been a collaborative effort that has involved more than 60 institutions since its establishment in 1999 (glims.org). In particular, the Randolph Glacier Inventory (RGI) project,

M. Waismann is with Amazon Inc., New York, NY, 10001 USA e-mail: ???.

S Budhwar is with ???.

A. Schwartzman is with the Halıcıoğlu Data Science Institute at the University of California, San Diego, La Jolla, CA 92093 USA e-mail: armins@ucsd.edu

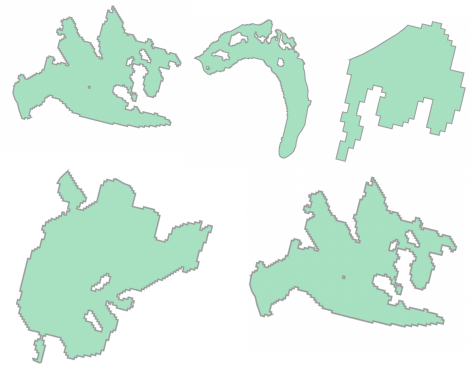

Fig. 1: Five sample glacier outlines from GlobGlacier

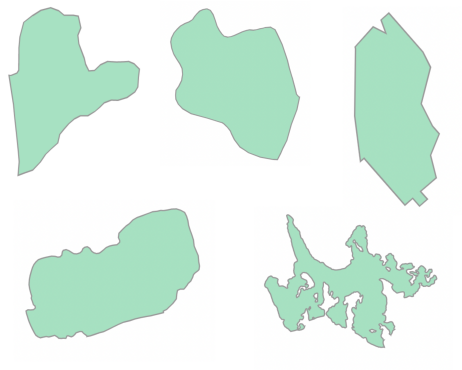

Fig. 2: Five sample glacier outlines from the Randolph Glacier Inventory

which represents 173,023 glaciers in GLIMS, has made a significant contribution. However, RGI prioritized achieving complete coverage over extensive documentary detail during its 1-2 year completion period (RGI paper citation). Other well-funded projects, like GlobGlacier, supported by the European Space Agency, have curated a carefully selected set of glacier outlines.

To ensure the highest-quality outlines for training our segmentation model, we filtered the GLIMS dataset down to 18,093 glaciers. Firstly, we selected outlines drawn after 2005 without a specific reason for this choice. Furthermore, we excluded glaciers from the Arctic and Antarctic GLIMS regions, such as Svalbard and the Antarctic Peninsula. This filtering process resulted in a dataset primarily comprising outlines from the Himalayas, European Alps, Western North America, the Andes, and the Tien Shan – Pamir – North Karakora mountain complex. GLIMS also provides surface area estimates for these glaciers, allowing us to systematically sample regions with a bias towards larger glaciers. Specifically, we selected glaciers with a surface area between 1 km^2 and 50 km^2. While the majority of outlines were for smaller glaciers, opting for larger glaciers provides two strategic advantages: (1) Larger glaciers are associated with larger satellite images, facilitating easier shape capture by the segmentation model, given the fixed size of the satellite images, and (2) The model should be well-acquainted with the largest glaciers, as they have the highest impact. Although many glaciers are larger

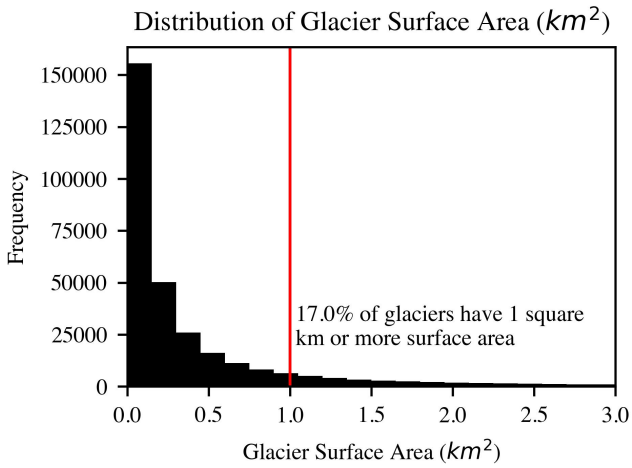

Fig. 3: Distribution of Glacier Surface Area in GLIMS

than 50 km^2, we set it as an upper bound to filter out outliers and potential human annotation errors.

C. Google Earth Engine: Images

To create the ground truth masks for our training data set, we defined bounding rectangles for each glacier outline, adding an additional 10% of spatial extent along each of the two dimensions. These bounding rectangles also defined the spatial extent of our training data inputs. Using the Google Earth Engine Python API, we exported images for the 18,093 glaciers from Google Earth Engine using the GLIMS-derived bounding rectangles. To allow for more sophisticated preprocessing of our training data images, we retrieved all Landsat 5, 7 and 8 images captured within an even 100-day window of the GLIMS attribute SRC_DATE, defined as the as-of date for the outline, usually the acquisition date of the source image. The specific Google Earth Engine image collections used were LANDSAT/LT05/C02/T1_L2, LANDSAT/LE07/C02/T1_L2, and LANDSAT/LC08/C02/T1_L2, representing Landsat image collection 2, Level-2, Tier 1 data. Landsat collection 2 is the second major reprocessing effort on the Landsat archive by the United States Geological Survey (USGS) and as of December 30th, 2022, its predecessor, Landsat collection 1, is no longer available for download from the USGS. Level-2 data represent an additional layer of processing on top of Level-1 scaled Digital Number data, providing atmospherically corrected, unitless, surface reflectance data which is more appropriate for time series analysis [might need to find a source]. Of the Level-2 data, Tier 1 scenes represent the highest quality of data and are considered suitable for time series analysis [USGS]. Since these data are Tier 1 and represent various time points with various Landsat satellite overlap and availability, the amount of Landsat images collected for in this 100-day window varies for each bounding rectangle. (INSERT A HISTOGRAM or SUMMARY STATISTICS HERE SHOWING THE DIFFERENT NUMBER OF IMAGES WE HAVE). Additionally, we obtained Digital Elevation Models (DEMs) for each bounding rectangle through Google Earth Engine's

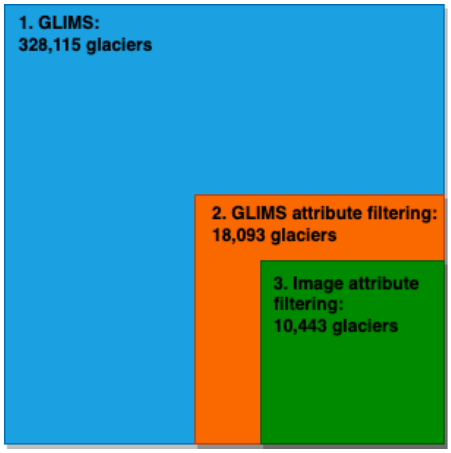

Fig. 4: Training dataset size

API, utilizing NASADEM, a NASA data product for global elevation modeling.

