MULTICLASS CLASSIFICATION OF FOREST COVER USING CONVOLUTION NEURAL NETWORK AND MULTISOURCE IMAGERY FOR NW ALBERTA

Bob Stankovic[1], Ridha Touzi[2],
[1] Wildfire Management Brunch, Alberta Agriculture and Forestry
[2] Canada Centre for Mapping and Earth Observation, Natural Resources Canada

ABSTRACT

Remote sensing has been increasingly used to derive land cover information by different automated classification algorithms. Unlike automated classifiers, artificial neural networks are difficult to parameterize and therefore it is really challenging to apply neural network in multiclass classification of natural landscape. In this paper, we report the methodology and result of a project aiming to develop some guides for multisource image (Sentinel-2 and Alos-2) classification utilizing Convolution Neural Network (CNN). Out project site is in NW Alberta near Zama Lake where the landscape mosaic is ravaged by forest fires. We compare the accuracy of CNN model output based on Sentinel 2 data and combination of Sentinel-2 and Alos2 Touzi scattering type phase $\phi\alpha s1$. The accuracy of our model using multisource data showed higher accuracy of 87% comparing to model output for Sentinel-2 data -83%.
Keywords: wetland, polarimetry, CNN

1. BACKGROUND

Artificial Intelligence (AI) is a general field that encompasses machine learning (ML) and deep learning (DL), but that also includes many more approaches that don't involve any learning. Deep learning is a specific subfield of machine learning: Machine learning is a technique in which you train the system to solve a problem instead of explicitly programming the rules. The deep in deep learning isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive (hidden) layers of representations. How many layers contribute to a model of the data is called the depth of the model.

.

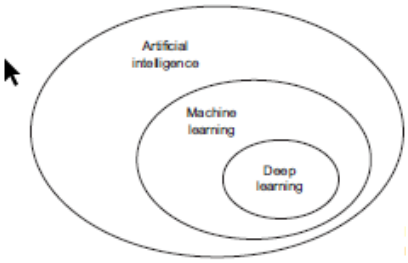

fig 1. Artificial intelligence, machine learning, and deep learning

A common machine learning task is supervised learning, in which you have a dataset with inputs and known outputs. The task is to use this dataset to train a model that predicts the correct outputs based on the inputs. The combination of the training data with the machine learning algorithm creates the model. Then, with this model, you can make predictions for new data. Among deep learning algorithms, Convolutional Neural Networks (CNNs) have gained popularity in computer vision and remote sensing fields, especially for image land cover classification, showing remarkable performance.
With neural networks, one starts with some random weights and bias vectors, make a prediction, compare it to the desired output, and adjust the vectors to predict more accurately the next time. The process continues until the difference between the prediction and the correct targets is minimal (figure 2). A major challenge in training neural networks is how long to train them, know when to stop the training and what accuracy target to set. It is mainly because of overfitting and underfitting scenarios. .

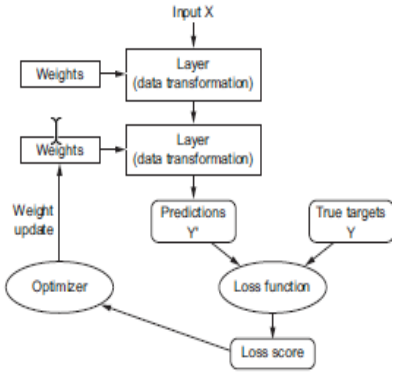

Figure 2. Forward and backward propagation Convolution Neural network

The objective and the central challenge of a neural network is to have a final model that performs well both on the data that we used to train it (e.g. the training dataset) and the new data on which the model will be used to make predictions. Large neural nets trained on relatively small datasets can overfit the training data. Too little learning and the model will perform poorly and underfit on training dataset and on new data. Too much learning and the model will perform well on the training dataset and poorly on new data and the model will overfit

