University of Victoria
ECE 470
Project Report
Summer 2021

Sea Ice Concentration Classification Using Sentinel-1 Optical Data:

Deep Learning Approaches on Pixel-based Image Segmentation

Sangwon Lim (V00923765)
Omar Kawach (V00925658)

July / 29 / 2021

Table of Contents

Abstract

Remote sensing can be a powerful tool across many applications. The utility of such technology is evident when attempting to study large areas. For example, on-ground measurements in polar regions would be too time-consuming and uneconomical for large-scale data collection. However, sea ice research using remote sensing still has its challenges. Synthetic Aperture Radar data only recently became more accessible to the public and the data itself may be difficult to deal with as it may require expert domain knowledge. On the other hand, optical data coupled with artificial intelligence techniques, may be easier to handle and may have comparable performance. To test this presumption, a model that performs supervised sea ice classification with a pixel-based approach was developed. After rigorous experimentation, the outcome was that out of a multilayer neural network, 1D convolutional neural network (CNN), and concatenation of both, the 1D-CNN with Grey-level Co-occurrence Matrix (GLCM) products, was the best classifier. Using GLCM products improved the overall classification since it had better outcomes when classifying open ice. In addition, of all the possible classification schemes, the 6-class scheme modified from Fisheries and Oceans Canada[5] was considered the best for real-world application. Any other n-class scheme was found to be somewhat impractical, particularly the 15-class scheme.

Introduction

Remote sensing is a technology in which by use of special devices, information regarding some phenomena in an area of study may be obtained. For example, the phenomena could be Earth's surface, the area of study could be some rural city, and the devices could be satellites. Once information about the phenomena in an area under study is obtained, the information may be converted to a rasterized image. By having information in such a format, one could begin conducting valuable research, such as sea ice concentration/type classification. The knowledge acquired from sea ice research would provide insight into issues like change detection (e.g., sea ice conditions over time), and icebreaker ship navigation [1][2]. The benefit to using remote sensing for sea ice research is that in comparison, on-ground (in-situ) measurements in polar regions would be too time-consuming and uneconomical for large-scale data collection. According to subject matter experts, one of the most popular ways to take sea ice measurements in remote sensing is by using a Synthetic Aperture Radar (SAR) system because of its advantages in surface texture information, day/night sensing and disruptive climate noise handling (clouds, extreme weather conditions, etc.) [3]. However, limitations in the use of SAR systems exist. SAR was initially developed for military applications, and publicly accessible SAR data did not come to fruition until recently (e.g., Sentinel-1 launched in 2014). Also, interpretation and manipulation of SAR data require expert knowledge of the electromagnetic characteristics of microwaves. For these reasons, the use of optical data in remote sensing coupled with artificial intelligence (AI) techniques to derive alternative sea ice detection models would be the next best option as the performance may be comparable. Therefore, instead of using microwave bands, the project uses visible and near-infrared (NIR) bands. The project aims

to perform supervised sea ice concentration clustering. Liu and Xia argue that a bigger segmentation unit may degrade the classification effectiveness in remote sensing problems [4]. Thus, a pixel-based approach is explored in this project, and the final developed model would empower and support subject matter experts in their sea ice research and real-world application in ice navigation [5].

Related Work

The project is based on the data provided by Sylvester, who worked on the same problem with an object-based 2-Dimension Convolutional Neural Network (2D-CNN) [6]. The dataset consists of 3362 Sentinel-2 satellite images with bands of 8 (NIR), 4 (Red) and 3 (Green), and their corresponding masks, which are the expert data. Each image has a patch number (Figure 1) indicating the location of the data collection in the Hudson Bay area. There are initially 15 classes existing in the dataset (Table 1), and the data distribution shows that there is a high concentration of pixels in the top three classes (Figure 2): 91 (9/10-10/10 or 9/10+), 100 (Land) and 1 (Less than 1/10). The data providers' previous work claims the output model of such architecture reaches about 83%. However, a further investigation suggests that this accuracy is measured on an imbalanced dataset, and it is assumed that the accuracy will be reduced to 55.8% if remeasured with an oversampled test dataset. Additionally, the work did not consider the spatial autocorrelation problem by randomizing the training and test data selections so that they can be sourced from shared patch locations. One goal of the project is to mitigate the overfitting by such characteristics of geographical data, where close together areas have similar values.

One main focus of the project is to perform feature engineering on the optical spectral data to enhance the performance of the sea ice concentration classification model. While most of the related works on this subject are using SAR data, not many experiments have been conducted using optical data with such an approach. Several related works on a sea ice type classification problem using SAR data applied a method of generating Gray-level Co-occurrence Matrix (GLCM) products to extract the surface texture information of an image [7], [8]. The method is relevant to the research topic since it approaches the problem in a pixel-based aspect. Thus, it can be used for feature engineering.

Due to the nature of the provided data and the pixel-based approach, a 1-Dimension Convolutional Neural Network (1D-CNN) architecture is suggested as an improvement of a regular Multi-layer Neural Network. Ketur et al. provide an exemplary architecture of 1D-CNN as a pixel-based classifier and feature extractors on a dataset with features of three optical spectral bands and their hue, saturation, and value values of pixels [9].

Problem Formulation

As revealed in the previous part, the result of using 2D-CNN for sea ice concentration classification is still unsatisfactory for 8 classes. One assumption about an object-based classification in remote sensing suggests that the approach can be risky when the classification

accuracy is low while the segmentation unit is extensive [4]. This project solely focuses on feature engineering and the exploration of deep learning architectures for pixel-based approaches.

To begin with, feature engineering is performed on the non-spectral metadata features that enhance the performance of the prediction and reduce overfitting by spatial autocorrelation and image indicators. Although deep learning architectures perform feature engineering automatically, this research project aims to maximize the classification accuracy with minimum geospatial data (i.e. the model should work on every location on earth outside of the Hudson Bay region). Therefore, it is required to identify the best feature selections that only assist the spectral data without such overfittings. Additionally, GLCM products are generated and added as features. The accuracy improvement by using the features is evaluated. For training, three Neural-Network architectures, multi-layer neural network, 1D-CNN and the concatenation of the aforementioned architectures, are tested with the best feature selections.

Methodology

Training and test datasets are created to evaluate the performance of a model's prediction. To handle the imbalance in the original data distribution, different sampling probabilities are used on different classes to maintain the balance in the outcome datasets (i.e. classes with a lower occurrence are assigned a higher probability). Nevertheless, the minority class can have a probability of up to 20% to mitigate spatial autocorrelation by spacing out between samples. Test samples are strictly sourced from the images that are not used for creating the training dataset to evaluate the overfitting by image indicator metadata features. Splitting the datasets is also achieved in the order sorted by patch number, which minimizes the number of samples in patches shared between the two datasets. For a better result, some features are reformatted to reduce the complexity of the training process. Month and day of the collection are combined to Day of Year (DOY), and patch numbers are reformatted to their centroids' (X, Y) coordinates to represent the latitude and longitude information. The initial features for the datasets include patch number, year, patch's Y coordinate, patch's X coordinate, Day of Year (DOY) of data collection, the hour of data collection, pixel's Y coordinate, pixel's X coordinate, band 8, band 4, band 3. The features other than the last three attributes are non-spectral metadata. All data are Min-Max normalized.

Since the project's goal is to maximize the pixel-based sea ice classification without overfitting by spatial autocorrelation, the use of metadata indicating the source image of a sample should be avoided in the training process. In the worst case, a model can classify a sample without spectral values by predicting the majority of the source image indicated by such features. However, optical data are collected using a passive sensor that uses sunlight as a source of energy. An assumption suggests that using just the spectral data can lose some critical information. For example, a pixel value can be lower when collected in the winter season, having less sunlight. To experiment with what non-spectral metadata features can be used, different

combinations are used for training along with the spectral data, and their effects on the overfitting are evaluated.

For every pixel, three GLCMs are generated corresponding to each band. To reduce the computation, 8-bit image is downscaled to 6-bit, resulting in a pixel value ranging from 0 to 63. The resulting GLCM is a 64×64 histogram matrix. For every pixel in a 5×5 sliding window, pixel value differences to eight directions from it within an interpixel distance of one are recorded to the GLCM histogram. The five GLCM products are calculated using the following equations:

$$Entropy = -\sum_{i=0}^{63}\sum_{j=0}^{63} GLCM[i][j] \times log(GLCM[i][j]) \quad (1)$$

$$Energy\ (ASM) = \sum_{i=1}^{63}\sum_{j=1}^{63} GLCM[i][j]^2 \quad (2)$$

$$Contrast = \sum_{i=1}^{63}\sum_{j=1}^{63} |i-j|^2 \times GLCM[i][j] \quad (3)$$

$$Homogeneity = \sum_{i=1}^{63}\sum_{j=1}^{63} \frac{GLCM[i][j]}{1+(i-j)^2} \quad (4)$$

$$Dissimilarity = \sum_{i=1}^{63}\sum_{j=1}^{63} GLCM[i][j] \times |i-j| \quad (5)$$

The calculations of the latter four features, (2), (3), (4) and (5), are performed using an existing Python package scikit-image [10], but a function was written for entropy calculation (1) in this project (Appendix A) [11]. The GLCM packages are also Min-Max normalized. The total number of GLCM features of a sample in the dataset is 3×5=15.

