Automated Detection of Eskers in Hillshaded Digital Elevation Models Using Convolutional Neural Networks

MWITONDI, Kassim and STORRAR, Robert <http://orcid.org/0000-0003-4738-0082>

Automated Detection of Eskers in Hillshaded Digital Elevation Models Using Convolutional Neural Networks

Centre for High Performance Computing (CHPC) National Conference 2020
Machine Learning and its Applications in High Performance Computing
30th November – 02nd December 2020
Pretoria, South Africa (Online Event)

Kassim S. Mwitondi (PhD)
k.mwitondi@shu.ac.uk
https://www.shu.ac.uk/about-us/our-people/staff-profiles/kassim-mwitondi
https://uk.linkedin.com/in/kassim-mwitondi-9602091b

Robert D. Storrar (PhD)
r.storrar@shu.ac.uk
https://www.shu.ac.uk/about-us/our-people/staff-profiles/robert-storrar
https://www.researchgate.net/profile/Rob_Storrar

Presentation Outline

1) Background
   i. Climate-sensitive processes that influence global and regional sea levels
   ii. Impact of sea level rise and its implication on climate change, poverty and food security
   iii. Study motivation, problem area, research question and objectives
   iv. Climate-sensitive processes influencing sea levels uncertainty
   v. Tracking footprints of ice sheets movements, eskers
   vi. Role of satellite imaging & machine learning in tracking sea level uncertainty

2) Methods
   i. Data Sources
   ii. General Implementation Strategy
   iii. Mechanics of Convolutional Neural Networks -architecture and key parameters
   iv. Construction of a CNN model and the Sample-Measure-Assess (SMA) algorithm

3) Analyses, results and discussions
   i. CNN Model Training, Validation and Testing in Python
   ii. Predicting presence and/or absence of esker Patterns in imagery data
   iii. Addressing global issues - data randomness, data openness, interdisciplinarity

4) Concluding remarks
   i. Revisiting the objectives
   ii. Challenges, opportunities and suggestions for potential new research directions

Abstract

Sea-level rise constitutes a significant risk for over 600 million people in the Low-Elevation Coastal Zone. Considerable uncertainty exists over the magnitude of possible future sea-level rise, because of poorly understood processes governing the stability of ice sheets (continental sized glaciers). One such uncertainty is how meltwater interacts with ice under a warming climate. Understanding of this process is limited by the inaccessibility of the subglacial zone, which lies beneath 100s or even 1000s of metres of ice. One approach to address this uncertainty is to investigate areas where ice sheets have retreated, i.e., where their beds are easily accessible. Eskers are landforms that record the location and dimensions of former subglacial meltwater channels, and are common in glaciated regions. Recent years have seen a dramatic increase in the availability of high-resolution Digital Elevation Models (DEMs) of glaciated regions, providing the opportunity to make detailed measurements of eskers from remotely sensed data. Manual mapping of these features at the required level of detail is not feasible over the large areas occupied by palaeo-ice sheets (e.g. most of Canada). We propose an automated method for detecting eskers in hillshaded digital elevation models, based on Convolutional Neural Networks (CNN). The automated method maps esker locations to facilitate detailed morphometric study of their form. Multiple CNN models are trained and tested via a specially–designed algorithm with built–in mechanism for selecting an optimal model. Training and testing imagery data were obtained from a test area in Canada, consisting of 1041 esker positive JPEG files and 37000 esker negative JPEG files. The CNN model performance on previously unseen images with and without eskers yields high sensitivity and specificity respectively and we use the model outputs to elicit esker features from the images. Discussions focus on how timely identifying esker locations enhance our understanding of why, how, and how fast the sea level rise might happen. We also highlight the importance of gaining such knowledge in a timely manner within the context of the United Nations Sustainable Development Goals (SDGs)–particularly SDG #13 and others relating to poverty and food security.

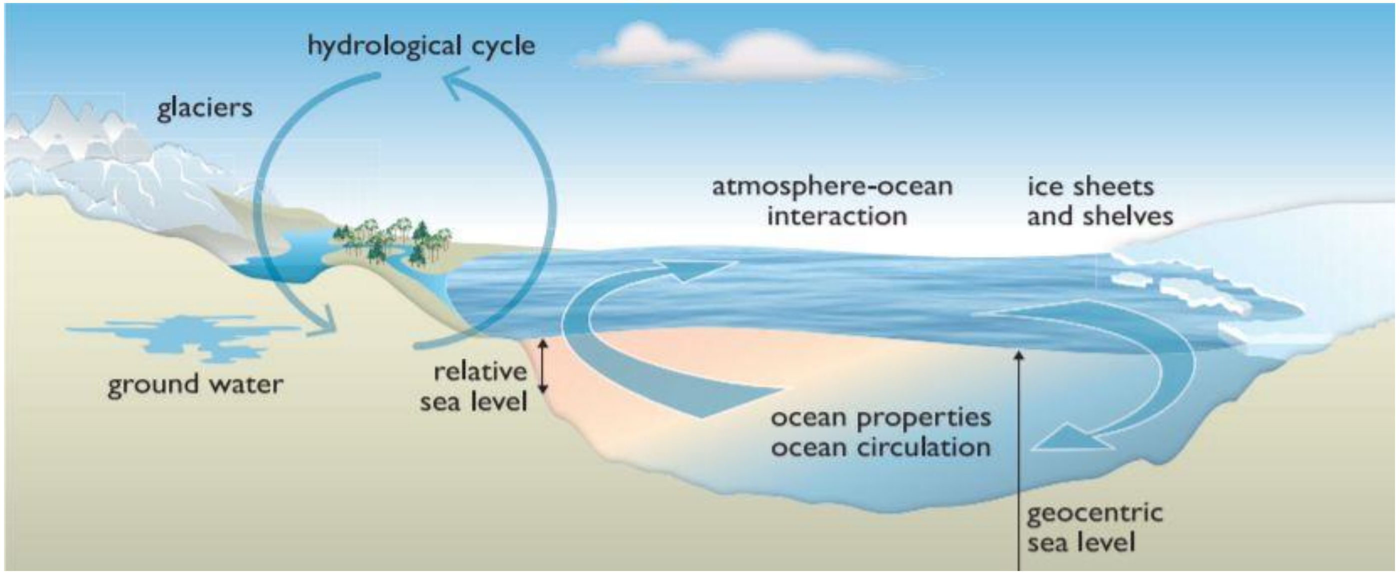
Key Words: *Big Data, Convolutional Neural Networks, Data Science, Digital Elevation Models, Eskers, Glacier Surface Elevation, Ice Sheet Hydrology, Machine Learning, Meltwater Channels, Supervised Modelling*

Background

- Approximately 10% of the world's population (> 600 million people) inhabit the low elevation coastal zone (< 10m) above sea-level [1] and are therefore vulnerable to the impacts of sea-level rise.
- Sea–level rise impacts millions of people world–wide. Understanding why, how, and how quickly the rise might happen is vital to the United Nations Sustainable Development Goals (SDG)–particularly SDG #13 (on combating climate change), SDG #1 (on poverty) and SDG #2 (on food security).
- Global mean temperature has increased by 1^{o}C since 1900, resulting in global sea-level rise of about 20cm & the Intergovernmental Panel on Climate Change (IPCC) predicts it to rise by approx. 60–74cm by 2100 [2].
- Research into sea-level changes over the glacial-interglacial cycles shows that sea-level was 5-10m higher than present [3], when temperatures were just 1-2^{o}C warmer than 1900 (i.e. comparable to the next 30-50 years). The source of this extra sea-level rise can only be from the Greenland and Antarctic Ice Sheets.
- Understanding the nature of the meltwater drainage systems beneath ice sheets is crucial to understanding their dynamics [4], and ultimately their future contribution to sea-level rise, but this remains problematic due to the difficulty of studying the inaccessible beds of modern ice sheets. This is the focus of this work.

Problem Area

- Sea–level rise has a significant impact on human habitats particularly those in the low elevation coastal zone and directly impacts on the regions' levels of SDG attainment, especially those relating to poverty and food security.
- On the left hand side-the complex interactions of the hydrological cycle, ground water and glaciers and their impact on the sea level while the right hand side shows the mutual dependence between ocean properties like temperature, salinity and density, on the one side, and ocean circulation on the other, affecting both relative and geocentric sea level, as they both vary with position.