D. Image Metadata

In addition to the images themselves, USGS and Google Earth Engine provide a rich set of metadata which we extensively processed to facilitate further post-processing and filtering. This metadata included fields such as cloud cover over the image and land portions, and image quality scores (1-9). The new Landsat Collection 2 data also provided land characterization bits that offer metadata on snow/ice confidence. The satellite images and DEMs were saved as GeoTIFF files, a format supporting georeferencing information embedded in raster images.

III. Data Pre-Processing

Before training, the data underwent extensive preprocessing. As the spectral bands differ slightly across the Landsat satellites, the selected bands have similar wavelength ranges but are not an exact match. Since we had access to all satellite images captured within a 100-day window around the annotated polygon date, we averaged the images across time using an arithmetic mean. This approach helped smooth out shadows, debris, and defects. To ensure compatibility with TensorFlow, we reshaped the images and masks to 128 x 128 pixels and normalized the pixel values to the unit interval. Additionally, we computed the Normalized Difference Snow Index (NDSI) and Normalized Difference Water Index (NDWI) from the existing bands, following their universal formulation. As a result, the training set comprised 128 x 128 masks and ¡n¿ 128 x 128 images with seven spectral bands, a DEM band, and NDSI & NDWI bands. We read the GeoTIFF rasters and polygon shapefiles into the Python runtime environment using the rasterio library. Since the rasters were defined in the UTM reference system while the polygons were defined in the WGS 84 reference system, we reprojected each raster to the WGS 84 reference system using rasterio. With matching reference systems, we obtained ground-truth masks by aligning the satellite images and polygon masks using rasterio. These raster images, comprising seven spectral bands, DEMs, and ground-truth masks, constitute the training set for the CNNs.

A. Data Pre-processing for the model

¡OUTDATED¿ Firstly, the dataset consisted of masks that were missing segmentation, so they had to be filtered out. We calculate the variance of each mask, and if variance was near zero, we discarded that mask and the corresponding band images of that GLIMS id. Consequently, the training corpus comprised a total of around 17,795 images and corresponding masks.

Secondly, the seventh band of Landsat-7 images were found to be corrupted, which might degrade the learning and predictive capabilities of the convolutional neural network (CNN). But since they might offer partial information about the segmentation, they were kept in the training data.

Finally, each band was normalised using min-max normalisation such that each pixel in a band has value ranging from 0 to 1. This helps during the training phase of the model. The 17,795 images were further partitioned into a training set (16015 images), and the test set (1780 images). ¡OUTDATED¿

B. Inference Data Selection

The selection for inference is determined by following SQL query identify_inference_glims_ids_geog_area_rollup_50.sql which runs over our AWS Athena metadata database. This query works by scanning over all of our training data metadata for GLIMS IDs that have at least one file with cloud_cover < 10, image_quality = 9, more than 50,000 pixels, 0 no data pixels, less than 10% zero pixels, and have geographic area that's not Canada (since those glaciers all appear on Ellesmere Island and are too far north for DEMs), and has an image in a low snow month (all of them do). Then we will take 50 GLIMS IDs per our 5 geographic area rollups based on db_area. In September 2024, we can an effort to expand the since of our inference dataset. We took the existing query and extended it to take 250 glaciers per of the 5 geographic area rollups. This time, when downloading the images from the Earth Engine API, we decided to push certain filters down to the server, so we could limit the amount of glaciers we're downloading which would only later get filtered out in our metadata filtering step.

C. Pre-inference Filtering

Before applying our U-Net for our semantic segmentation task on our extracted satellite images, we have a metadata-based filtering step where we only run inference for images which meet our criteria for inference. This criteria is applied via a SQL query (Github file link here) and the actual filtering happens before the images read into local memory with rasterio.

Filter criteria:

- cloud cover: less than 10% of the image
- summer months only: May-October in the northern hemisphere and November-April in the southern hemisphere
- number of pixels $>$ 50000
- we should break down the 1,000,000 ways this is a dumb thing to do
- no data pixel count = 0
- percentage_zero_pixels ¡ 0.1