Once all the features are ready, three architectures are devised to train the sea ice concentration classification. First of all, a simple multi-layer neural network with one hidden layer is used to test various feature selections and provides the baseline accuracies of the project (Figure 3; Appendix C). The hidden layer's activation function is Rectified Loss Unit (ReLU) and passed into the output layer with a softmax activation function for multi-class classification. The input layer size is the number of features accepted, and the output layer size is the number of classes to predict. The hidden layer size is determined using the rule-of-thumb methods for determining "an acceptable number of neurons to use in the hidden layers" suggested by Heaton (Appendix B) [11], [12]. The second architecture, 1D-CNN, consists of three convolutional layers with ReLU activation (Figure 4). It is intended to train a model with spectral features and their GLCM products. Kernel size of five is selected to maximize the information in convolution,

where the center value is convolved with two different products of two other bands (Figure 5; Appendix D). To involve every possible combination of features, the first five values are padded at the end of their sequence, which results in an input layer size of 23×1. Convolutional layers have 64 filters, and their outputs get fully connected with a flatten layer with a size of 1472×1. It is then fed into the hidden layer and finally classified with the output layer. The same strategies of the multi-layer neural network for deciding layer sizes are used. The last architecture concatenates the multi-layer and 1D CNN (Figure 6; Appendix D). It enables the involvement of non-spectral features with convolved spectral data. On the left-hand side of the architecture is a multi-layer neural network where the input is a set of non-spectral features and the three spectral bands. It is then fed into a hidden layer to be processed and appended to the fully connected 1D-CNN architecture.

The last experiment compares four types of models trained with the three architectures using GLCM products. The first two models are trained with neural network architecture, where one uses non-spectral metadata features while another one does not. The third set of models are trained with 1D-CNN without the non-spectral metadata, and the fourth set of models uses the concatenated architecture to facilitate such features. These sets of models will be produced for all four class schemes.

Since 15-class classification is expected to have low accuracy, some are merged to experiment with a fewer number of classes. Only the adjacent classes are merged to maintain the logical formation. There are four merging schemes (Table 2): 15-class scheme with no merging, 8-class scheme defined by Sylvester [6], 6-class scheme modified from Fisheries and Oceans Canada where 9/10 and +9/10 concentrations are merged to Close Ice and Consolidated/Compact Ice [5], and 4-class scheme with classes of Open Water, <5/10, ≥5/10 and Land. Merging classes can cause an imbalance in both training and test datasets, so the oversampling strategy is used.

All models are trained in the Keras framework on Apple's M1 chip with a batch size of 1000 to maximize the learning rate. The maximum number of epochs is limited to 1000, where early stopping is enabled when there are no improvements in the loss value for 50 epochs. K-fold cross-validation is performed on every training with K=10. Validation samples are selected with stratification and without shuffling. The averaged accuracy using the method provides a reliable estimate of the true accuracy if the amount of training data is sufficiently large and the unseen test data follows the same distribution as the examples [13]. The resulting models are evaluated quantitatively by comparing the averaged test accuracies (6) and F1 scores (7) for each class, while qualitative evaluation is assessed based on the diagonality of the confusion matrices. The classes are ordered in the way that adjacent classes are similar to each other so that confusions occurring near the diagonal are considered as a better result.

$$Accuracy = \frac{number\ of\ correctly\ classified\ samples}{total\ number\ of\ test\ samples} \quad (6)$$

$$F1\ Score = 2 \times \frac{True\ Positive}{2 \times True\ Positive + False\ Positive + False\ Negative} \quad (7)$$

Results and Discussion

Following the methodology, 113013 and 24504 samples with 26 features were collected for the training and test datasets. Five feature combinations, all features, DOY/hour, Latitude/Longitude, DOY/hour & Latitude/Longitude, and only spectral features, are tested with the multi-layer neural network architecture for 15-class classification (Figures 7 and 8; Table 3). The results show that using any kind of combination results in the increment of not only the accuracy but also the discrepancy between the test and validation accuracies. Since the validation sampling method allows shared patches and images between validation and training datasets unlike the test sampling method, it is assumed that such discrepancies are caused by overfitted classifiers that are highly dependent on spatial autocorrelation characteristics and image indicator features. Also, the variance within the test and validation accuracies tends to get larger when non-spectral features are used. An assumption of overfitting by spatial autocorrelation or image indicator features suggests that the accuracy is highly affected by the number of shared images or patches between the training and validation sets. Since different folds create different sets each time, the variance would be high if there are features that cause overfitting. There is still a discrepancy between the test and validation accuracies for the model trained using only the spectral features that cannot be negligible (~6.7%). It is suspected that different amounts of sunlight could have affected the pixel values in different patches and images. Due to this reason, using the sunlight control factors, DOY and hour, might be a potential option that can improve the performance of the classifier model by minimizing such overfitting. Of all non-spectral metadata feature combinations, it is identified that the least variance and discrepancy between test and validation accuracies are obtained using DOY and hour features. Furthermore, the confusion matrices of the models using such features are generally more diagonal than the ones using only the spectral data (Figures 9 and 10), while there are some vertically shaped confusion matrices resulting from using latitude and longitude features (Figure 11). The biggest concern of using the DOY and hour features is that the misclassification of class 8 (6/10 concentration) with class 0 (Ice Free) is significant.

GLCM features are also tested with multi-layer neural network architecture for every classification scheme (Table 4). It is observed that, by using GLCM products, the accuracy is improved about 8.5% for 15-class classification and 17.4% for 4-class classification. The improvement is even better when compared with the result using raw GLCM features (Table 5). Furthermore, the fluctuations in the learning curve are more prominent when GLCM products are not normalized, which makes the model less reliable (Figures 12 and 13). Therefore, it is highly recommended that using normalized GLCM products for sea ice research. Using the GLCM products, the F1 scores have improved in every class except for Bergy Water, which only degraded by 0.0061. The most significant improvements are on the classes of Land, 4/10 and 5/10. The latter two classes are considered as Open Ice that is the hardest to be classified

considering the F1 score results. Therefore, using GLCM features play a considerable role in sea ice concentration classification problems.

Since it is identified that DOY and hour feature combination is the best selection of all metadata features, the last experiment is conducted by comparing the models using only spectral and GLCM products versus the models trained with the additional features. The results show that using the DOY and hour features improved the performance of the 15-class classifier by 6.8%~7.5% regardless of the architecture used (Table 7). However, this improvement is not apparent in the 8-class, 6-class and 4-class schemes, and some of the models are degraded using the DOY and hour features. It is assumed that the confusions made by the models trained using only GLCM and spectral data have occurred near the diagonal of the confusion matrices, resulting in a comparable or better performance than those using the DOY and hour features when classes are merged. Furthermore, the misclassification issue of 6/10 concentration as Ice Free is still significant even when DOY and hour features are trained with GLCM products even with some improvement in concatenation architecture. This indicates that there is a high chance of overfitting towards the metadata features (Figures 14 and 15; Table 8). The 47.65% accuracy for 8-class classification using the 1D-CNN is lower than that of Sylvester's 55.3% accuracy. However, the two models can be considered comparable since this project's model is evaluated with a stricter test dataset to factor out the spatial autocorrelation problem. The validation accuracy of the model is 57.89%, which is performed on less strict datasets. Classifying 2/10, 3/10, 4/10, 5/10 and 6/10 are generally the hardest to be classified in all architectures (Table 8).

In the interest of sea ice navigation, predicting sea ice concentration over a particular route may provide simpler expeditions since icebreakers prefer less than +9/10 sea ice concentration [5]. In Figure 15, the predicted 6-class sea ice concentration classification of Figure 17 using 1D-CNN can be seen to be comparable to the mask in Figure 16. Therefore, such a rapidly predicted route would assist these expeditions. The process is assumed to be rapid since navigators could predict numerous images in a short period of time. Each GLCM operation took approximately 306 seconds while the overall predicted operation with GLCM took approximately 314 seconds.

Conclusion and Future Work

Out of all the machine learning trials conducted, the classification schemes utilizing 1D-CNN with GLCM products had the optimum performance with confusions occurring near-diagonal, however the GLCM slowed down the runtime. The model appeared to struggle with predicting ice concentrations between 30-60%, which may mean the feature engineering should be improved. Given the differences in accuracy, the 15-class classification scheme was found to be impractical whereas the 6-class classification scheme was found to have the best real-world application. Though in some cases the DOY and hour features were useful when coupled with optical spectral data, the usefulness was not entirely consistent due to the possibility of overfitting. The data provider's 55.8% accuracy was higher than the project's

discovered 48.7% accuracy since the data provider did not consider spatial autocorrelation which may provide insight into why the validation accuracy and test accuracy are different. There have been some limitations in strictly clustering sea ice concentration levels in a limited number of classes. Instead, a regression model could benefit from this project. For example, confusion occurring in an adjacent class to the target class degrades the performance of the classification model but would have worked well with a regression model. Though 1D-CNN with GLCM products was found to have the best performance, the research may benefit from attempting to study the performance of 2D-CNN with GLCM features as there could potentially be improvements in the model. The research may also benefit from gathering other sources of expert data to determine whether the current expert data is generalized and as a result, impacting performance.

References

[1] Kruk, R., Fuller, M. C., Komarov, A. S., Isleifson, D., & Jeffrey, I. (2020). Proof of Concept for Sea Ice Stage of Development Classification Using Deep Learning. *Remote Sensing*, *12*(15), 2486.

[2] Natural Resources Canada. (2015). *Ice type and concentration*. https://www.nrcan.gc.ca/maps-tools-and-publications/satellite-imagery-and-air-photos/remote-sensing-tutorials/ice-type-and-concentration/9399.

[3] Barber, D. G., Yackel, J. J., & Hanesiak, J. M. (2001). Sea ice, RADARSAT-1 and Arctic climate processes: A review and update. Canadian journal of remote sensing, 27(1), 51-61.

[4] D. Liu and F. Xia, "Assessing object-based classification: Advantages and limitations," *Remote Sensing Letters*, vol. 1, no. 4, pp. 187–194, 2010.