A graphical illustration of the climate–sensitive processes and components that can influence global and regional sea levels [2]

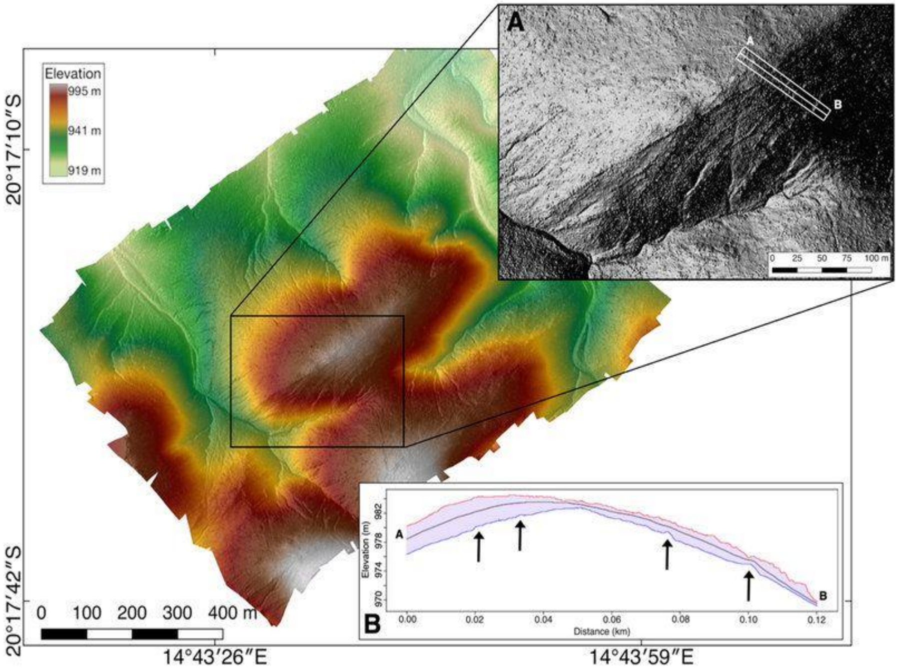
- Our understanding of how meltwater interacts with ice under a warming climate is constrained by the inaccessibility of the subglacial zone.
- One plausible approach is to investigate areas where ice sheets have retreated, which we seek to achieve via an automated mapping of high-resolution DEMs of glaciated data, using deep learning techniques and the imagery data.

Background

- Lakes on the surface of the Greenland Ice Sheet can drain under the ice, affecting ice velocity (and so delivery of water to the sea)
- Little is known about how this water moves underneath the ice because it is inaccessible
- We can make use of areas where ice sheets used to be present (e.g. Canada) to better understand how water moved underneath the ice
- Long, winding hills made of sand and gravel, known as eskers, form a record of meltwater drainage
- Measurements of eskers can be used to understand how much water and sediment moved beneath ice sheets, therefore improving our understanding of how water is likely moving beneath the modern Greenland Ice Sheet.

Background

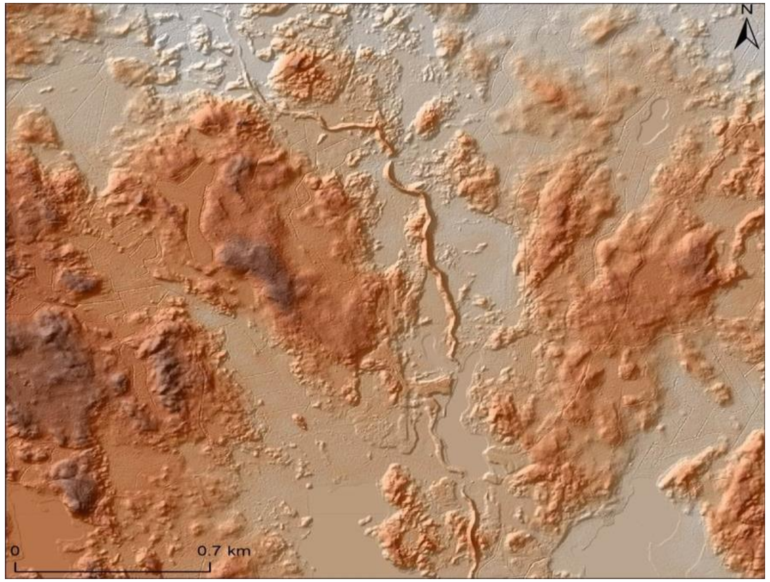
- Esker measurements in 3D have only recently been made possible, following a dramatic increase in the availability of high-resolution Digital Elevation Models (DEMs) of formerly glaciated regions, warranting detailed measurements of eskers from remotely sensed data.



- Manual mapping of these features at the required level of detail is not feasible over the large areas formerly occupied by ice sheets (e.g. most of Canada).

Background

- We investigate subglacial meltwater flow via areas where ice sheets have retreated–i.e., where their beds are easily accessible. Interest is in eskers–landforms that record the location & dimensions of former subglacial meltwater channels, commonly found in glaciated regions.
- Esker patterns and properties have been used to understand the long-term and large-scale behaviour of subglacial meltwater during periods of climate warming and ice sheet retreat [9, 7, 10, 11, 12].
- However, to date no work has systematically quantified esker dimensions in 3D at the very large scales of ice sheets, due to limited availability (until recently) of sufficiently high-resolution elevation data.
- This work takes first steps towards a systematic morphometric analysis, focusing on automatic detection of eskers from processed digital elevation data, using Convolutional Neural Networks (CNN).



Motivation

Advances in computing power and explosions in data generation, have triggered data–intensive research across disciplines, through, *inter–alia*, different applications aimed at addressing global challenges and opportunities, a typical example being the United Nations Sustainable Development Goals (SDG), as sources of Big Data [13, 14].

- By training a CNN to identify eskers based on a training set of images where eskers have been manually identified as either present or absent, we aim to be able to identify the locations of eskers over very large areas automatically.

- The increased availability of high-resolution Digital Elevation Models (DEMs) covering glaciated areas present the opportunity to extract valuable metrics on esker dimensions such as height, width and volume, which can be used to inform numerical models of subglacial hydrology. However, the volume of data available and its very high resolution mean that processing the data and mapping features is a non-trivial task.
- Previous efforts to quantify esker morphometry have used manual mapping of eskers from satellite imagery [11]. Such an approach would be too time-consuming to be practical to map features to the level of detail required for high resolution 3D morphometry over extremely large areas.
- We therefore adopt a Big Data approach [13, 14] and develop an automated process for detecting eskers in hillshaded DEMs, based on Convolutional Neural Networks (CNN).

Research Question and Objectives

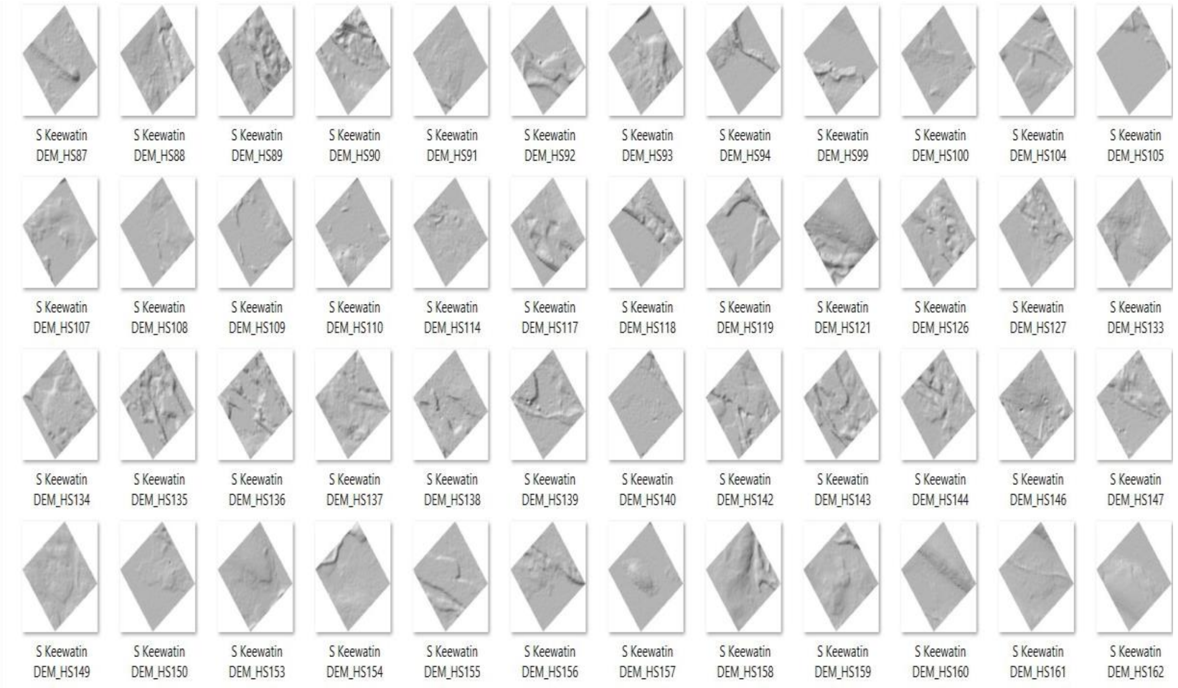
We therefore define the research problem as: **How can automatic detection of eskers enhance our understanding of rising sea levels?** The general aim of the work would then be to identify which images have eskers present and which haven't. We set the following objectives.