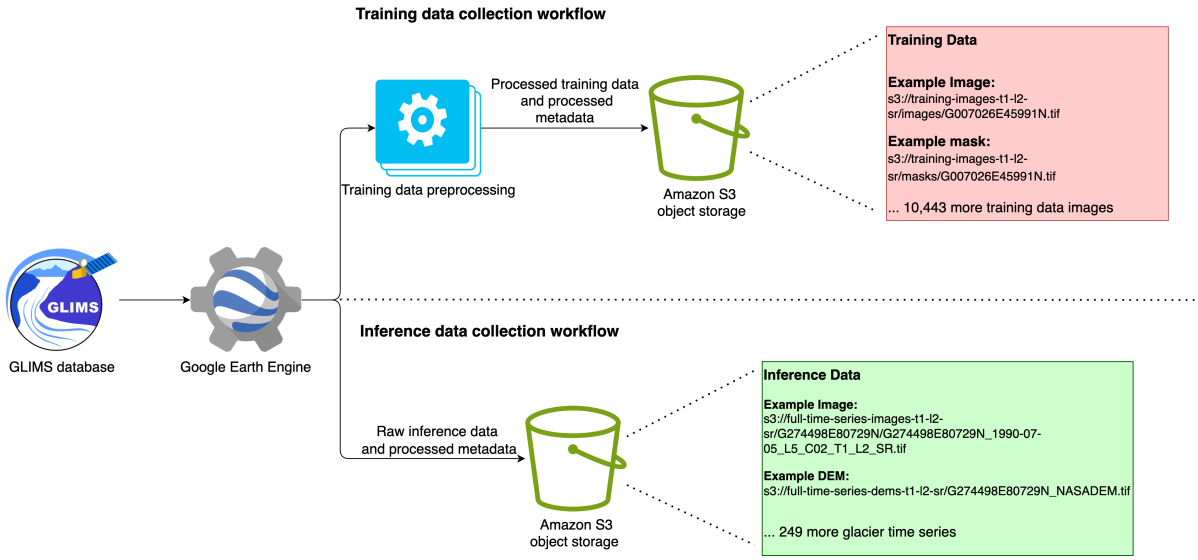

Fig. 5: Training and inference data collection workflow

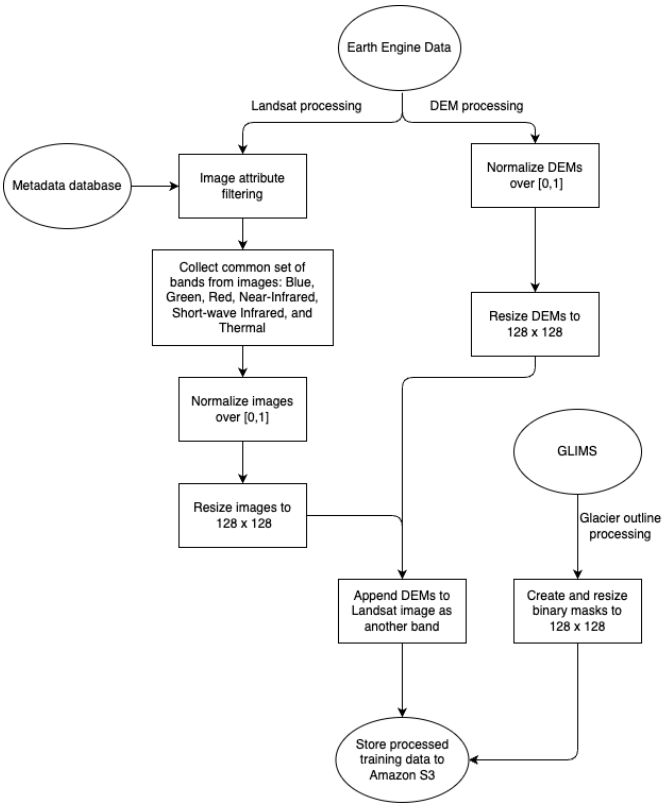

Fig. 6: Training data processing workflow

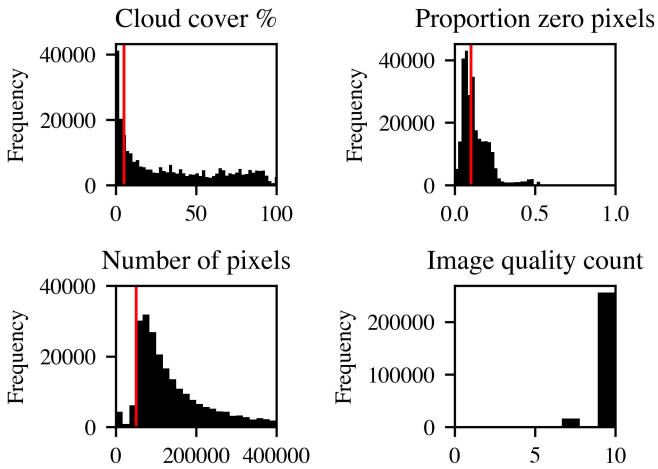

Fig. 7: Distribution of Landsat image metadata with the following filters: at most 5% cloud cover, at most 10% zero-value pixels, at least 50,000 total pixels (across all bands), and 10 as the image quality score. The sub-panel x-axis is the same as the title.

D. Metadata Collection

We persist the metadata provided by earth engine. Earth engine provides these metadata via the Image object's get-Info() method. This call returns a JSON blob of metadata for each of the individual images being downloaded (this is done iteratively, at the same time as download time). At the same time as we download these images, we persist these JSON blobs in a list per GLIMS ID per Landsat satellite (tech debt - we could've been a lot cleaner with how we store this metadata - the script which relationalizes these metadata has to navigate a strange file structure)

Example folder structure of list of of JSON blobs:

- landsat/
- /G007068E45470N
- /G007068E45470N_1984-04-18_L5_C02_T1_L2_SR.tif
- ...
- /G007068E45470N_2024-01-01_L8_C02_T1_L2_SR.tif
- /meta_data/
- /.../metadata_list_l5
- /.../metadata_list_l7
- /.../metadata_list_l7

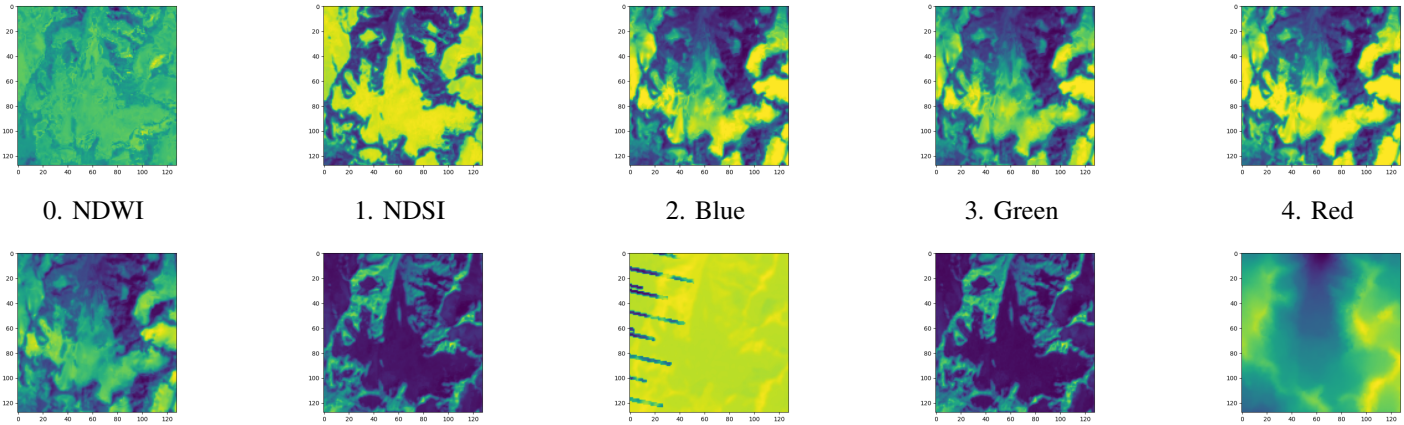

Fig. 8: The 9 channels used in the CNN. Channels 2-7 were captured from Landsat while channels 0-1 were calculated using other bands. *** Increase font

*** Remove SWIR2 because of quality.

- / ... other GLIMS IDs

Earth engine may sometimes provide more than one metadata record per image (having two different cloud cover values for the same image is a common scenario that we see), so we perform deduplication, choosing one of the records arbitrarily to ensure we have one record per file name.

In addition to the metadata provided by Earth Engine, we also generate our own metadata. As each image is downloaded from earth engine, we collect logs for the purpose of monitoring the download process. Note, these metadata are really only helpful for monitoring download progress because all of the metadata fields provided here come directly from the Earth Engine provided metadata.After every image has been downloaded. We iterate over each downloaded file, collect metadata about the file itself and open the image and collect metadata about the image itself.

The time_series_metadata_handler.ipynb file processes (and collects in the case of the image and file attributes) all of our metadata into relational CSV files, where each record belongs to a single file (file name primary key). These CSV files are then uploaded to Amazon S3, registered in AWS Glue as individual data tables for easy consumption with a tool like AWS Athena.These files were uploaded to s3://full-time-series-metadata-t1-l2-sr and crawled over by the full-time-series-metadata-t1-l2-sr-crawler in us-west-1.

IV. Image Segmentation Model

Since this work aims to perform image segmentation, a U-Net architecture was used. The U-Net architecture has been shown to perform well in image segmentation tasks [10]. The basic U-Net consists of two parts. First, an encoder condenses the image into a latent representation of the input image. Second, the decoder up-scales the latent representation to recreate the segmented image. To help the segmentation task, the corresponding layers from the encoder are also combined with each up-scaled layer, which is referred to as adding "skip connections" [9]. The overall architecture is shown in Fig. 9 and described in detail next.