[5] Fisheries and Oceans Canada, "Ice Navigation in Canadian Waters," 2012.

[6] A. Sylvester, "Sea-Ice: Data science project producing sea ice concentration maps from satellite images.," *GitHub*. [Online]. Available: https://github.com/asylve/Sea-Ice. [Accessed: 28-Jul-2021].

[7] R. Ressel, A. Frost, and S. Lehner, "A neural network-based classification for sea ice types on x-band sar images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 7, pp. 3672–3680, 2015.

[8] H. Liu, H. Guo, and L. Zhang, "SVM-Based sea Ice classification Using textural features and concentration FROM RADARSAT-2 Dual-Pol Scansar Data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 4, pp. 1601–1613, 2015.

[9] R. Kestur, S. Farooq, R. Abdal, E. Mehraj, O. Narasipura, and M. Mudigere, "UFCN: A fully convolutional neural network for road extraction in Rgb Imagery acquired by remote sensing from an unmanned aerial vehicle," *Journal of Applied Remote Sensing*, vol. 12, no. 01, p. 1, 2018.

[10] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu and the scikit-image contributors, "scikit-image: Image processing in Python" 2014, doi: 10.7717/peerj.453

[11] S. Lim, O. Kawach, "sea_ice_rs: Sea Ice Remote Sensing" *GitHub*. [Online]. Available: https://github.com/sum1lim/sea_ice_remote_sensing.

[12] J. Heaton, “The number of hidden layers,” *Heaton Research*, 20-Jul-2020. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html. [Accessed: 29-Jul-2021].

[13] P. Refaeilzadeh, L. Tang, and H. Liu, “Cross-Validation,” *Encyclopedia of Database Systems*, pp. 532–538, 2009.

Figures

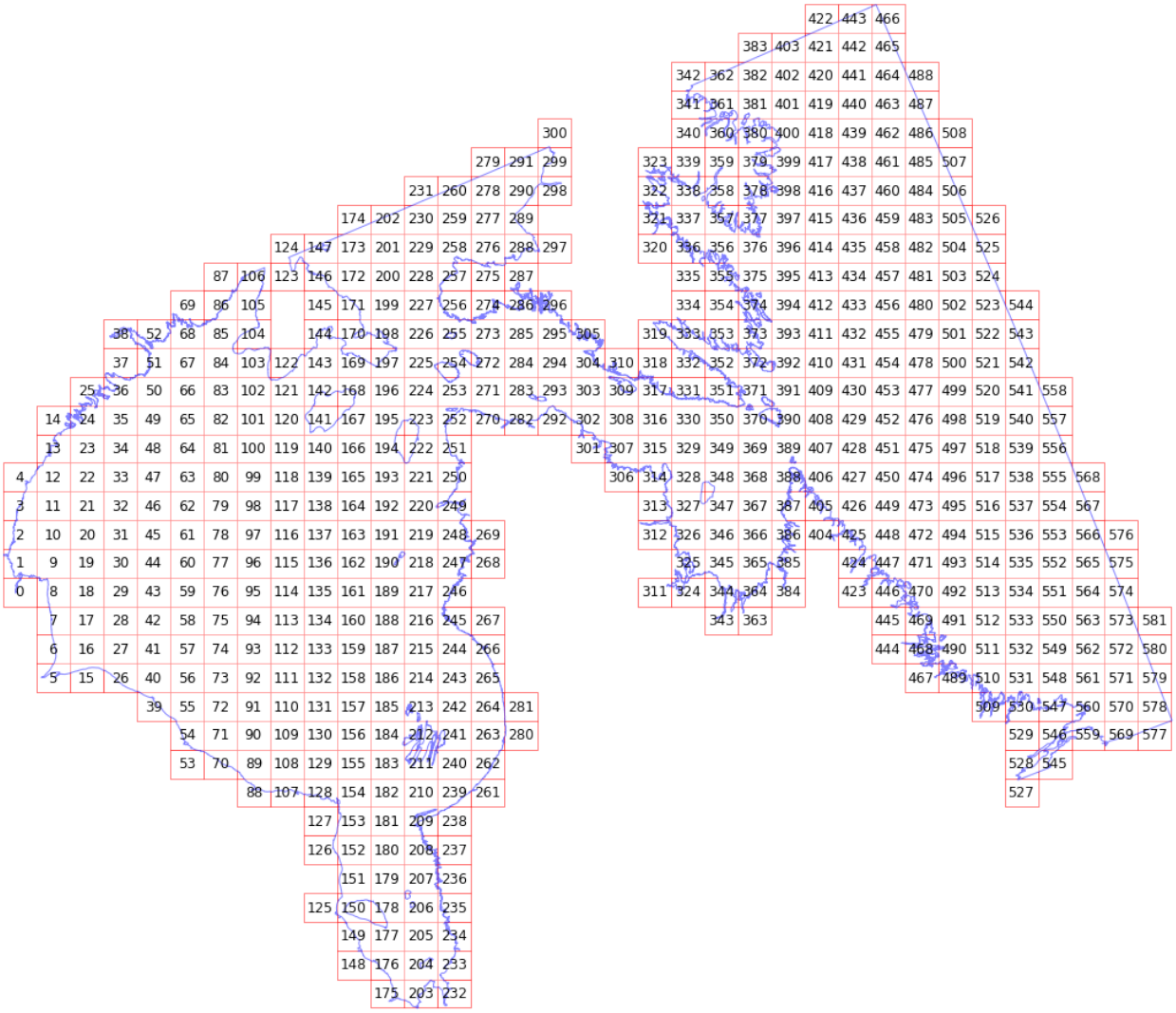

Figure 1. Region of Interest (ROI) of the project [6]. Patches are visualized as tiles in the Hudson Bay area.

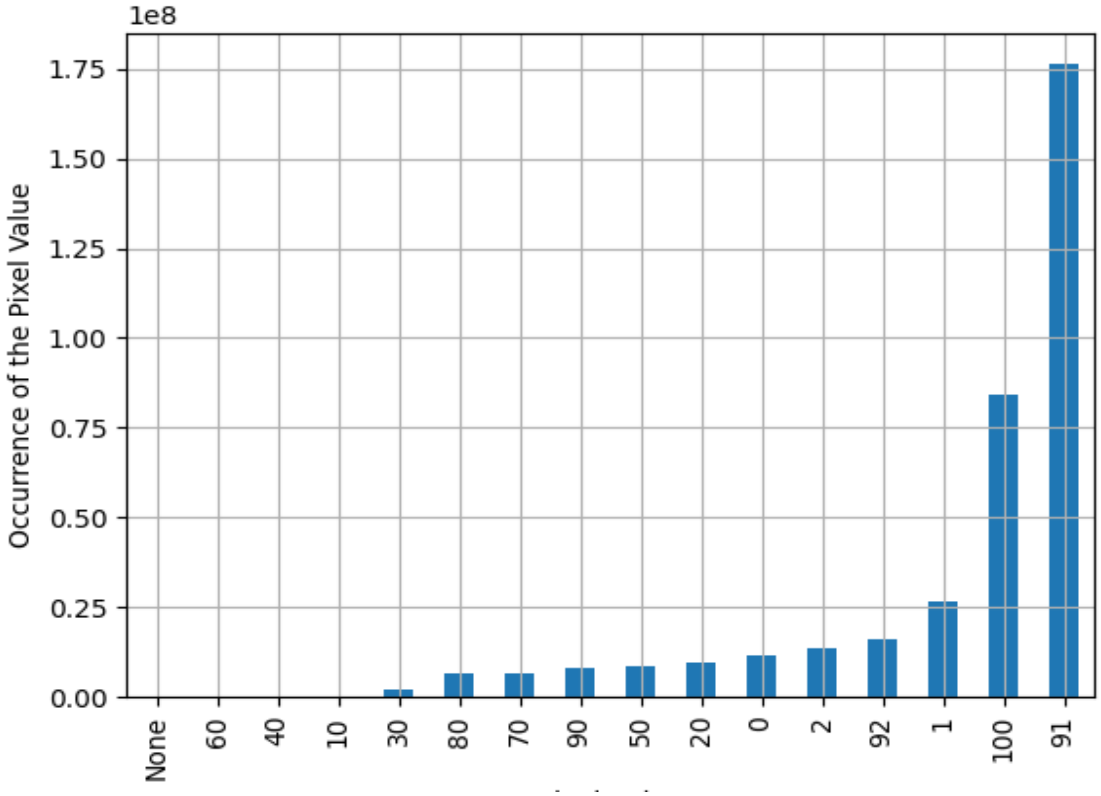

Figure 2. Data distribution of the expert data.