1) To evaluate relevant past and current studies on meltwater interactions with ice.
2) To prepare, train, validate and test high–resolution imagery data for CNN modelling.
3) To address the issue of data randomness in training, validation and testing.
4) To evaluate extracted features against real images.
5) To highlight paths for interdisciplinary research studies in addressing global challenges.

Responding to the foregoing question and fulfilment of the objectives are based on deep learning underlying rules in thousands of esker positive and esker negative imagery data using convolutional neural networks via a specially designed algorithm for handling issues relating to data randomness. The adopted methods are outlined in next exposition.

Methods: Data Sources

Training, validation and testing imagery data were obtained from a test area in Canada, consisting of 1041 esker positive JPEG files and 37000 esker negative JPEG files. The data files are Digital Elevation Models (rasters where the pixel value represents elevation), processed into a hillshade (a model of shadows based on an illumination angle, pixel values representing shadow). From these hillshades the eskers can be identified and mapped.

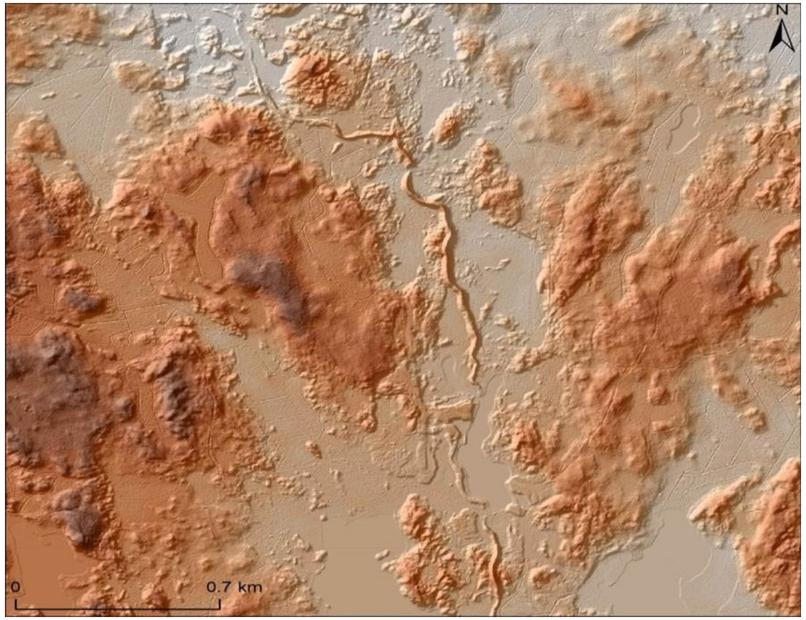


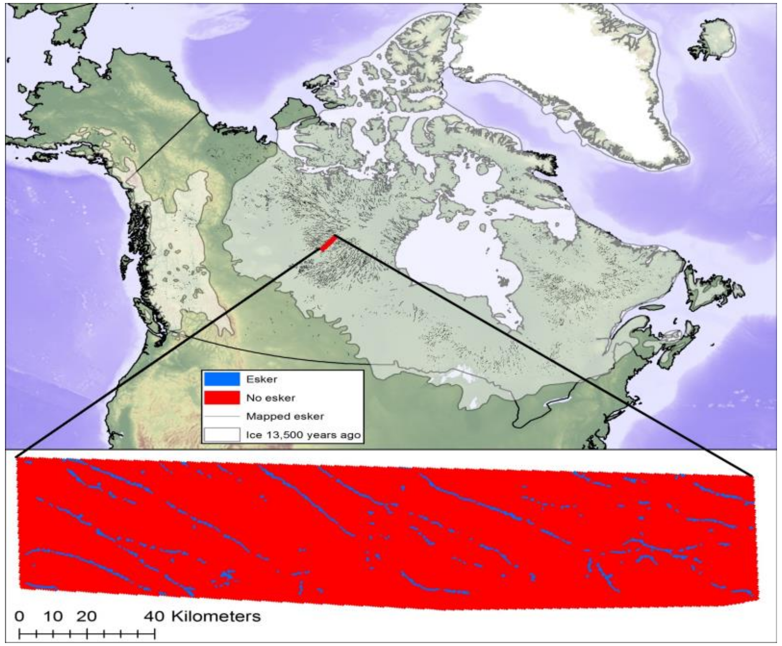
Esker-positive images

Esker-negative images

Methods: Data Sources

The chosen test area in Canada is a very large hillshade, reduced into 500m x 500m (250 x 250 pixels) JPEG files, placed into two folders depending on if an esker is present or not. The idea is to train a CNN model on these positive and negative examples in order to identify what an esker looks like, so that we could then feed it unmapped areas in order to detect patterns.

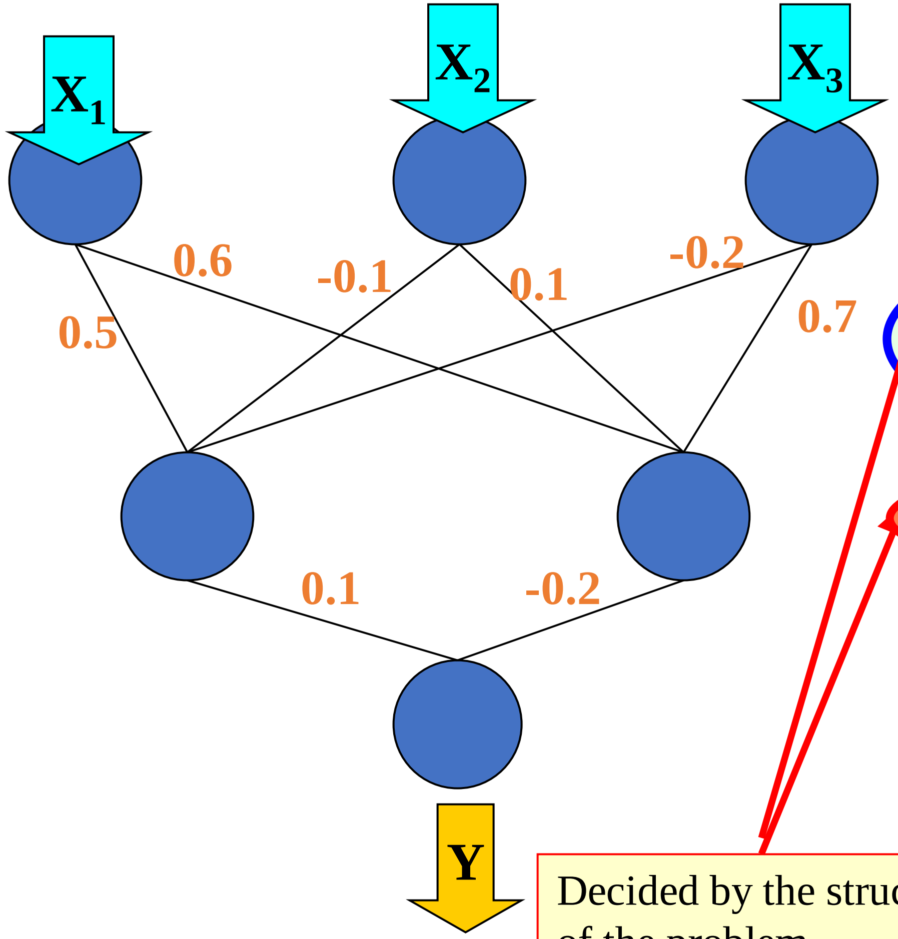




Eskers are long, winding ridges of stratified sand and gravel (LHS and middle panels), found in glaciated and formerly glaciated regions of Europe and North America. The RHS panel is the sampled hillshaded DEM area in Canada