A. *Encoder*

The encoder section is a modification of a model called Resnet-50. Resnet-50 is a multi-layer image segementation model pre-trained on a diverse image dataset with complex shapes, patterns and features [4]. The advantage of using a pre-trained model is that the weights are pre-trained to identify low-level features commonly found in images such as edges and corners. The pre-trained model thus constitutes a good starting point for obtaining a good representation of images of glaciers with less training time.

Two modifications were made to Resnet-50 according to our situation. First, the original Resnet-50 model takes a 3-channel input while our dataset has a 10-channel input. Therefore, we replaced the first layer of Resnet-50 with a 10-channels input custom layer with randomly initialized weights to be updated during training. Second, since we need to use Resnet-50 as an encoder and not as a classifier, its last two layers were removed. The resulting modified Resnet-50 encodes the 10-channel input, each of size 128×128, as a 2048-channel output, each of size 4×4. The full encoder is depicted on the left hand side of Figure 9 and can be described as follows.

The encoder takes in 10 channels as the input, each of size 128×128. It begins with a 2D Convolution layer with 64 filters. Each filter has a kernel size of 7×7 pixels, stride of 2×2 pixels, and adds a padding of 3 pixels outside the margins of each image. These dimensions are chosen such that the dimensions of the output feature map from the first layer align with the dimensions required by the upcoming Resnet layer. The output is then batch normalized, passed through a ReLU activation function and then Max Pooled. It is then passed to the four major modules of Resnet-50 which are shown in Fig 10. The first module consists of 3 sequential sub-modules where each sub-module consists of 3 convolution layers. Similarly, the following modules consist of 4, 6 and 3

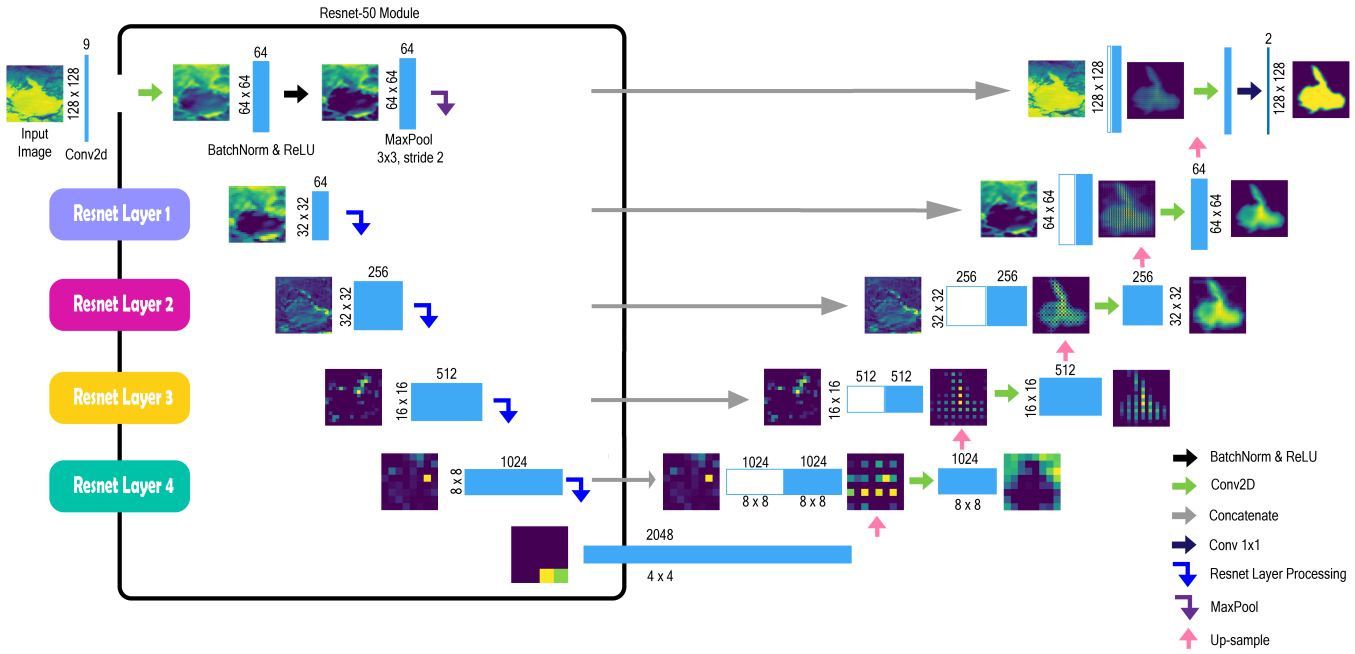

Fig. 9: CNN U-Net Architecture: encoder, including the modified ResNet architecture (left), and decoder (right).

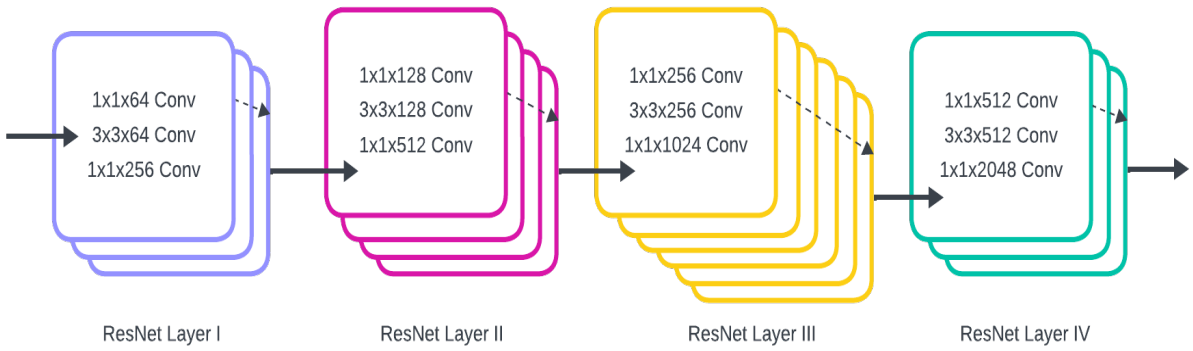

Fig. 10: ResNet-50 architecture used in the encoder part of the CNN architecture depicted in Figure 9. The four colors indicate the match between the four ResNet modules depicted here and those in the encoder depicted in Figure 9.

sub-modules each. After processing, the encoder outputs 2048 channels of 4×4 images. These channels contain the latent representation of the original satellite image bands.

B. Decoder

The decoder section (right half of Fig. 9) involves upsampling the 2048 channel input we obtained from the modified Resnet-50 into a 2-channel image of size 128×128. The two channels represent the probability of a pixel belonging to the class 'glacier' or the class 'background'. Although a single channel would suffice, having two channels makes it technically easier to train the model using the standard PyTorch library, which expects the number of output channels to be equal to the number of classes being predicted. In one output channel, each pixel contains the probability of it being a glacier, and similarly in the other output channel, each pixel contains the probability of it being the background. In this work, one output channel is the complement of the other.