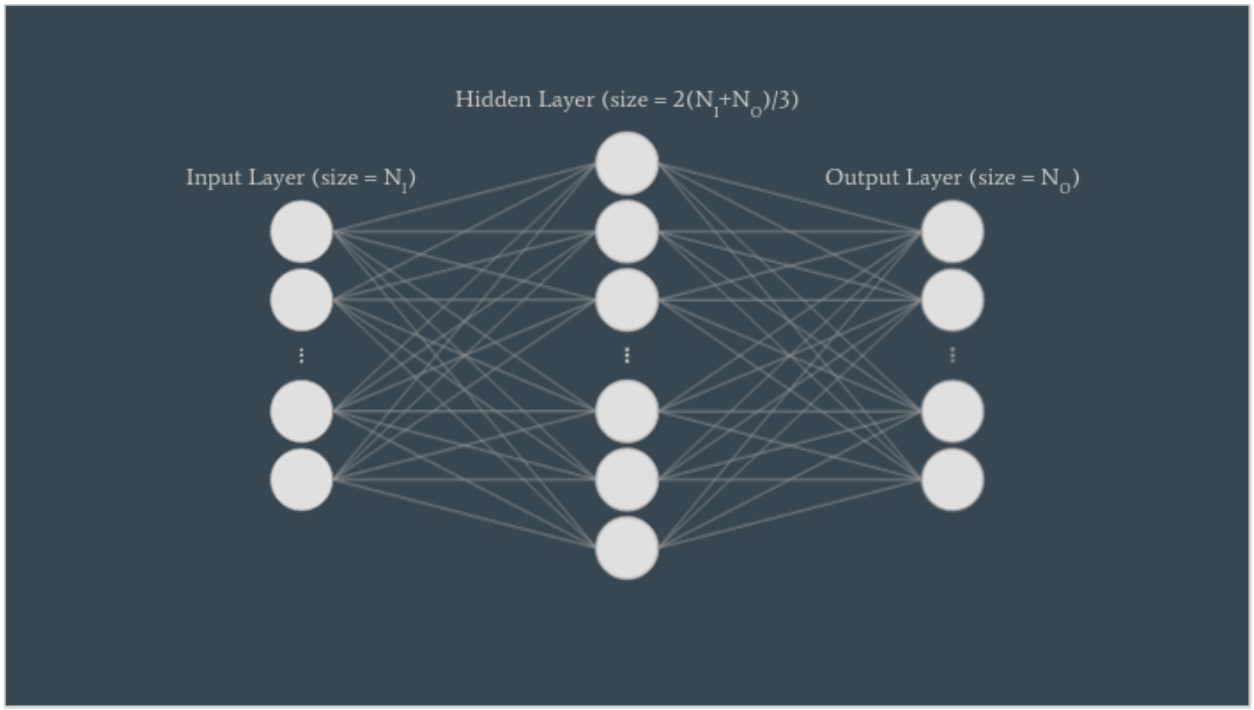

Figure 3. An illustration of a multi-layer neural network with one hidden layer.

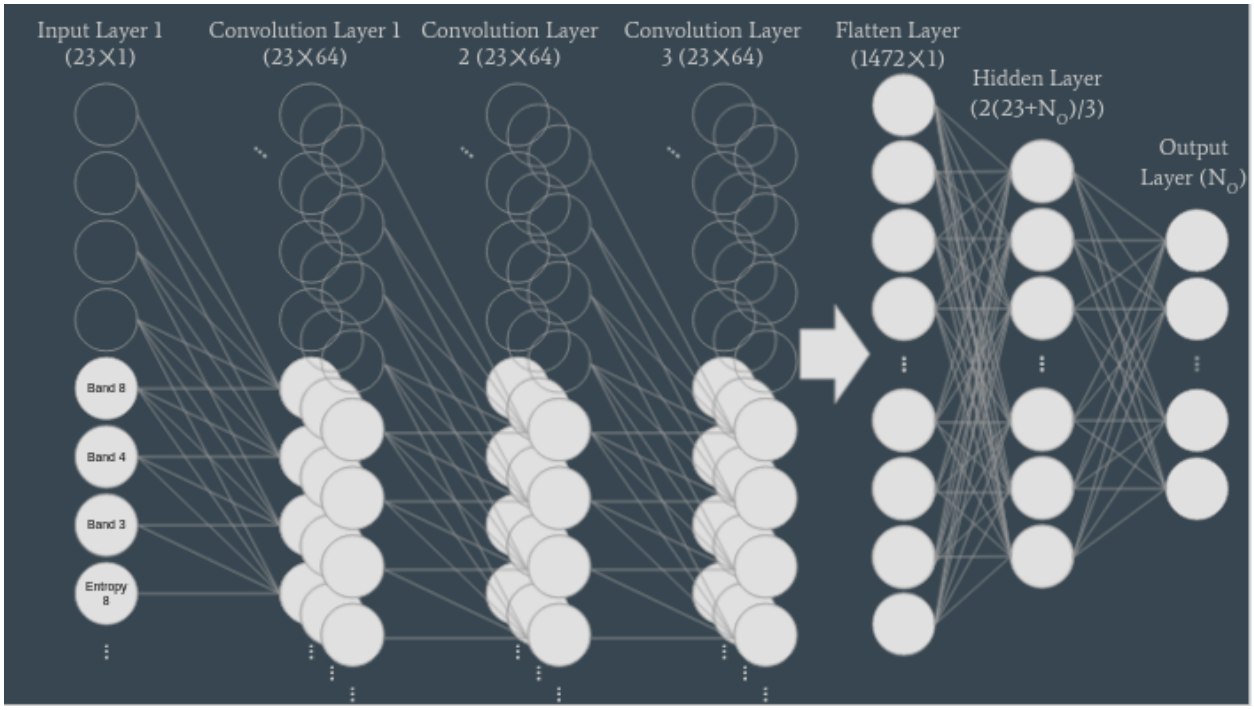

Figure 4. An illustration of a 1D-CNN with three convolutional layers.

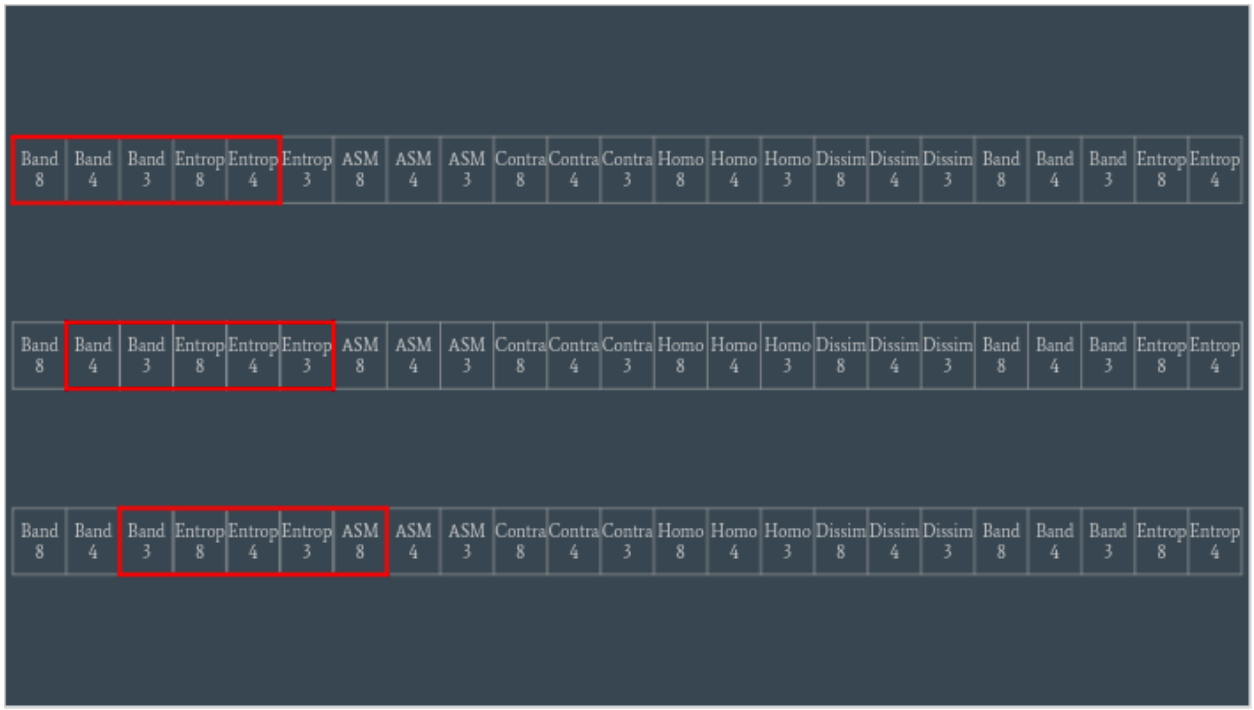

Figure 5. An illustration of the sliding kernel in the first convolutional layer.

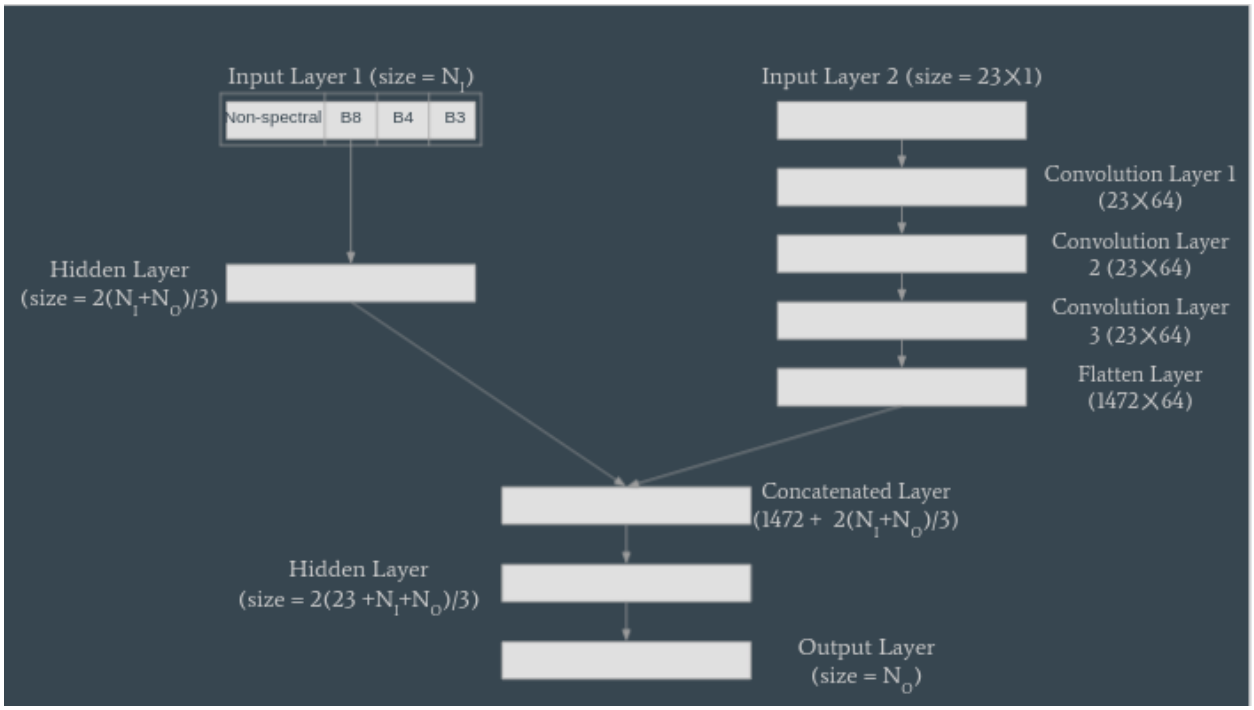

Figure 6. An illustration of the concatenation of a multi-layer neural network and a 1D CNN.