Methods: Techniques (Artificial Neural Networks)

Input: $X_1\ X_2\ X_3$ **Output:** Y **Model:** $Y = f(X_1\ X_2\ X_3)$



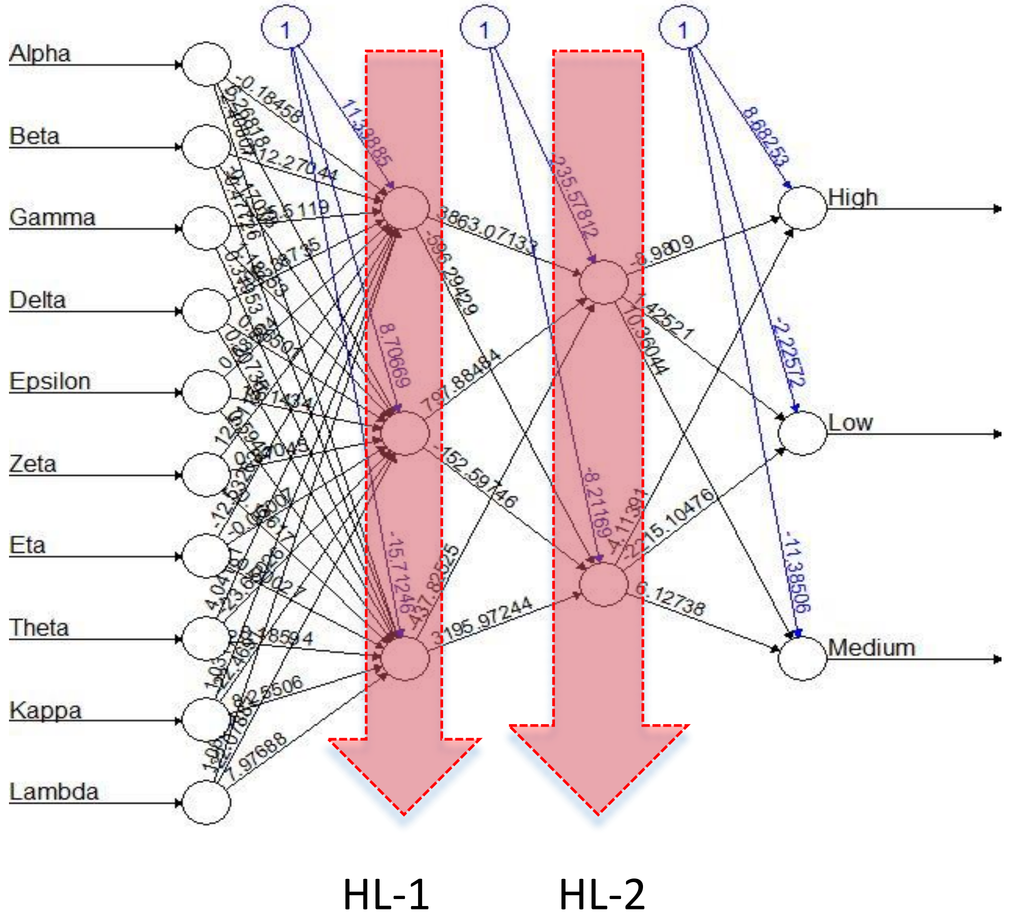
Parameters	Example
# Input Neurons	3
# Hidden Layers	1
# Hidden Layer Size	2
# Output Neurons	1
Weights	Specified

Decided by the structure of the problem
Input neurons/Number X's
Output Nrns = # of Y's

Free parameters

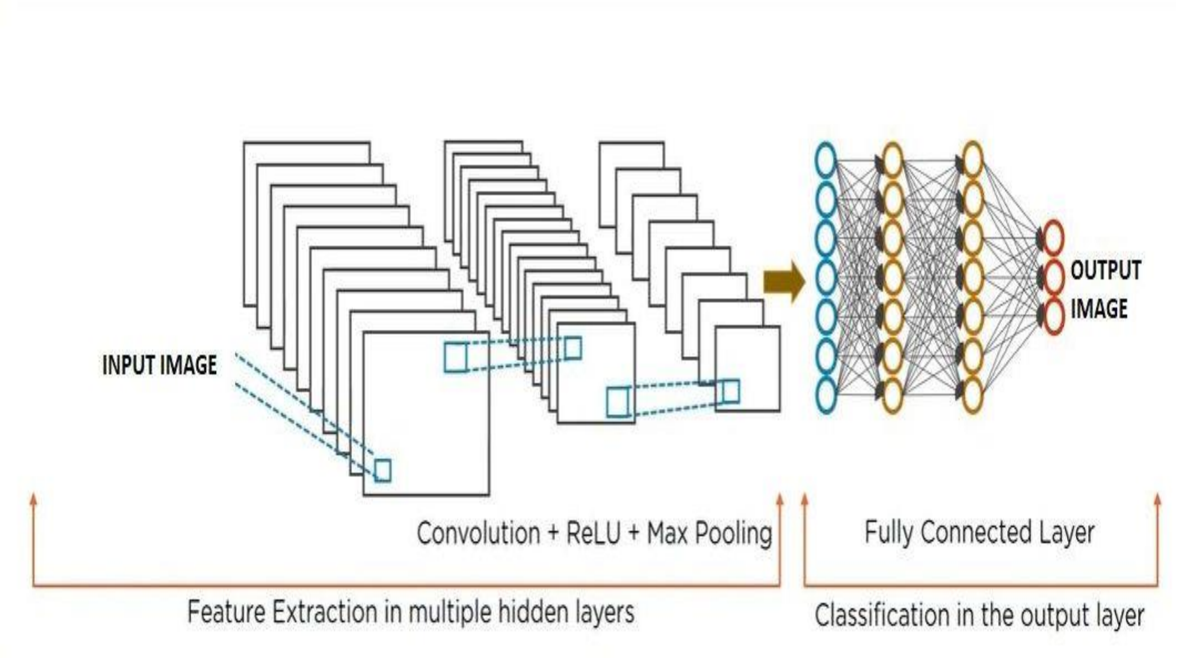
Convolutional Neural Networks for Imagery Data

- ❑Neurons at HLs are one dimensional while with CNN, they are 3 dimensional (Width, Height and Depth). The role of hidden layers in CNN is played by Convolutions. Unlike ANNs, CNNs don't require full connectivity.
- ❑Curse of dimensionality: A 20x20 image would require 1200 sets of weights-that is 20x20x3 (RGB). Convolution 1 will transform... Pass its output on to convolution 2 which will process and generate the output, in this case, an image.



Convolutional Neural Network Architecture

- **Input Image:** This is, in our case the two COVID-19 scenarios – positive and negative
- **Feature Extraction:** Hidden layers – they consist of convolution, ReLU & Max Pooling.
- **Fully Connected** layer then receives this as input, for classifying the image…



- There are 3 types of these layers, namely: **Convolutional (CL), Pooling (PL)** and **Fully Connected (FC). Pooling** layers lie between CLs and aim to reduce parameters & improve robustness, by retaining only the most important features.
- The main idea of CNN is to "convolve" an image with a filter aimed at extracting the most important features from the image. That is to so what a radiographer would do with a naked eye.
- Think of CNNs as a sequence of layers each transforming a volume of activations to another through a differentiable functions (partial derivatives) and ultimately connecting the output and input.