In the first step of upsampling, the decoder deconvolves the 2048 channels to 1024 channels of size 8×8 and then applies ReLU activation and batch normalization to it. In the subsequent step, the decoder concatenates the upsampled channels with the corresponding channels of the encoder as shown in Fig. 9. This concatenation is called a skip connection, and it helps provide the semantic information from the encoder that may be lost during subsequent encoding. This is akin to using the original image underneath a tracing paper to draw accurate outlines. The concatenated layers are then passed through a convolution layer to reduce the channel, and again an activation function and batch normalization are applied. Subsequently, we get the 2-channel output we desire.

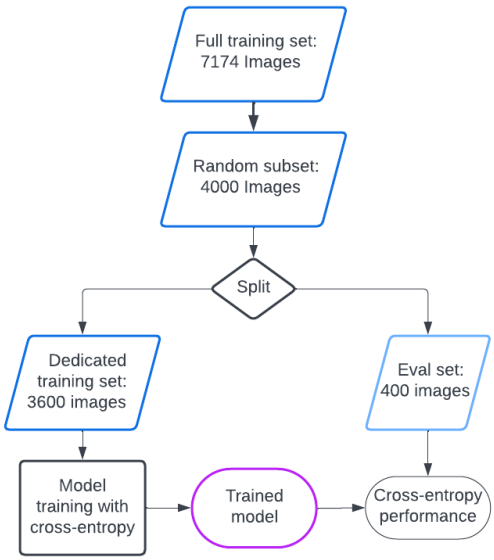

Fig. 11: Hyper-parameter tuning flowchart

V. Model Training

A. Loss function

Training of the model was implemented using cross-entropy loss. While the objective is binary image segmentation, cross-entropy loss is a softer loss function in the sense that it does not simply penalize right or wrong pixel classification, but penalizes more if there is a higher probability difference in the classification. This score is appropriate for training with blurred images, as in our case, where the distinction between ice and ground could be unclear. Since most images exhibit glacier pixels and background pixels in a similar proportion, there was no severe class imbalance and other loss functions such as focal loss were not necessary. The performance of the final model was evaluated using the Dice score, which is more easily interpretable as a measure of the segmentation accuracy of the model.

B. Hyper-parameter tuning

To begin training, we must first find the appropriate hyper-parameters for the model. To tune these hyper-parameters, a subset of 4000 images were selected at random from the full training set of 7174 glacier images (Figure 11). Of those 4000, 90% were used to train the model and the remaining 10% to evaluate its performance.

Table I shows the cross-entropy loss for various combinations of the learning rate and batch size. As can be seen in the table, reducing the learning rate had an important impact on performance but not so the batch size. Based on the smallest cross-entropy score obtained, the learning rate was set to 0.00001 and the batch size to 32.

As additional checks on hyper-parameters, we found that a slightly higher Dice score was obtained by unfreezing the Resnet encoder weights, as opposed to freezing the original pre-trained weights, and keeping the output classification threshold to the default of 0.5.

Learning rate	Batch size	Cross-entropy loss
0.001	16	3.35
0.0001	16	0.68
0.00001	16	0.50
0.00001	8	0.98
0.00001	8	0.53
0.00001	**32**	**0.49**

TABLE I: Hyper-parameter tuning: Loss for each hyper-parameter combination on the same data.

C. Data Augmentation

To improve model training, the training dataset was augmented by adding different transformations to each image in the training phase. Each image had one of three transformations randomly applied to them: vertical flip, horizontal flip and spatial blur. That is, each input image and its corresponding target image were either flipped vertically, flipped horizontally, or spatially blurred with a Gaussian kernel. Augmenting the training data by these transformed images allows the model fitting to adapt more generally to glaciers of different spatial orientations and images of less quality.

While all three transformations could be applied to each image, this would quadruple the training set, substantially increasing computational training time. Instead, for each image, each of the transformations was applied with a probability of 0.5. This means a probability of 0.125 of no transformation being applied, 0.375 probability of one transformation being applied, 0.375 probability of two transformations being applied, and 0.125 probability of all three transformations being applied. This process thus increased the training set size by only 75%, while still being richer in the desired manner. The spatial blur consisted of convolution with a 3×3 pixel Gaussian kernel with standard deviation 0.8, which is the default in the TensorFow function tf.gaussian_filter2d.

D. Training procedure

To perform the training of the model, the full training set of 7174 glacier images was randomly split into three subsets of sizes 90%/10%/10% of the total (Figure 12). The larger subset of 6456 images was augmented according to the procedure described above resulting in a proper training set of 11298 images.

Following the hyper-parameter tuning results, the training was carried out using a learning rate at 0.00001 with Adam optimizer and batch size of 32, with unfrozen encoder weights. The 10% evaluation set was used to track the training. The final trained model was chosen as the model that minimized the cross-entropy loss on the evaluation set during the training procedure.

VI. Inference

*** This paragraph belongs in Data Description We performed inference on time series of satellite images for 505835 images for 1083 glaciers. Each time series for each glacier has a different number and distribution of time points. Some glaciers have more than 1000 images while others have less

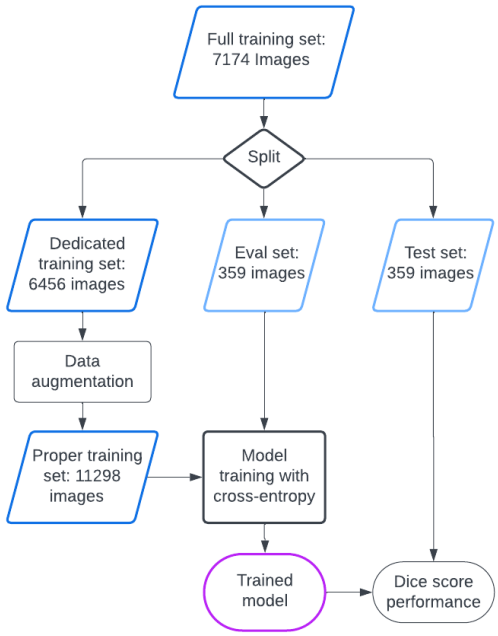

Fig. 12: Model Training Flowchart

than 500. The distribution of number of images per glacier in the inference set is shown in Figure 13(left). The observations are not equally spaced over time, with sparser data during the 1990s and more data available at later times. The number of glaciers with available images per year for inference is shown in Figure 13(right). Note that these images are filtered using criteria *** (Put filters in table after verifying with Matt) such as low cloud cover, high image quality, low zero pixel count, etc. (Verify with Matt the exact filters we used.) *** check this

A. Inference procedure

The inference process is shown in Fig. 14. For each glacier, we sort the available images according to their time stamps. Then, resize each image using Pytorch's resize function which interpolates the image to 128×128 pixels because the CNN requires the images to be of these dimensions. Next, we apply pixelwise Gaussian smoothing over time to reduce noise. This is implemented as a pixelwise time Gaussian filter with standard deviation equal to 20 time points. After smoothing, we input the image to the CNN and segment the glacier. Next, each image is resized to its original size.