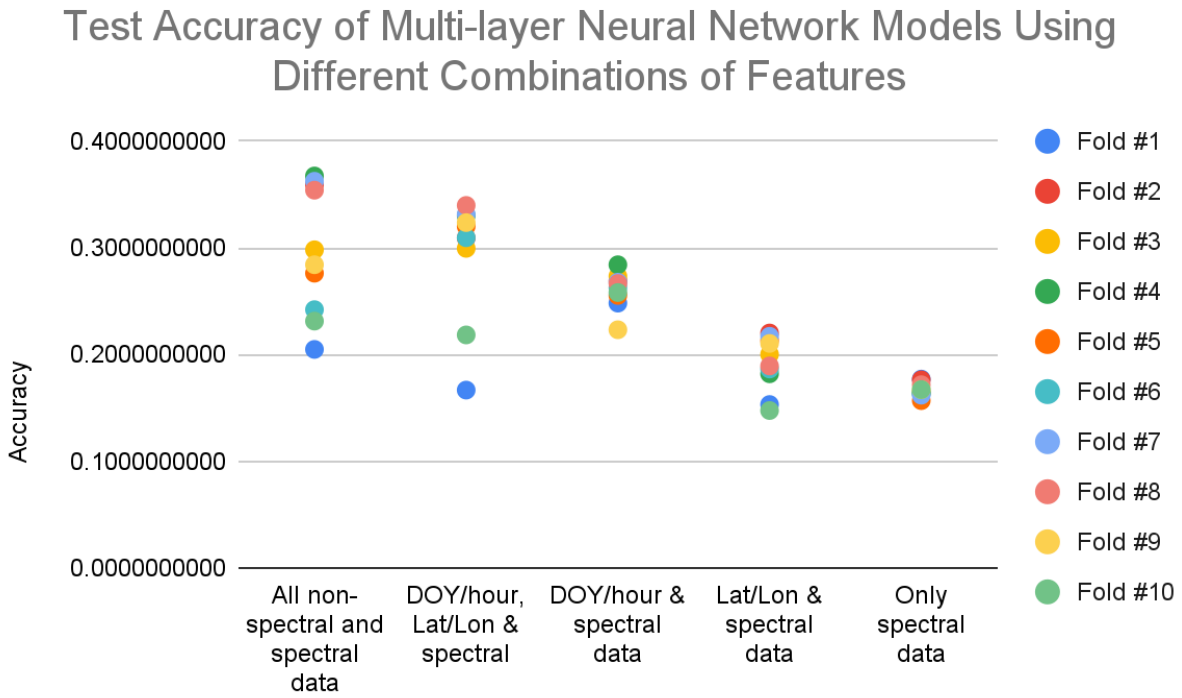

Figure 7. Plotted test accuracies of the models resulting from the multi-layer neural network architecture recorded for every fold.

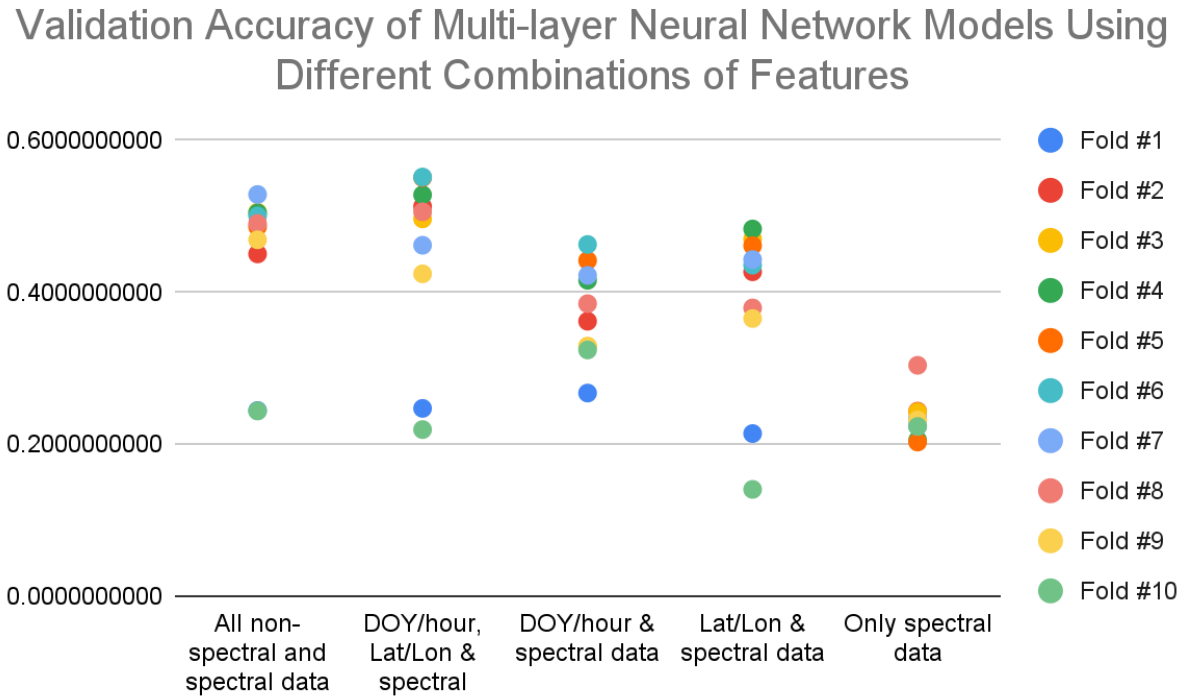

Figure 8. Plotted validation accuracies of the models resulting from the multi-layer neural network architecture recorded for every fold.

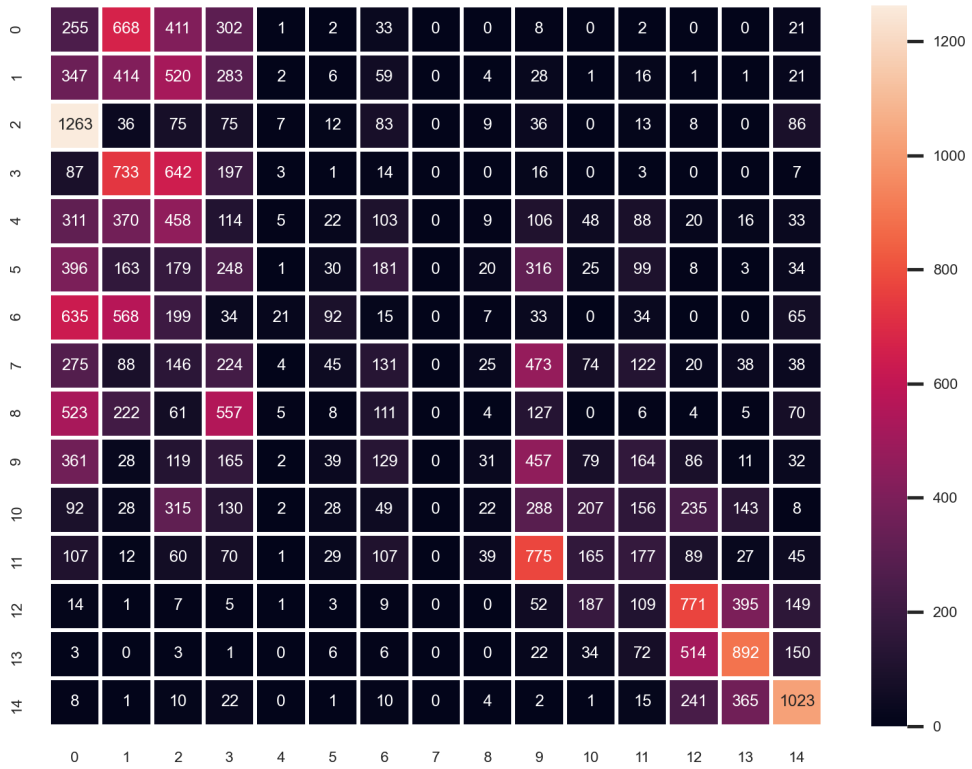

Figure 9. Resulting 15-class confusion matrix of the model created using the multi-layer neural network with only three bands of spectral data at first fold.