Methods: Activation Functions – the Engines of ANN/CNN

- The sigmoid function returns 1 for all positive large inputs and 0.00 for all values less of equal to zero while the hyperbolic tangent function values are [-1.0, 1.0].
- Large networks using these functions tend to lose useful gradient information as the error is back-propagated into the network. Each additional layer tends to decrease the back-propagated error and as this is determined by the partial derivatives of the weights, it causes a problem commonly referred to as vanishing gradient.

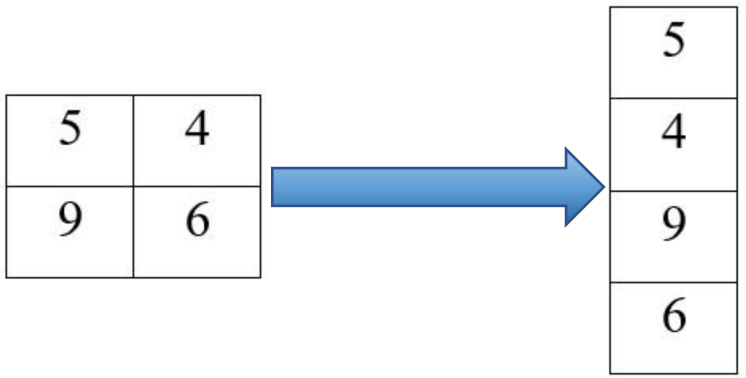
- The weight in a CNN play a similar role as coefficients in a linear regression model. It expresses the rate of change in the total loss would change, as the weight changes by one unit – i.e., the derivative of the Loss (L) with respect to W.

$$\frac{\partial L}{\partial w} = \lim_{\partial w \to 0} \left[\frac{L(w + \partial w) - L(w)}{\partial w} \right]$$

Rectified Linear Unit ReLU

- The **pooling** layer reduces the dimensionality of the rectified feature map. It uses different filters to identify different parts of the image – like edges, corners, curves etc.

- **Flattening** converts the 2-D arrays from the pooled layer into a one-dimensional vector.

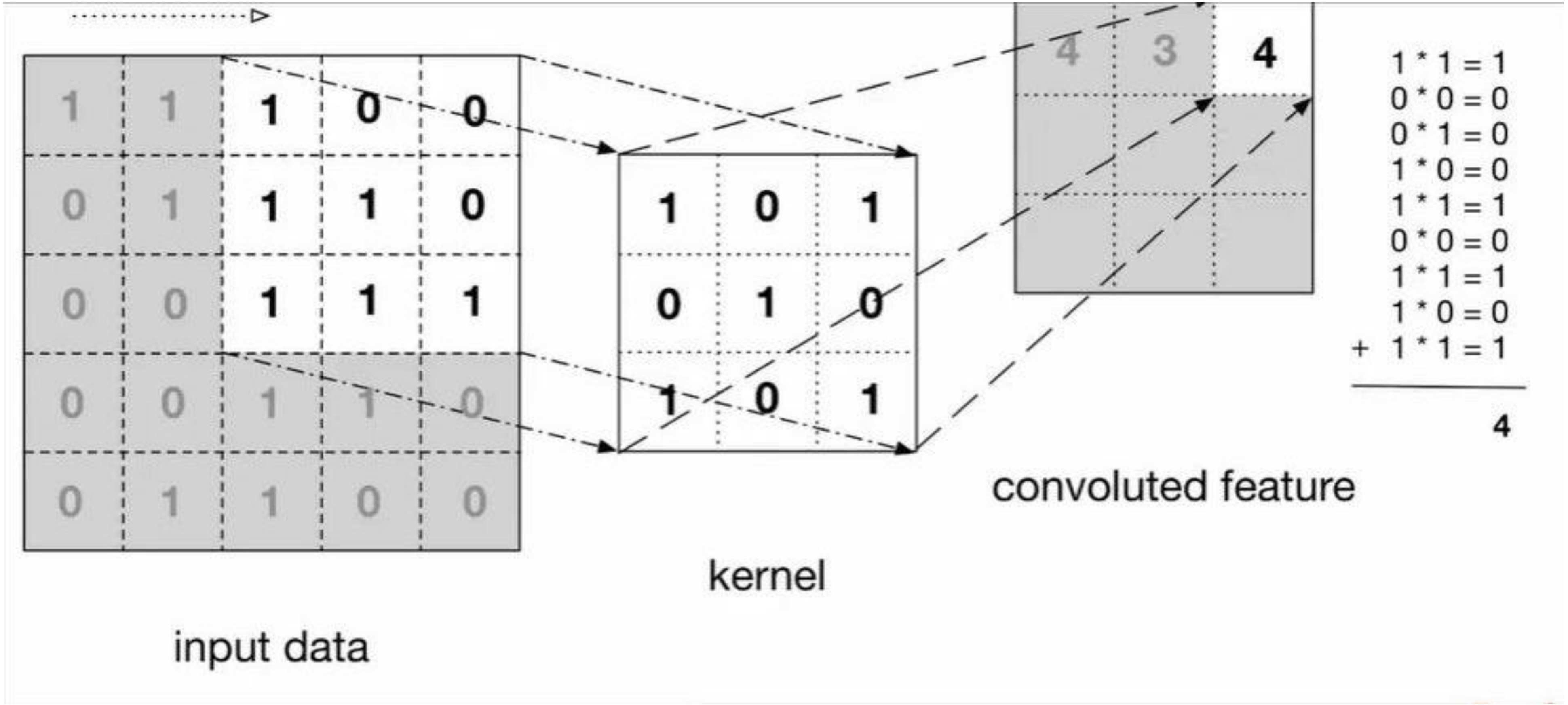


- The **Fully Connected** layer then receives this as input, for classifying the image. The Rectified Linear Unit (ReLU) applies the activation function, the most common of which being the following two:

$f(x) = \begin{cases} 0, & \text{if } \; x < 0 \\ x, & \text{otherwise} \end{cases}$	$f(x) = \dfrac{1}{(1 + e^{-x})}$
Maxi	Sigmoid

- Maxi is a piecewise linear function that outputs the input directly if is positive or zero otherwise. In other words, it is a binary rule that outputs the maximum positive number or zero otherwise.

Using the kernel to extract convoluted features from input data



Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

[RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html](https://www.rapidtables.com/web/color/RGB_Color.html)

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

[RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html](https://www.rapidtables.com/web/color/RGB_Color.html)

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

[RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html](https://www.rapidtables.com/web/color/RGB_Color.html)

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

[RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html](https://www.rapidtables.com/web/color/RGB_Color.html)

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

Note that the filter has reduced the input matrix to a smaller dimension of its own size.

RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html

Using the kernel to extract convoluted features from input data

To get the convolutional values, slide the kernel over the input data, multiplying the corresponding values and summing up and fill the matrix of the same dimension as the kernel.

0	0	1	1	0
0	1	1	1	0
1	1	1	0	1
1	0	0	1	0
1	1	1	0	0

Input Data

0	1	0
1	0	0
1	1	1

Kernel/Filter

3	4	4
2	2	2
5	3	1

Convolutional Features

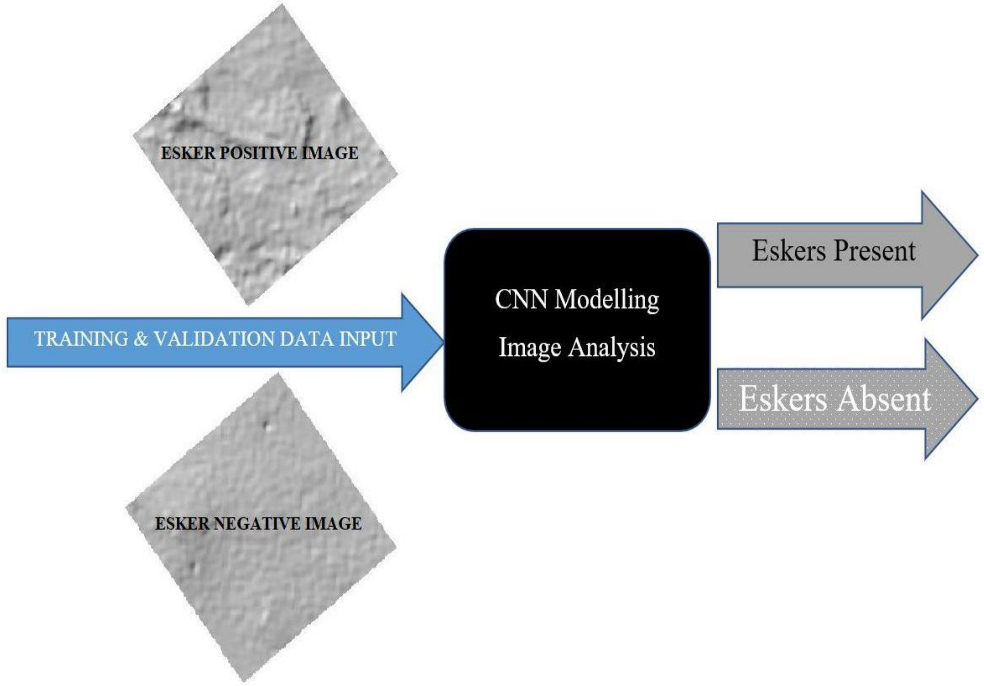
Note that the filter has reduced the input matrix to a smaller dimension of its own size.