the problem. Both cases (overfit and underfit) result in a model that does not generalize well. The ability to perform well on previously unobserved inputs is called generalization {13] pg.110. A compromise is to train on the training dataset but to stop training at the point when performance on a validation dataset starts to degrade. This simple, effective, and widely used approach to training neural networks is called "Early Stopping". To overcome over-fitting we may use more training data**.** There are two ways to approach an overfit model. We can reduce overfitting by changing the complexity of the network. The capacity of a neural network model, its complexity, is defined by both its structure in terms of nodes and layers and the parameters in terms of its weights. Therefore, we can reduce the complexity of a neural network while reducing number of weights and values of weights. Regularization methods are so widely used to reduce overfitting that the term "*regularization*" may be used for any method that improves the generalization error of a neural network model.
We can address underfitting by increasing the capacity of the model. Capacity refers to the ability of a model to fit a variety of functions. Increasing the capacity of a model is easily achieved by changing the structure of the model, such as adding more layers and/or more nodes to layers. Good Fit Model is a model that suitably learns the training dataset and generalizes well to the not seen dataset. A number of factors need to be considered when parameterizing neural networks, which include input data, training samples, output settings, networks architecture, initial weights, and training parameters.
In terms of software and infrastructure one needs to bear in mind that CNN is time/resource consuming. It is necessary to have a GPU to work with image and video data for most deep learning projects. One can build a deep learning model on a laptop/PC without the GPU as well, but then it would be extremely time-consuming to perform model run. The GPU issue can easily be solved with one of multiple Cloud Computing resources that provide GPUs either for free or for an extremely low cost.

In this project we intend to use multisource satellite data where we combine polarimetric
PALSAR (ALOS-2) data and Sentinel-2 data. Many studies pointed out that this combination (optical and Radar data) increase accuracy of delineation certain wetland features such as bogs and fens. Touzi decomposition [15] has become very popular and permit the promotion of the unique information provided by polarimetric SAR in various key applications, such as urban, wetland, and peatland mapping. In fact, the scattering type phase $\phi\alpha s1$ appears to be the most suitable polarimetric SAR parameter for bog and fen discrimination.
The objective of our study was to develop a guideline for how to use CNN when doing land cover classification of multisource data and how to improve accuracy while setting the internal parameters of the neural networks.

2. RESEARCH METHODOLOGY

2.1 Study Area and Data

Figure 3. Project area

Our test site is located in the north-west corner of Alberta provincial border. This project area, is mostly Northern Mixedwood. Black and White Spruce dominate the landscape. Bogs are widely spread features and landscape is shaped by legacy of old and recent burns.

2.2. DATA PREPARATION

Target scattering decompositions [16] have become the most popular methods for polarimetric image classification. In our previous work, we have found that adding a Touzi phase $\phi\alpha s1$ decomposition component to multispectral data improve the accuracy of image classification work. A Sentinel-2 image obtained on August 8, 2020 and fully polarimetric Alos-2 acquired on June 27 2019 courtesy of Japanese Space Agency (JAXA) were used to derive land cover information. In our approach to land cover classification with CNN, we applied the same model to a multispectral Sentinel-2 dataset and another dataset containing Sentinel 2 and Touzi scattering type phase. All Sentinel-2 bands including Touzi phase $\phi\alpha s1$ component were resampled to 20 m. We converted all our training polygons for the area to raster and stacked with multispectral image. The values of the raster represented different cover types matching Alberta Satellite Land Classification (ASLC) codes. For instance Black Spruce polygon has ASLC code of 51 while Aspen has 55. Our stack had in total 10 bands where nine are features (Sentinel2 bands) and one is labels (raster of training data).
Next, we generated regularly spaced point shapefile (figure 2) which we used to read values of all bands in the stack. We had 17 different cover types (ASLC classes) including burns-41, different pure coniferous (Sb-51, Pl-52, Sw-53), mixed conifer-54, decidusous-55 and water 80, mixedwood and wetlands

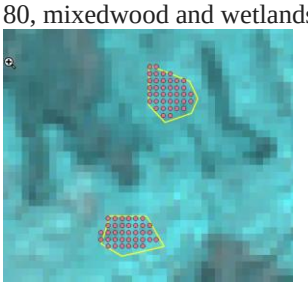

figure 4. Regularly spaced points 20 m

Then we converted that shapefile to CSV file. Finally, CSV file and imagery were uploaded to Colab python environment into Google Cloud for processing. Instead of using regular ASLC codes (ex. Sb is 51), we added another field ASLC_ to order them from 1 to 17. For instance, Burns with ASLC code 40 were assigned id 1 in "ASLC_" field and last young Pine with ASLC code 552 received id 17. Our predictors (features) are bands B2, B3..B12 and our response variable (label) is "ASLC_".