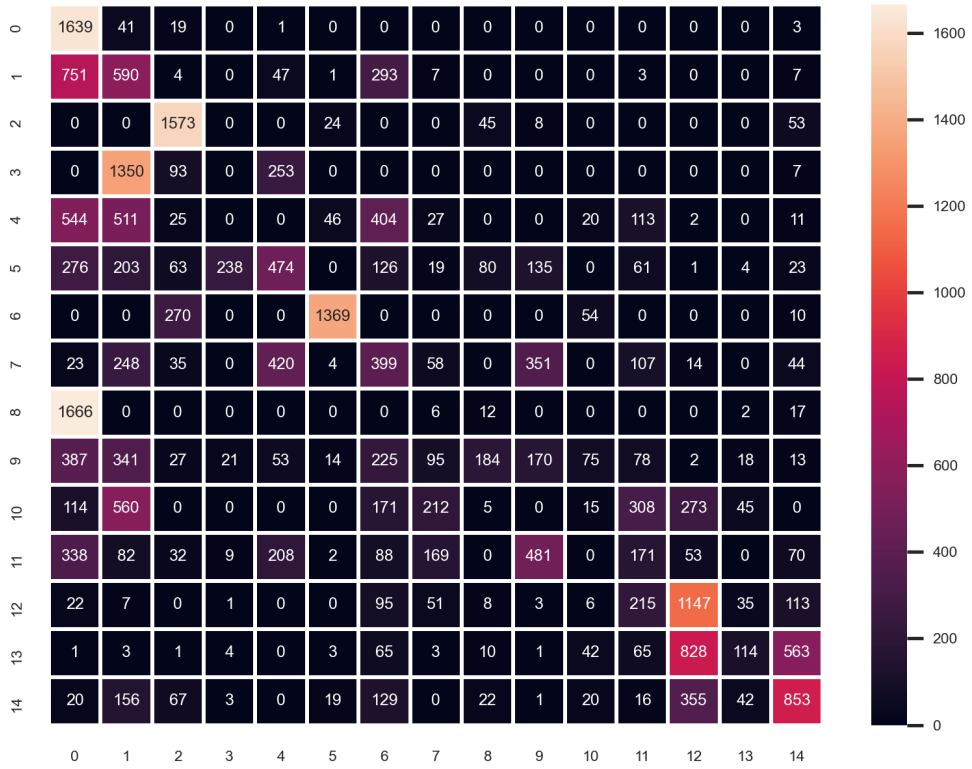

Figure 10. Resulting 15-class confusion matrix of the model created using the multi-layer neural network using DOY/hour with three bands of spectral data at first fold.

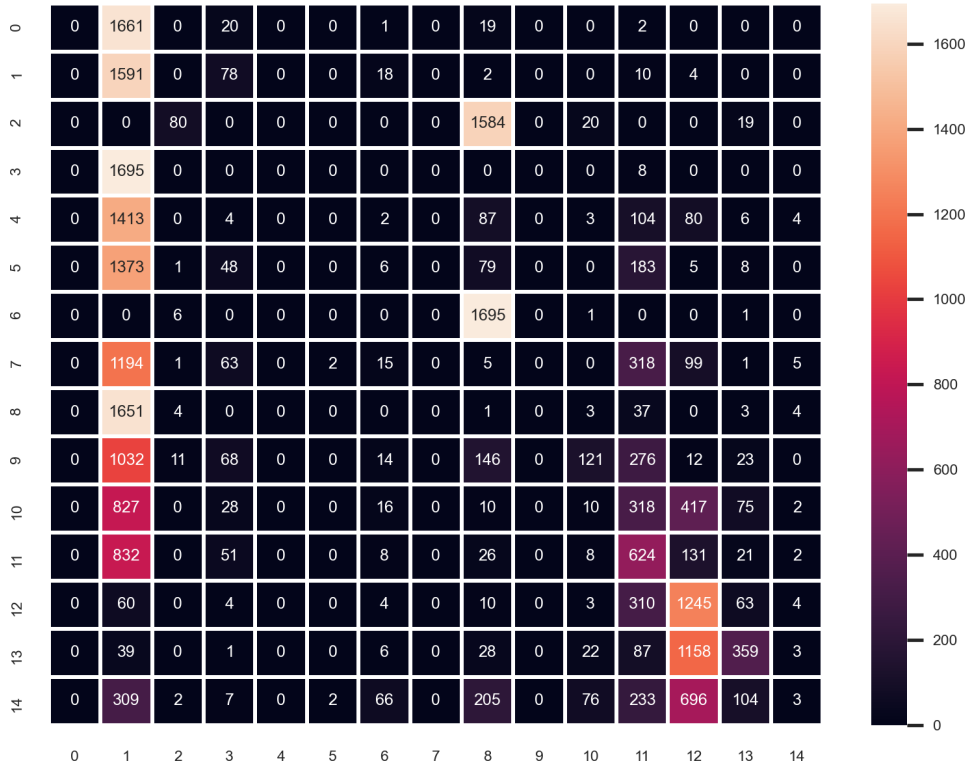

Figure 11. Resulting 15-class confusion matrix of the model created using the multi-layer neural network using Lat/lon with three bands of spectral data at first fold.

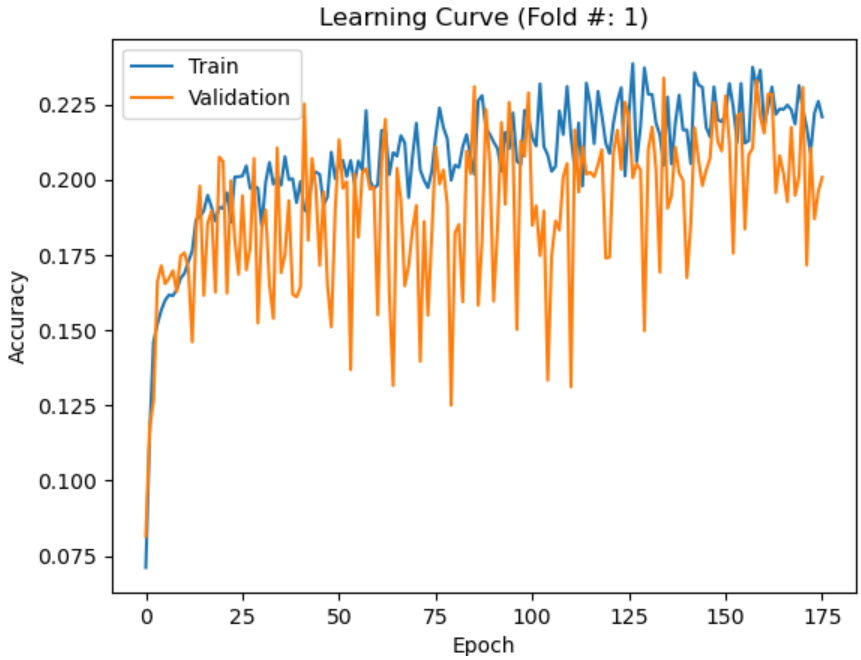

Figure 12. Learning curve of the 15-class classification using the multi-layer neural network with spectral bands and unnormalized GLCM products.

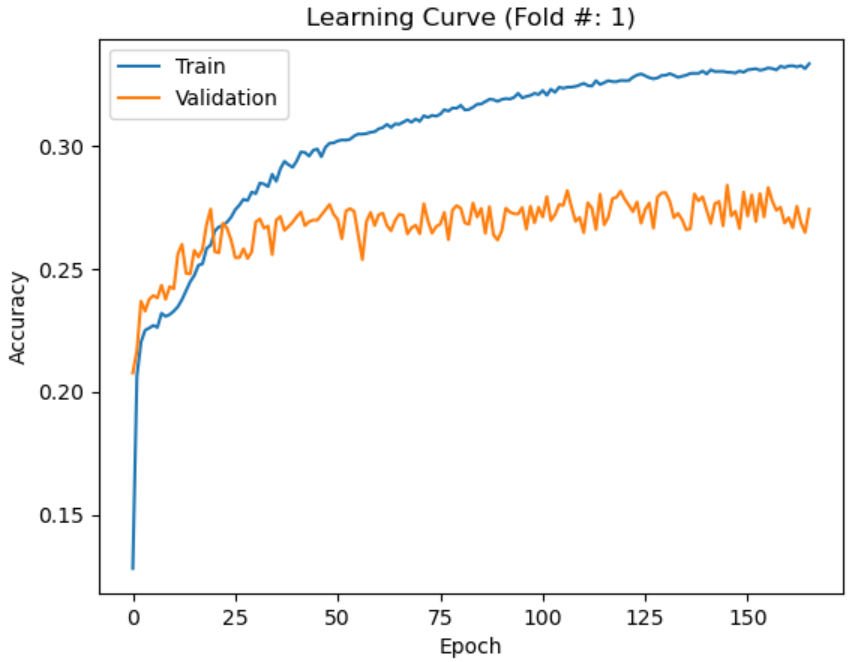

Figure 13. Learning curve of the 15-class classification using the multi-layer neural network with spectral bands and normalized GLCM products.