RGB colour codes https://www.rapidtables.com/web/color/RGB_Color.html

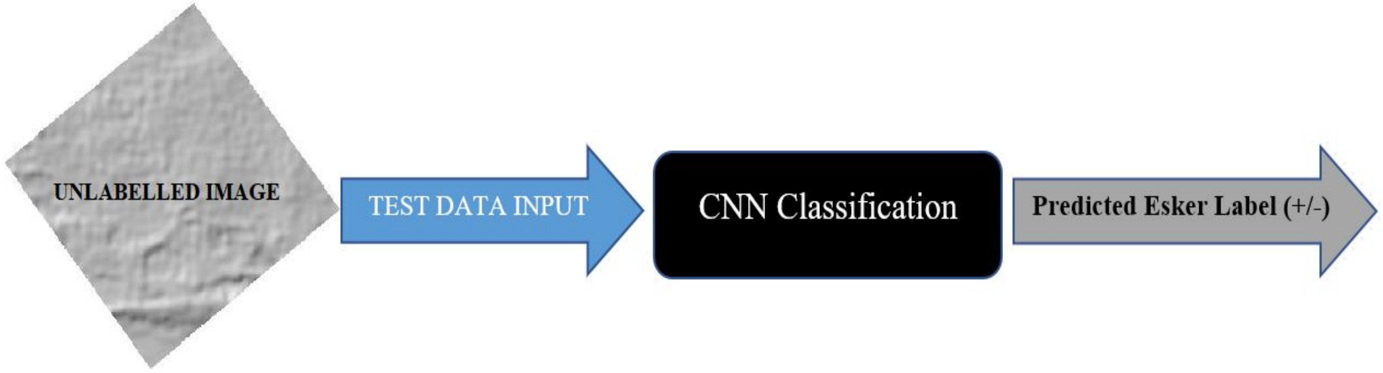
The Kernel

- A 2x2 pooling layer, filtering with a sliding of 2 down-samples at every depth of the input discards 75% of the activations.
- In this example, we applied a 2D convolution filter (***m*** *x* ***m***), which has a third dimension, equal to the number of channels of the input image. For the grey-scale images, it is ***m*** *x* ***m*** x1 (black and white channel), whereas for colour images, it is ***m*** *x* ***m*** *x 3* (RGB).
- Changing the dimension of the kernel has an impact on the way features are learnt on the image under study.

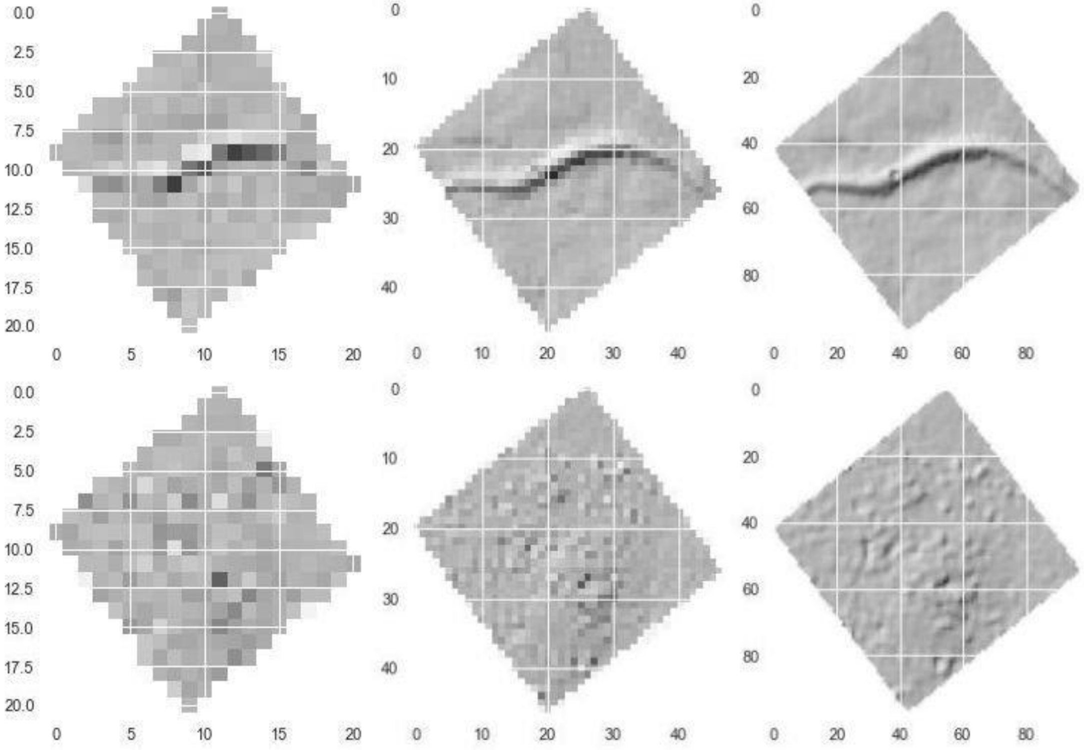
Training &Validating CNN Model

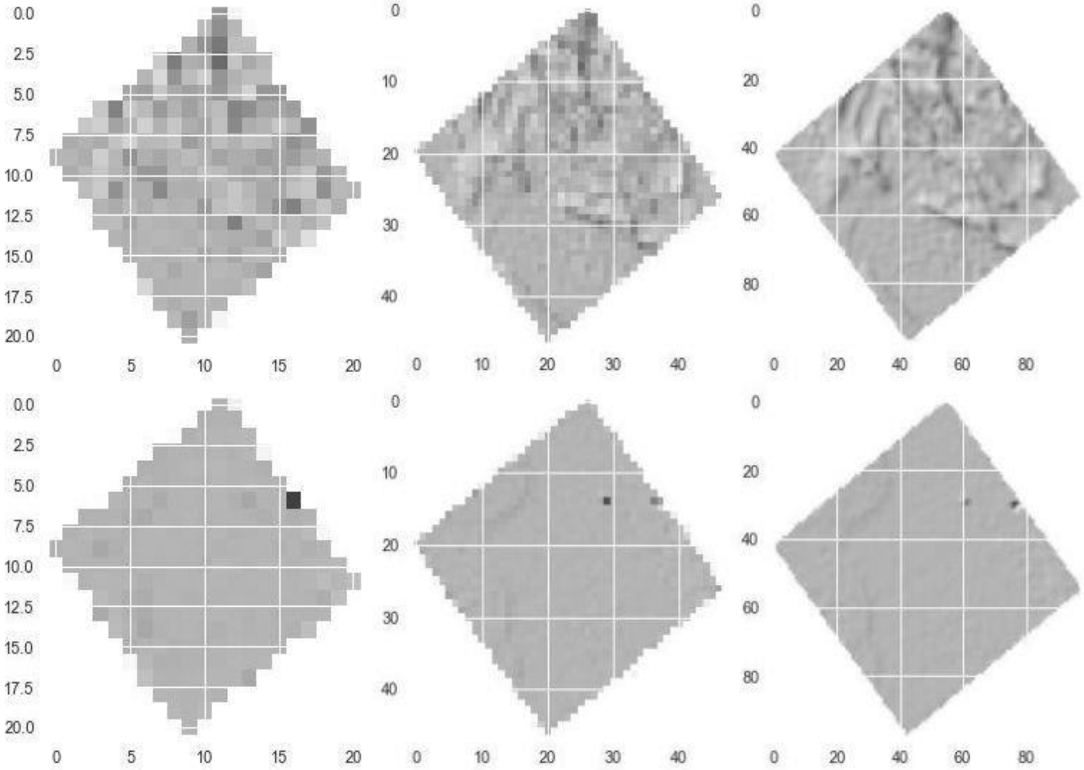


Testing CNN Model Performance



Two Examples of Esker Positive & Negative Images as Seen at Different CNN Model Convolutions