As shown in Figure 9 and described above, the output of the CNN decoder at each pixel is a number between 0 and 1, interpreted as a probability of that pixel belonging to the glacier. Figure 15 shows an example of the soft mask segmentation obtained by the CNN. The soft segmentation is turned into a hard segmentation by thresholding the output at 0.5, so the segmented image consists of pixels that are either a background pixel (0-0.5) or a glacier pixel (0.5-1). The area of the glacier in pixels at any given time point is obtained by simply adding the values of the thresholded binary image. Since the resolution of Landsat images is 30 meters by 30 meters, each pixel measures 900 meters squared. Thus the true area of the glacier in kilometers squared is obtained by multiplying the pixel area by $900/(10^6)$.

VII. Results

A. Model evaluation metric

For this model, we chose to use Sørensen–Dice coefficient, also known as Dice score [3], as an evaluation metric. The reason we choose this metric is two-fold. Firstly, the Dice loss results in marginally better models compared to the Jaccard loss in image segmentation tasks [2]. Secondly, the Jaccard index has the union of sets as the denominator, while the Dice score has the sum of sets as the denominator, which does not change during training, thus providing a simpler loss function during training.

B. Example output: Trient glacier

For the Trient glacier, we obtain 464 images at various time points from 1984-04-18 to 2023-10-07. These images were obtained from Landsat-5 and Landsat-7 images by applying the following filters. Cloud cover should be less than 20, Percentage of zero pixels in the image should be less than 15%, Image quality should be greater than or equal to 9 (out of 10), no data pixels (NaN values) should be zero, and number of pixels in the image should be greater than 50,000. Next, we extract the following bands from each satellite image, 'blue','green','red','nir','swir','thermal'. Then we resize each image to 128x128 pixel because this is the size our model was trained to process. Note that we save the original size of the image for resizing the output for area prediction, and since original image size varies marginally between Landsat-5 and Landsat-7, we only choose one size. Since the images are still prone to errors, and missing information, we apply Gaussian Smoothing with Sigma 20 to the series of 464 images. In other words, each pixel is smoothed normally based on neighbouring pixels at different time points. Finally, we calculate the NDSI and NDWI of each of the smoothed image to create our inference dataset.

Each image in the inference dataset is then passed through the model to get the segmented image. Each segmented image is then resized to the original size. Finally, to calculate the area we simply sum all the pixel values in the image and multiply it with $900/(10^6)$ since each pixel is 30x30 meters. So we get the area of the glacier in kilometers squared.

As a case example we show results for the Trient glacier in Switzerland (G007026E45991N). The available data corresponding to the Trient glacier consists of 464 images at various time points from 1984-04-18 to 2023-10-07. The estimated area using our trained CNN is plotted over time in Fig. 16.

As seen in the graph, the total area of the Trient glacier shows an overall decline over the last four decades. However, the decline is not uniform over time, with most of the decline occurring in the periods 1985-1988 and 2002-2015. The glacier appears to have been stable over the last ten years. To better understand this behavior spatially, Fig. 17 shows satellite images of Trient in 1995 and 2005, overlaid with the soft segmentation output from the trained CNN. The results

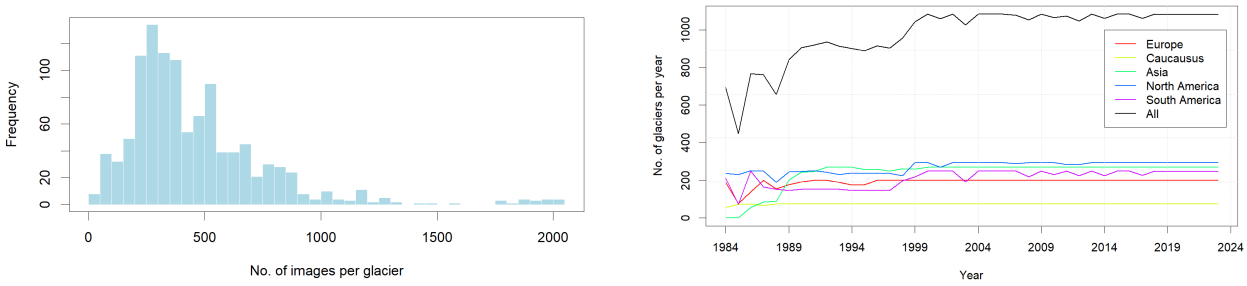

Fig. 13: Summary of inference set. Left: Number of images per glacier. Right column: Number of glaciers per year.

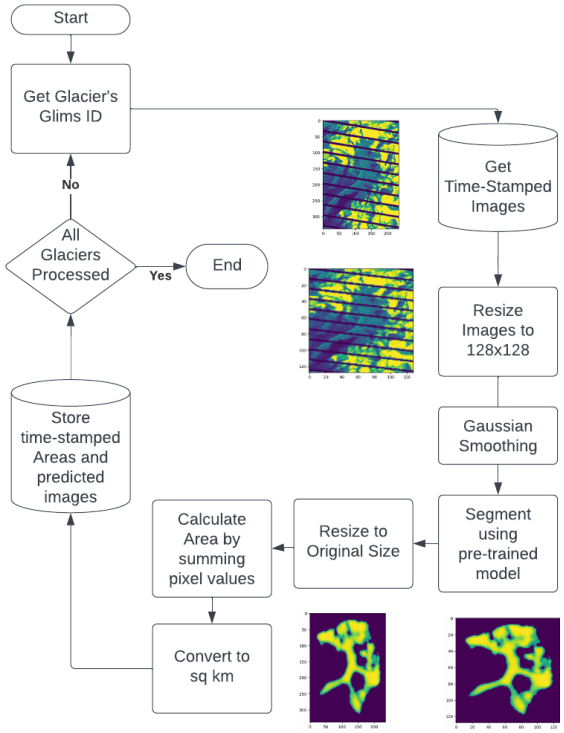

Fig. 14: Inference: Area estimation flowchart

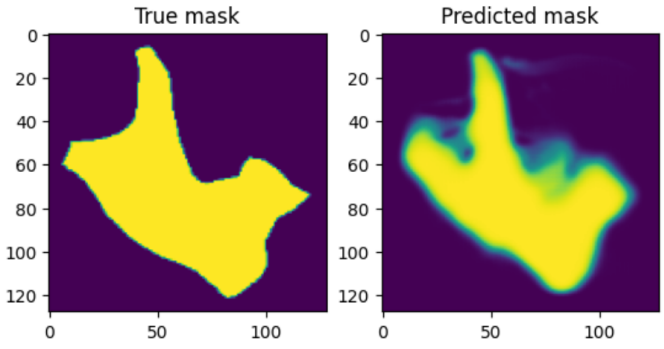

Fig. 15: Sample ground truth and output: Actual mask on the left versus Predicted mask with soft boundaries on the right of the G099490E34798N glacier.