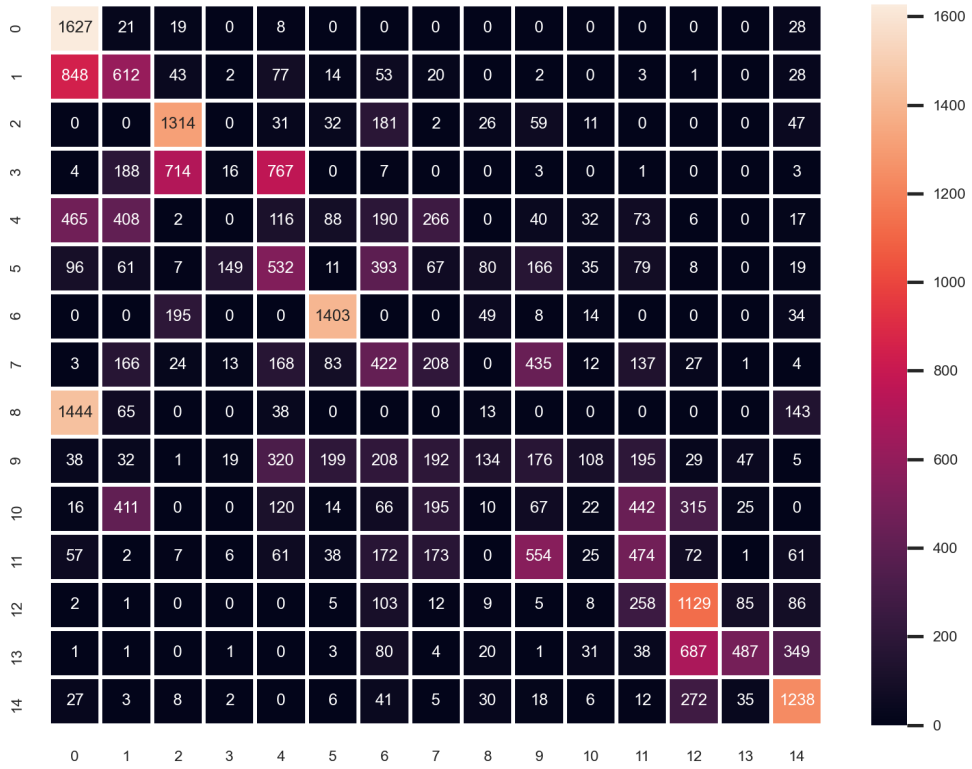

Figure 14. Confusion matrix of a classifier trained using a multi-layer neural network with features of DOY, hour, spectral bands and GLCM products.

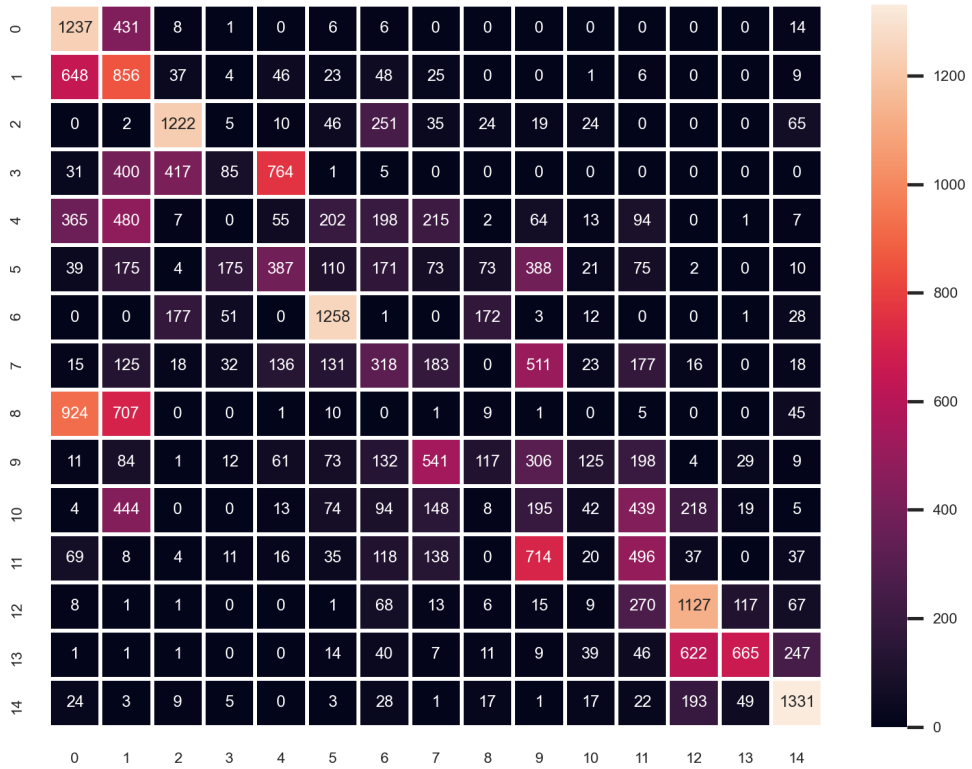

Figure 14. Confusion matrix of a classifier trained using a concatenated multi-layer neural network and 1D-CNN with features of DOY, hour, spectral bands and GLCM products.

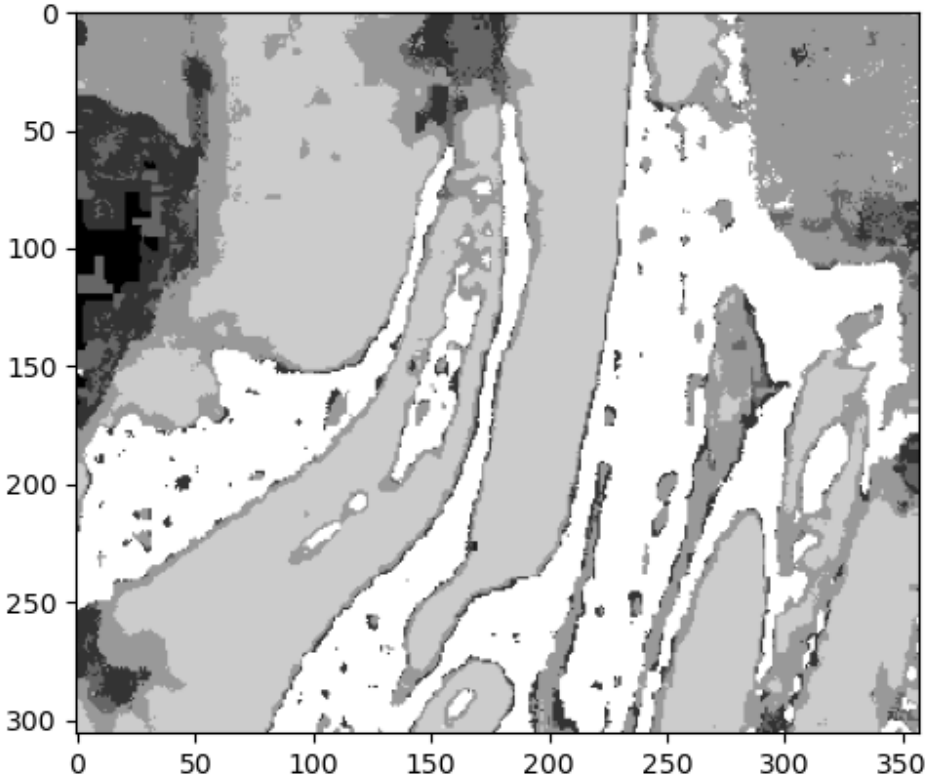

Figure 15. Predicted 6-class sea ice concentration classification using a 1D CNN model.

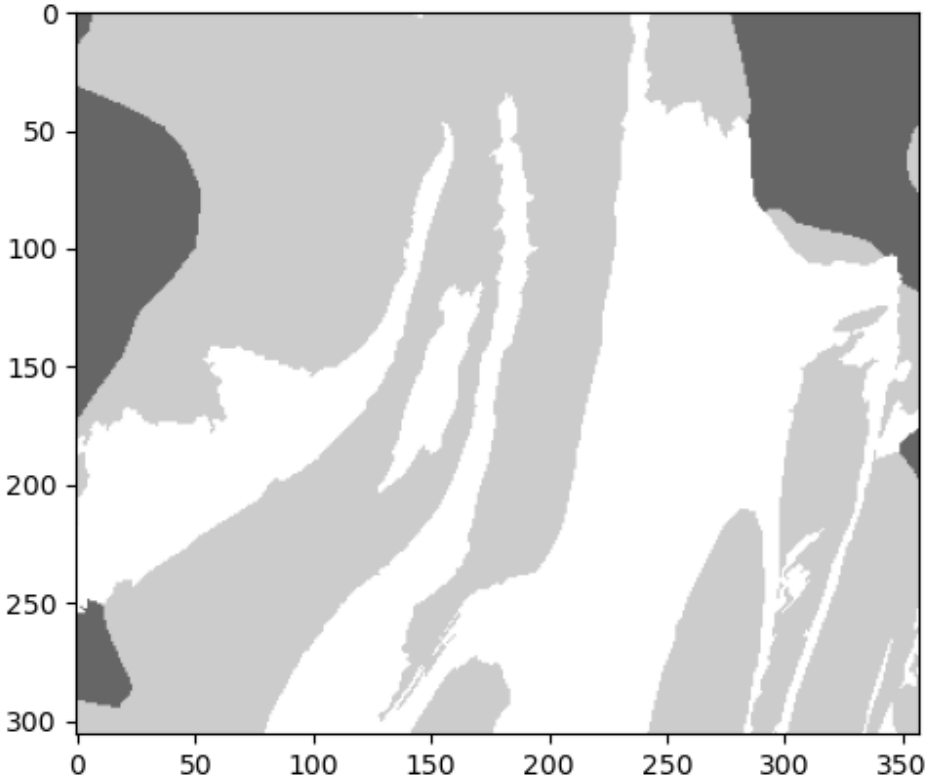

Figure 16. Expert 6-class sea ice concentration class map.