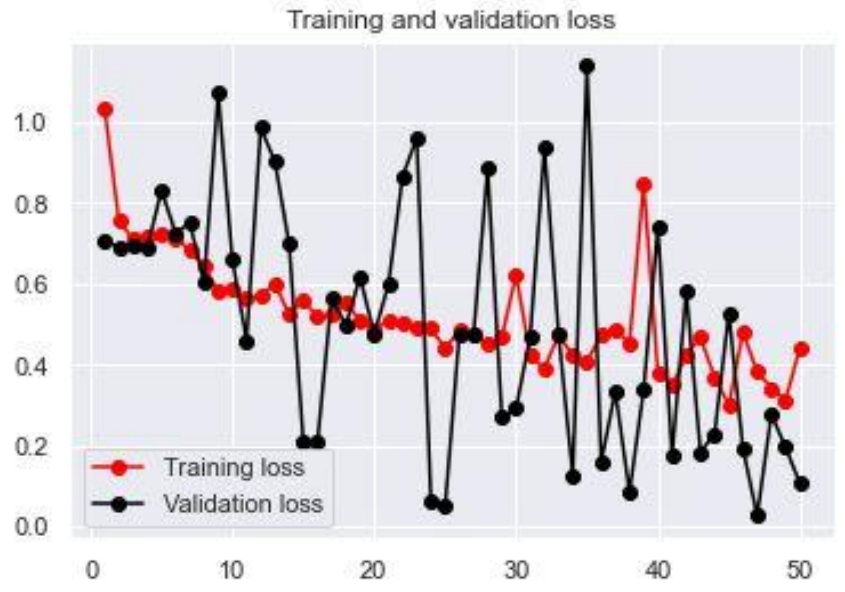




The left hand side panels exhibit example of esker positive and esker negative images as seen at different CNN convolutions.

The right hand side panels exhibit another example of esker positive and esker negative images as seen at different CNN convolutions. In both cases the presence or absence of protruding features on the images have a great impact on whether the image will be predicted as positive or negative.

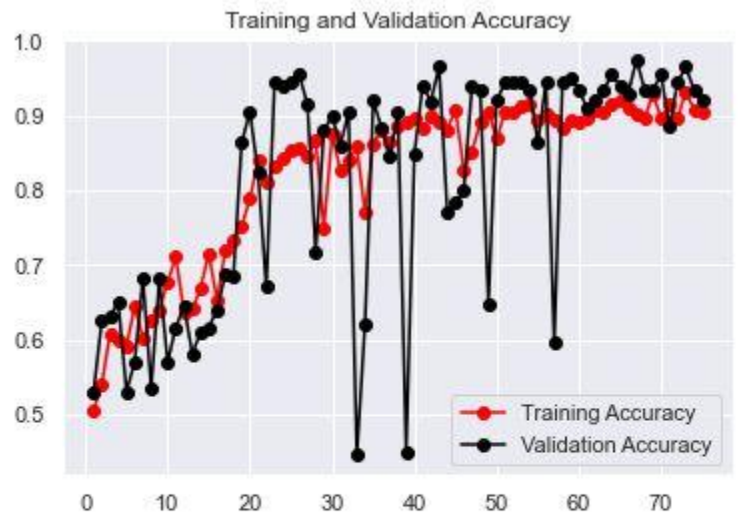
Training &Validating CNN Model on Esker Positive/Negative Data (100x100 pixels & 50 Epochs)

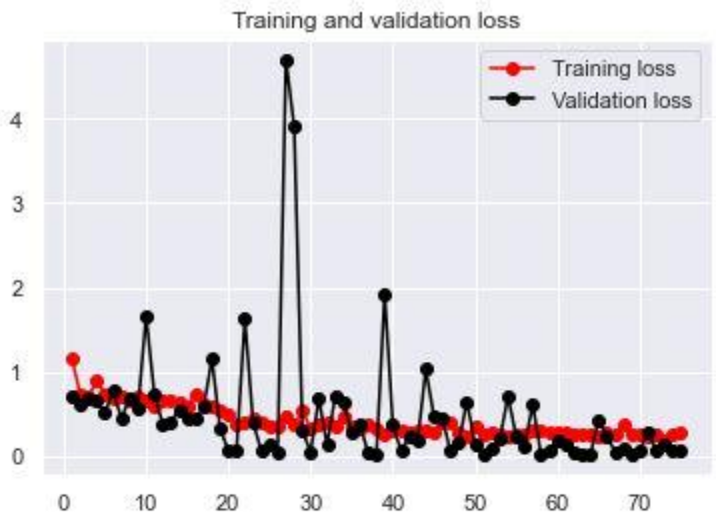


Attaining training and validation accuracy (LHS panel) is crucial for the performance of all learning algorithms. In this case here, both the training and validation accuracies increase with the number of epochs, although validation appears to oscillate. The latter remains below training (unsurprisingly)…

- Loss is the quantitative measure of deviation or difference between the predicted and actual values-it measures the mistakes the CNN makes in predicting the output. When loss exceeds validation loss, we have the case of underfitting, a rarity.
- The most common scenario is that of overfitting - i.e., when loss is significantly less than validation loss, which implies that the model is adapting so well to the training data that it considers random noise as meaningful data. In other words, the model fails to generalize well to previously unseen data.
- The ideal scenario is when loss is approximately equal to validation loss, as that would mean that the model is perfectly fitting on both training and validation data.

Training &Validating CNN Model on Esker Positive/Negative Data (250x250 pixels & 75 Epochs)





The ultimate goal of training, validating and testing machine learning, or indeed any, models, is to attain replicability to new data. Learning rules from palaeo-ice sheet images using CNN as done in this study is affected by a wide range of factors – from data randomness to specific parameter settings of the CNN model. Data randomness can have a particularly significant influence on modelling results. Hence, we deploy the SMA algorithm.

Dealing with Data Randomness: Sample-Measure-Assess (SMA) Algorithm

Algorithm 1 SMA-Sample, Measure, Assess

1: **procedure** SMA
2: Set $\mathbf{X} = [x_{i,j}]$: Accessible Data Source
3: Learn $F(\phi) = \underbrace{(P)}_{x,y\sim D} [\phi(x) \neq y]$ based on a chosen learning model
4: Set the number of iterations to a large number K
5: **Initialise:** $\Theta_{tr} := \Theta_{tr}(.)$: Training Parameters
6: **Initialise:** $\Theta_{ts} := \Theta_{ts}(.)$: Testing Parameters
7: **Initialise:** $\Pi_{cp} := \Pi_{cp}(.)$: Comparative Parameters
8: **Initialise:** s as a percentage of $[x_{\nu,\tau}]$, say 1%
9: s_{tr} : Training Sample $[x_{\nu,\tau}] \leftarrow [x_{i,j}]$ extracted from $\mathbf{X} = [x_{i,j}]$
10: s_{ts} : Test Sample $[x_{\bar{\nu},\tau}] \leftarrow [x_{l\neq i,j}]$ extracted from $\mathbf{X} = [x_{i,j}]$
11: **for** $i := 1 \rightarrow K$ **do**: Set K large and iterate in search of optimal values
12: **while** $s \leq 50\%$ of $[x_{\nu,\tau}]$ **do** Vary sample sizes to up to the nearest integer 50% of X
13: **Sampling for Training:** $s_{tr} \leftarrow X$
14: **Sampling for Testing:** $s_{ts} \leftarrow X$
15: **Fit Training and Testing Models** $\hat{\mathcal{L}}_{tr,ts} \propto \Phi(.)_{tr,ts}$ with current parameters
16: **Update Training Parameters:** $\Theta_{tr}(.) \leftarrow \Theta_{tr}$
17: **Update Testing Parameters:** $\Theta_{ts}(.) \leftarrow \Theta_{ts}$
18: **Compare:** $\Phi(.)_{tr}$ with $\Phi(.)_{ts}$: Plotting or otherwise
19: **Update Comparative Parameters:** $\Pi(.)_{cp} \leftarrow \Phi(.)_{tr,ts}$
20: **Assess:** $P(\Psi_{D,POP} \geq \Psi_{B,POP}) = 1 \iff \mathbb{E}[\Psi_{D,POP} - \Psi_{B,POP}] = \mathbb{E}[\Delta] \geq 0$
21: **end while**
22: **end for**
23: **Output the Best Models** $\hat{\mathcal{L}}_{tr,ts}$ based on $\mathbb{E}[\Delta] \geq 0$
24: **end procedure**