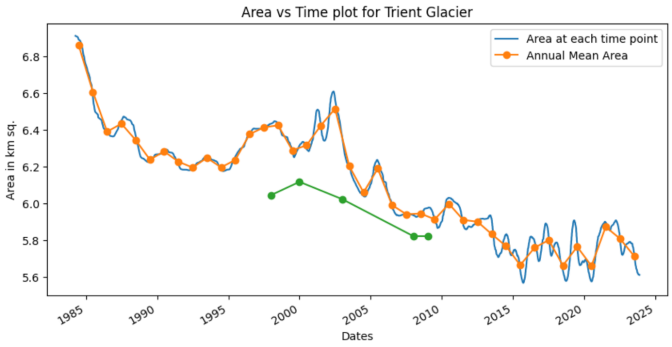

Fig. 16: Trient Area prediction plot including annual mean areas.

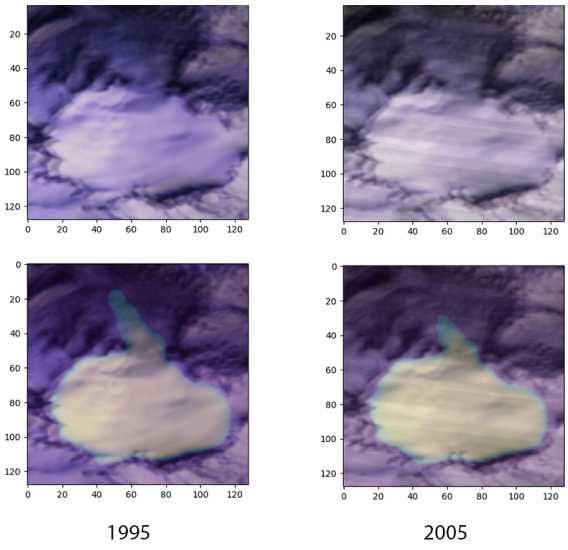

Fig. 17: Top: Trient satellite images on 1995 and 2005. Bottom: Trient overlayed area prediction on 1995 and 2005

show that the algorithm is not only able to capture the northern terminus of the glacier despite the lack of illumination. The time comparison shows that it is precisely near the north terminus where the ice is being lost.

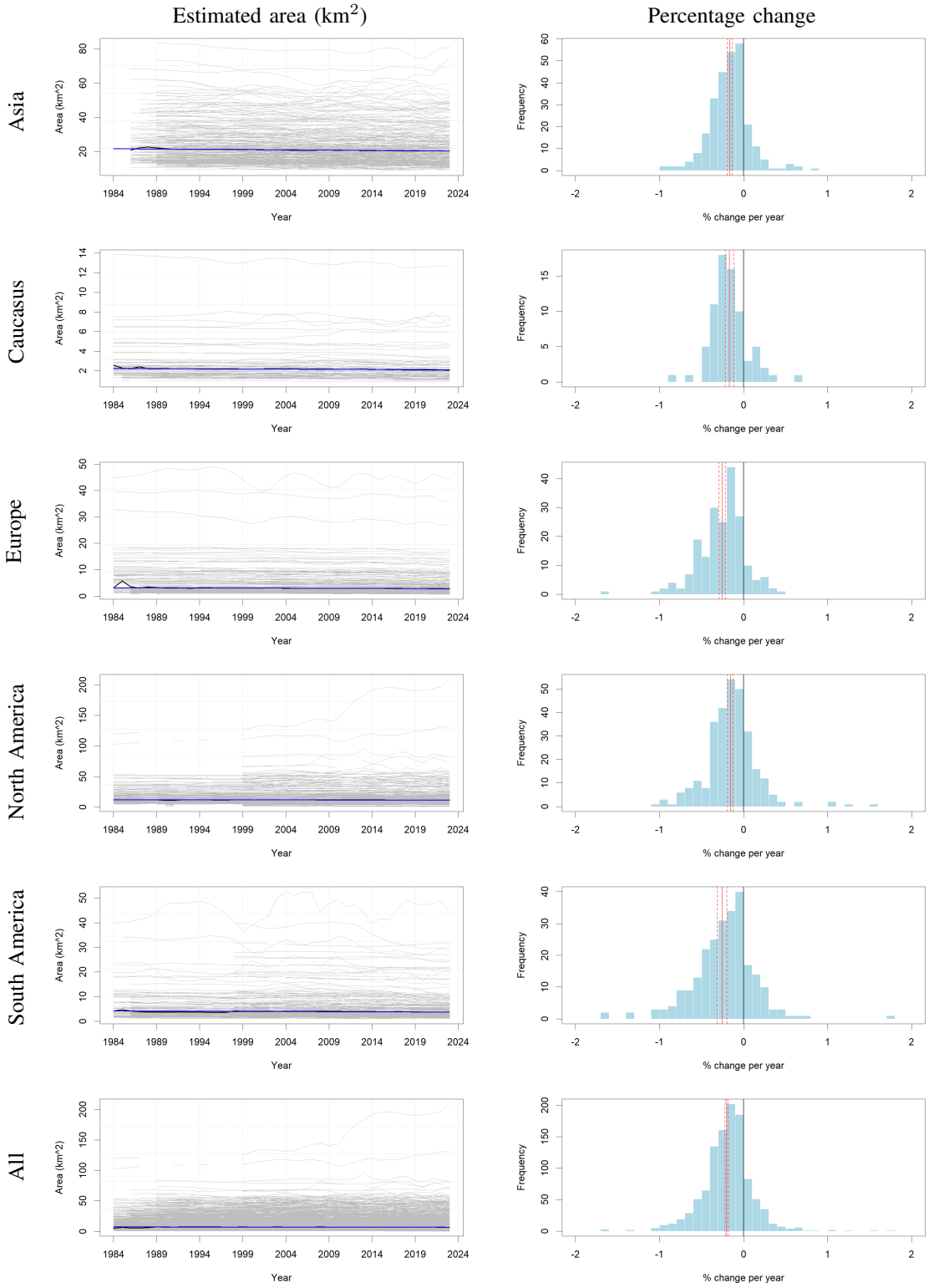

Fig. 18: Summary of results by geographical region. Left column: Estimated area for individual glaciers (gray), pointwise average (black), regression line (blue). Right column: Histogram of area percentage change per year for individual glaciers (blue), weighted average (red) plus/minus two standard errors (dashed red).

Region	Relative area change	Standard error	Relative std. error
Europe	-0.00256	0.00019	0.075
Caucasus	-0.00167	0.00025	0.149
Asia	-0.00163	0.00015	0.090
North America	-0.00155	0.00017	0.110
South America	-0.00256	0.00029	0.115
Total	-0.00199	0.00010	0.049

TABLE II: Summary of analysis results by region. Left column: Estimated relative area change, fitted by log-linear regression. Middle column: Standard error on the relative area change, estimated from the regression. Right column: Relative standard error (middle column divided by left column).