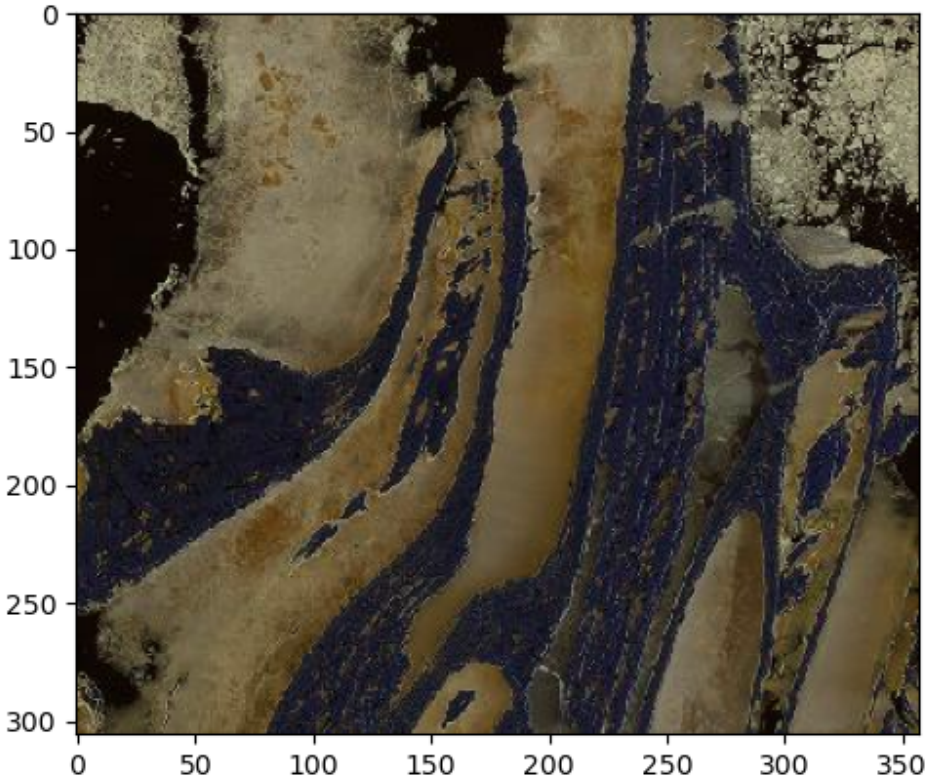

Figure 17. Optical Sentinel-2 imagery of the predicted area.

Tables

Table 1. Fifteen classes in the original expert data [5], [6].

Concentration	Mask code
Ice Free	00
Open Water	01
Bergy Water	02
1/10 (Very Open Ice)	10
2/10 (Very Open Ice)	20
3/10 (Very Open Ice)	30
4/10 (Open Ice)	40
5/10 (Open Ice)	50
6/10 (Open Ice)	60
7/10 (Close Ice)	70
8/10 (Close Ice)	80
9/10 (Very Close Ice)	90
9/10~10/10 or +9/10 (Very Close Ice)	91
10/10 (Compact/Consolidated Ice)	92
Land	100

Table 2. Sea ice class merging schemes

Label	Open Water	<1/10	1/10	2/10	2/10	3/10	4/10	5/10	6/10	7/10	8/10	9/10	> 9/10	10/10	Land
15 Classes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
8 Classes	0			1		2		3		4		5		6	7
6 Classes	0			1			2			3			4		5
4 Classes	0			1				2							3

Table 3. Summary of feature selection results

		All non-spectral and three bands	DOY/hour, lat/lon and three bands	DOY/hour and three bands	Lat/lon and three bands	Three bands only
Test Accuracy	Average	29.8062%	29.4911%	26.0932%	19.1995%	16.7062%
	Variance	0.0036439424	0.0031997170	0.0002739805	0.0006562648	0.0000406112
Validation Accuracy	Average	44.2260%	44.9790%	37.3730%	38.1920%	23.3550%
	Variance	0.0113669360	0.0145604477	0.0038682068	0.0133014373	0.0007924606
Accuracy Discrepancy		14.41978%	15.48791%	11.27983%	18.99255%	6.64880%

Table 4. Averaged test accuracy improvement by using GLCM products.

	15-class	8-class	6-class	4-class
Without GLCM	16.7062%	33.9799%	42.4865%	51.3087%
With GLCM	25.2104%	44.7590%	57.4437%	68.7601%
Improvement	8.5042%	10.7790%	14.9572%	17.4514%

Table 5. Test accuracy improvement for 15-class classification by normalizing GLCM products.

	Raw	Normalized
Spectral data	16.2940%	16.7062%
Spectral data & GLCM	22.9896%	25.2104%

Table 6. F1 score improvement using GLCM products for 15-class classification with the multi-layer neural network.

Class	Only Spectral	Spectral and GLCM	Improvement
Ice Free	0.0764	0.1900	0.1136
Open Water	0.0836	0.1700	0.0864
Bergy Water	0.0291	0.0230	-0.0061
1/10 (Very Open Ice)	0.0327	0.2420	0.2093
2/10 (Very Open Ice)	0.0082	0.1220	0.1138
3/10 (Very Open Ice)	0.0145	0.0860	0.0715
4/10 (Open Ice)	0.0182	0.0540	0.0358
5/10 (Open Ice)	0.0009	0.1590	0.1581
6/10 (Open Ice)	0.0155	0.2000	0.1845
7/10 (Close Ice)	0.1791	0.2370	0.0579
8/10 (Close Ice)	0.1945	0.2140	0.0195
9/10 (Very Close Ice)	0.1355	0.1920	0.0565
9/10~10/10 or +9/10 (Very Close Ice)	0.3482	0.4720	0.1238
10/10 (Compact/Consolidated Ice)	0.4982	0.6160	0.1178
Land	0.5200	0.7200	0.2000

Table 7. Averaged test accuracies for the classifiers trained using GLCM products.

# classes	15	8	6	4
Multi-layer NN (Without DOY & hour)	25.2104%	44.75895%	57.4437%	68.7601%
Multi-layer NN (With DOY & hour)	32.0736%	43.81627%	59.2780%	68.0894%
1D CNN (Kernel size = 5)	24.6624%	47.64721%	59.8267%	70.5028%
Concatenation (With DOY & hour)	32.1511%	46.68985%	60.5355%	69.0355%

Table 8. F1 scores for 15-class classification models trained using GLCM products.

Class	Multi-layer NN (Without DOY & hour)	Multi-layer NN (With DOY & hour)	1D CNN (Kernel size = 5)	Concatenation (With DOY & hour)
Ice Free	0.1900	0.4930	0.132	0.4370
Open Water	0.1700	0.3320	0.205	0.3120
Bergy Water	0.0230	0.6020	0.028	0.6350
1/10 (Very Open Ice)	0.2420	0.1070	0.221	0.1660
2/10 (Very Open Ice)	0.1220	0.1200	0.148	0.1150
3/10 (Very Open Ice)	0.0860	0.0450	0.071	0.0750
4/10 (Open Ice)	0.0540	0.0000	0.012	0.0000
5/10 (Open Ice)	0.1590	0.1600	0.099	0.1530
6/10 (Open Ice)	0.2000	0.1000	0.100	0.0090
7/10 (Close Ice)	0.2370	0.1990	0.212	0.2320
8/10 (Close Ice)	0.2140	0.1260	0.201	0.1440
9/10 (Very Close Ice)	0.1920	0.2610	0.178	0.2390
9/10~10/10 or +9/10 (Very Close Ice)	0.4720	0.5490	0.449	0.5830
10/10 (Compact/Consolidated Ice)	0.6160	0.5420	0.547	0.5870
Land	0.7200	0.7070	0.678	0.7580

Appendices

Appendix A. A python function used to calculate GLCM entropy

```
def generate_entropy(GLCM):
    e = numpy.finfo(float).eps

    return [
        np.sum(-np.multiply(GLCM[:, :, :, i], np.log(GLCM[:, :, :, i] + e)))
        for i in range(GLCM.shape[-1])
    ]
```

Appendix B. A Python function used to calculate the hidden layer size

```
def calculate_hidden_layer_size(input_layer_size, output_layer_size, user_defined=None):
    if user_defined == None:
        hidden_layer_size = ((input_layer_size + output_layer_size) * 2) // 3
    else:
        hidden_layer_size = user_defined

    if hidden_layer_size > 2 * input_layer_size:
        hidden_layer_size = 2 * input_layer_size

    if hidden_layer_size < 2:
        hidden_layer_size = 2

    return hidden_layer_size
```

Appendix C. A Python function for neural-network architecture

```
def NN(hidden_layer_size, input_layer_size, output_layer_size):
    model = Sequential()
    model.add(Dense(hidden_layer_size, input_dim=input_layer_size, activation="relu"))
    model.add(Dense(output_layer_size, activation="softmax"))

    model.compile(
        loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
    )
    model.summary()

    return model
```

Appendix D. A Python function for 1D-CNN architecture and its concatenation with the multi-layer neural network architecture.

```
def CNN(
    hidden_layer_size, conv_layer_size, cat_layer_size, output_layer_size, kernel_size
):

    conv_input = Input(shape=(conv_layer_size, 1), name="conv")
    cat_input = Input(shape=(cat_layer_size), name="cat")
    if cat_layer_size > 3:
        cat_layer = Dense(
            calculate_hidden_layer_size(cat_layer_size, output_layer_size),
            activation="relu",
        )(cat_input)

    conv_layer = Conv1D(
        64,
        kernel_size,
        activation="relu",
        input_shape=(conv_layer_size, 1),
        padding="causal",
    )(conv_input)
    conv_layer = Conv1D(64, kernel_size, activation="relu", padding="causal")(
        conv_layer
    )
    conv_layer = Conv1D(64, kernel_size, activation="relu", padding="causal")(
        conv_layer
    )
    conv_layer = Flatten()(conv_layer)

    if cat_layer_size > 3:
        # Turn off the layer if nothing to concatenate
        conv_layer = concatenate([cat_layer, conv_layer])

    hidden_layer = Dense(hidden_layer_size, activation="relu")(conv_layer)
    output_layer = Dense(output_layer_size, activation="softmax")(hidden_layer)

    model = Model(inputs=[conv_input, cat_input], outputs=[output_layer])

    model.compile(
        loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"]
    )
    model.summary()

    return model
```