ALLOCATION RULE ERRORS DUE TO DATA RANDOMNESS			
Population	Training	Cross-Validation	Testing
Ψ_{POP}	Ψ_{TRAIN}	Ψ_{XVALID}	Ψ_{TEST}

The quantity $[x_{\nu,\tau}] \leftarrow [x_{i,j}]$ is the training dataset and its initialisation, in step 8, as a percentage of the full data depends on the nature and magnitude of the data source. The same applies to initialisation of $[x_{\nu,\tau}] \leftarrow [x_{l\neq i,j}]$. The loop from step 11through 19 involves sampling through the data with replacement, fitting the model and updating the parameters. The choice for the best performing model is carried out at step 20, where $P(\Psi_{D,POP} \geq \Psi_{B,POP})$ denotes the probability of the population error being greater than the training error.

Predicting New (Previously Unseen) Cases of Eskers/No Eskers

The CNN model is set to run with a "check point" monitoring both training and validation accuracy, saving the **best weights** and reporting each time performance improves, as follows.

```
checkpointer = ModelCheckpoint(filepath="best_weights.hdf5", monitor = 'accuracy', verbose=1, save_best_only=True)

tracks=model.fit_generator(training, steps_per_epoch=trainsamples//batchsize, epochs=epochs, callbacks=[checkpointer], validation_data=validation, validation_steps=validsamples//batchsize)
```

The saved best weights (the model) are then used to predict the class of any previously unseen image as follows.

```
#Esker positive images==[[1]]
#Esker negative images==[[0]]
# predicting previously unseen images
img1_path = "G:/ACADEMIC AND RESEARCH-RELATED/CHPC-DIRISA/eskerpostests/DEM_HS744.jpg"
img1 = image.load_img(img1_path, target_size=(100,100))
x = image.img_to_array(img1)
x = np.expand_dims(x, axis=0)
images = np.vstack([x])
classes = model.predict_classes(images, batch_size=10)
print("Predicted class is:",classes)
Predicted class is: [[1]]
```

```
#Esker positive images==[[1]]
#Esker negative images==[[0]]
# predicting previously unseen images
img2_path = "G:/ACADEMIC AND RESEARCH-RELATED/CHPC-DIRISA/eskernegtests/DEM_HS1182.jpg"
img2 = image.load_img(img2_path, target_size=(100,100))
x = image.img_to_array(img2)
x = np.expand_dims(x, axis=0)
images = np.vstack([x])
classes = model.predict_classes(images, batch_size=10)
print("Predicted class is:",classes)
Predicted class is: [[0]]
```

Conclusion

- Ultimately, this work highlights the potential of machine learning in enhancing our understanding of the subtle processes contributing to sea-level rise.
- By automatically identifying eskers, we can quickly and easily map large areas of ice sheet beds and will be able to constrain the dimensions of subglacial meltwater channels at unprecedented scales.
- This morphometric information on esker size can be used to inform numerical models of ice sheet hydrology, which in turn can feed into models predicting future ice loss and hence sea level rise.
- While CNN models can detect patterns that might go unnoticed to the human eye, for all their power and complexity, they do not provide thorough interpretations of the imagery data. For that we need interdisciplinarity.
- Going forward, we have the following suggestions for the scientific community.
- There can be no better way to view this bigger picture than through the Sustainable Development Goals (SDGs) initiative. We will need an interdisciplinary approach to respond to new challenges and exploit new opportunities in sectors like manufacturing, agriculture, business, health and education-sectors that have been badly hit by the pandemic. Tracking global variations in recovery strategies in various sectors and addressing real-life issues like food security, innovation, productivity etc., will be crucial.
- The esker images used in this paper are bare basics of deep and machine learning methods in glaciology and related fields, which we will need to explore further as a way of addressing climate change.

References

1. McGranahan, G.; Balk, D.; Anderson, B. The rising tide: assessing the risks of climate change and human settlements in low elevation coastal zones. Environment and Urbanization 2007, 19, 17–37.
2. Church, J. A.; Clark, P. U. Sea level change, in Climate Change 2013: The Physical Science Basis. Contribution of Working Group 1 to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. 2013; pp 1137–1216.
3. Hearty, P.; Hollin, J.; Neumann, A.; O'Leary, M.; Mcculloch, M. Global sea-level fluctuations during the Last Interglaciation (MIS 5e). Quaternary Science Reviews 2007, 26, 2090–2112.
4. Nienow, P.; Sole, A.; Slater, D. Recent Advances in Our Understanding of the Role of Meltwater in the Greenland Ice Sheet System. Glaciology and Climate Change: Current Climate Change Reports 2017, 3, 330 – 344.
5. Dirscherl M, K. C. K. C., Dietz AJ Automated Mapping of Antarctic Supraglacial Lakes Using a Machine Learning Approach. Remote Sensing 2020, 12, 1203.
6. Lai, K. J. W. M., CY. Vulnerability of Antarcticas ice shelves to meltwater-driven fracture. Nature 2020, 584, 574578.
7. Brennand, T. A. Deglacial meltwater drainage and glaciodynamics: inferences from Laurentide eskers, Canada. Geomorphology 2000, 32, 263 – 293.
8. Stroeven, A. P. et al. Deglaciation of Fennoscandia. Quaternary Science Reviews: Special Issue: PAST Gateways (Palaeo-Arctic Spatial and Temporal Gateways 2016, 147, 91 – 121.
9. Clark, P. U.; Walder, J. S. Subglacial drainage, eskers, and deforming beds beneath the Laurentide and Eurasian ice sheets. Geological Society of America Bulletin 1994, 106, 304–314.
10. Storrar, R. D.; Stokes, C. R.; Evans, D. J. Morphometry and pattern of a large sample (>20,000) of Canadian eskers and implications for subglacial drainage beneath ice sheets. Quaternary Science Reviews 2014, 105, 1 – 25.
11. Storrar, S. C., R.D.; Evans, D. Increased channelization of subglacial drainage during deglaciation of the Laurentide Ice Sheet. Geology 2014, 42, 239–242.
12. Livingstone, S.; Storrar, R.; Hillier, J.; Stokes, C.; Clark, C.; Tarasov, L. An ice-sheet scale comparison of eskers with modelled subglacial drainage routes. Geomorphology 2015, 246, 104–112.
13. Zhou, L.; Pan, S.; Wang, J.; Vasilakos, A. V. Machine learning on big data: Opportunities and challenges. Neurocomputing 2017, 237, 350 – 361.
14. Yan, M.; Haiping, W.; Lizhe, W.; Bormin, H.; Ranjan, R.; Zomaya, A.; Wei, J. Remote sensing big data computing: Challenges and opportunities. Future Generation Computer Systems 2015, 51, 47 – 60.
15. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 1980, 36, 193202.
16. LeCun, Y.; Jackel, L. D.; Boser, B.; Denker, J. S.; Graf, H. P.; Guyon, I.; Henderson, D.; Howard, R. E.; Hubbard, W. Handwritten Digit Recognition: Applications of Neural Net Chips and Automatic Learning. IEEE Communication 1989, 41–46, invited paper.
17. LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation 1989, 1, 541–551.
18. Krizhevsky, A.; Sutskever, I.; Hinton, G. E. In Advances in Neural Information Processing Systems 25; Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q., Eds.; Curran Associates, Inc., 2012; pp 1097–1105.Impact of the digital divide in the age of COVID-19. Journal of the American Medical Informatics Association 2020, 27, 1147-1148.

Thank You

Any Questions?