C. Results by region

The estimated area of all glaciers analyzed in this study is plotted in the left column of Figure 18, grouped by five major geographical regions: Asia, Caucasus, Europe, North America, and South America. The graphs show the estimated time series over the period 1985-2023. Notice that the data is sparser in earlier years, especially before 1990, due to lack of Landsat images during that early period.

Since glaciers are of different sizes and rates of change, averages using absolute values in km squared would be dominated by the largest glaciers. Instead, to make results comparable between glaciers, we estimate the relative change. This is done in two different ways via linear regression, as follows.

The first method fits a log-linear regression model (see Appendix **??**) to the time series corresponding to each glacier, resulting in an estimated relative change over the entire period 1985-2023 for each glacier. These results are shown as histograms in the right column of Figure 18. For all five geographical regions, the majority (about 2/3) of glaciers exhibit a relative reduction in area, although some glaciers exhibit a relative increase in area.

The average of the relative changes is a weighted average, where the value for each glacier is weighted by the number of valid observations associated with it: glaciers with more observations are given higher weight. A 95% confidence interval is shown for the average within each region. The weighted averages for all five regions are negative, indicating glacier shrinkage, and are statistically significant, as their associated confidence intervals do not cover the value 0.

The second method computes the geometric average of all glaciers within a region pointwise at each time stamp, shown in the left column of Figure 18 with a solid black line. The geometric average is a log-linear average, analogous to log-linear regression. Then, a log-linear regression (see Appendix **??**) is fitted to the geometric average time series. The slope of that line is the relative change shown in the right column of Figure 18 in red.

VIII. Conclusion

In this paper we created a dataset of 10,443 glaciers and trained a UNet convolutional neural network to segment the glacier in the image. We achieved a Dice score of 0.92 on the test set. Using the model we applied it to segment certain glaciers and using the segmented image we estimated the area of the glacier. Based on the the inference of Trient glacier we note that its area is declining, especially at the tail of the glacier, and there was a noticeable drop in the area after the year 2000.

So, image segmentation offers a great way to estimate the area of the glaciers on a global level. However, as we have seen in this paper, the model and the predictions are only as good as the data. The scan line corrector failure errors, cloud cover, image quality and other factors affected the training of the model and the inference. With a better resolution and clearer data a better performance can be expected. Moreover, with the advent of Attention mechanisms, a UNet architecture with Attention gates can also be explored.

Appendix A
Log-linear regression for estimating relative change

A. Log-linear regression for a single glacier

Let $(t_1, y_1), \ldots, (t_n, y_n)$ be a time series of n measurements $y_1, \ldots, y_n$, in this case surface area estimates, at time points $t_1, \ldots, t_n$. Note that the time points do not need to be equally spaced. The proposed log-linear model poses the relationship

$$\log(y_i) = \beta_0 + \beta_1 t_i + e_i, \qquad i = 1, \ldots, n. \tag{1}$$

Assuming that the measurement errors are independent between time points and have the same variance, this model can be fitted by ordinary least squares, yielding the fitted equation

$$\log(\hat{y}_i) = \hat{\beta}_0 + \hat{\beta}_1 t_i, \qquad i = 1, \ldots, n. \tag{2}$$

Then the ratio between two measurements y_i and y_j at any two time points t_i and t_j is given by

$$\frac{\hat{y}_j}{\hat{y}_i} = \exp\left(\log(\hat{y}_j) - \log(\hat{y}_i)\right) = \exp\left[\hat{\beta}_1(t_j - t_i)\right].$$

If the coefficient $\hat{\beta}_1$ is small in absolute value, as it is in our case, then a first-order Taylor approximation of the exponential gives that the relative change is approximately

$$\frac{\hat{y}_j - \hat{y}_i}{\hat{y}_i} = \frac{\hat{y}_j}{\hat{y}_i} - 1 \approx \hat{\beta}_1(t_i - t_j).$$

Therefore, the coefficient $\hat{\beta}_1$ represents the relative change in the measurement per unit of time. For example, a value of $\hat{\beta}_1 = -0.02$ represents a relative reduction in surface area of 2% per year.

B. Log-linear regression for regional averages

The regional surface area averages shown in black in the left column of Figure 18 are modeled similarly to the log-linear model (1), except that the measurement errors cannot be assumed to be homoscedastic because the average at each time point is computed from a different number of observations. Assuming the measurement errors from different glaciers to be independent, the average at each time point has a variance that is inversely proportional to the number of glaciers. The regression is fitted in the log domain similar to the description in Section A-A except that the average at each time point is weighted by the number of observations that constitute that average.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] A Arendt, T Bolch, JG Cogley, A Gardner, JO Hagen, R Hock, G Kaser, WT Pfeffer, G Moholdt, F Paul, et al. Randolph glacier inventory [v2. 0]: A dataset of global glacier outlines, global land ice measurements from space, boulder colorado, usa. *Digital Media*, 2012.

[2] Jeroen Bertels, Tom Eelbode, Maxim Berman, Dirk Vandermeulen, Frederik Maes, Raf Bisschops, and Matthew B. Blaschko. *Optimizing the Dice Score and Jaccard Index for Medical Image Segmentation: Theory and Practice*, page 92–100. Springer International Publishing, 2019.

[3] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Romain Hugonnet, Robert McNabb, Etienne Berthier, Brian Menounos, Christopher Nuth, Luc Girod, Daniel Farinotti, Matthias Huss, Ines Dussaillant, Fanny Brun, et al. Accelerated global glacier mass loss in the early twenty-first century. *Nature*, 592(7856):726–731, 2021.

[6] Matthias Huss and Regine Hock. Global-scale hydrological response to future glacier mass loss. *Nature Climate Change*, 8(2):135–140, 2018.

[7] Ben Marzeion, J Graham Cogley, Kristin Richter, and David Parkes. Attribution of global glacier mass loss to anthropogenic and natural causes. *Science*, 345(6199):919–921, 2014.

[8] Alexander M Milner, Kieran Khamis, Tom J Battin, John E Brittain, Nicholas E Barrand, Leopold Füreder, Sophie Cauvy-Fraunié, Gísli Már Gíslason, Dean Jacobsen, David M Hannah, et al. Glacier shrinkage driving global changes in downstream systems. *Proceedings of the National Academy of Sciences*, 114(37):9770–9778, 2017.

[9] A. Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities, 2018.

[10] Jian Wu, Wanli Liu, Chen Li, Tao Jiang, Islam Mohammad Shariful, Hongzan Sun, Xiaoqi Li, Xintong Li, Xinyu Huang, and Marcin Grzegorzek. A state-of-the-art survey of u-net in microscopic image analysis: from simple usage to structure mortification, 2022.