X	Y	Z	id	B2	B3	B4	B5	B6	B7	B8	B8A	B11	B12	Phi	ASLC	Latitude	Longitude	ASLC_
371348.0284	6569848.128	0	2940	291	590	383	248	582	1669	2027	2020	2166	1266	0.867016	552	371348.0284	6569848.128	17
371368.0284	6569848.128	0	2941	277	624	410	270	640	1744	2084	2124	2277	1310	0.648082	552	371368.0284	6569848.128	17
371328.0284	6569828.128	0	2968	296	612	393	255	626	1757	2143	2166	2337	1286	0.913669	552	371328.0284	6569828.128	17
371348.0284	6569828.128	0	2969	299	619	438	288	621	1775	2129	2297	2262	1286	0.828769	552	371348.0284	6569828.128	17
371368.0284	6569828.128	0	2970	270	643	393	269	634	1779	2125	2192	2264	1341	0.722980	552	371368.0284	6569828.128	17

2.2.1 DATA BALANCING

Balancing among classes is really important when it comes to accuracy assessment. Our training polygon samples range from 50 to 1000 regularly spaced points. When we look at the unique classes we see that close White Spruce (Sw) stands have in total 1085 samples, while graminoid wetland ,Black Spruce (Sb)- lichen and young Pine (82,87, 552) stands have barely 50 samples. To make them balanced, we down-sample those with high and up-sample those with low number of training samples. Balancing will result with each cover type having similar amount of training pixels(~ 500). Now we can split the data into training and validation.

2.2.2 SPLITTING THE DATA INTO TRAIN AND TEST SUBDATASET

In order to evaluate the performance of the model at a later stage and see how our model performs on unseen data, we need to create a validation set. This is done by partitioning the training set data using various ratios, but in this case we opted for 80:20. Before splitting the data we do shuffling so that different "ASLC_" codes are randomly spread. Now, we will split the data for training and validation. This is done to make sure that the model has not seen the test data and it performs equally well on new data. Otherwise, the model will overfit and perform well, only on training data. The validation data set will be used to optimize the model parameters during training process.

2.2.3 NORMALIZING/ SCALING

In the next step we normalize/scale the data which is also important so that all features (bands) are treated equally. Neural networks are sensitive to the distribution of data. Bands with reflectance e values (7,8 and 9) will play larger role in fitting the model unlike bands (B2 or B3) with lower reflectance values.

Therefore we need to scale them in a range between 0-1 or -1 to 1. There are many ways to do it, but here we will normalize it under standardized normal distribution.

→ z = (value – mean)/standard_deviation.

2.2.4 ENCODING

While doing multiclass classification, hot encoding is necessary. We need to transform labels into an array, because labels can be strings or numbers and it is called 'hot encode' where label 1 and label 2 would look like [1. 0. 0. 0. 0. 0. 0.] [0. 1. 0. 0. 0. 0. 0.]

2.3. DEFINING MODEL ARCHITECTURE

The simplest deep learning neural network consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. **A** hidden layer is any layer between the input (first) layer and output (last) layer. There can be multiple hidden layers. We will use the **sequential** model (figure 5) and add the layers one after another one which are fully connected. The *Dense* function in Keras (python library) constructs a fully connected neural network layer, automatically initializing the weights and biases. A hyper-parameter is a parameter whose value is set before the learning process begins. Essentially, the hyper-parameters are crucial to the performance, speed, and quality of the machine learning models and they play a massive part in deciding how good the predictions will be. Often, we choose them based on our experience and through the trial-and-error process. It is very manual and doesn't guarantee the best for our models. A good idea is to pick these values based on existing research/studies or to keep experimenting with the values until you find the best match but this can be quite a time consuming process. Hence they should be optimized. There are some basic techniques such as Grid Search, Random Search and also some more sophisticated techniques such as Bayesian Optimization, Evolutionary Optimization and etc. The key inputs *parameterization* include the hyperparameters that will be tuned such as "number of hidden layers", "neurons per layer", "dropout rate", activation function, optimizer, "learning rate", "batch size" and etc. The Dense function arguments include number of nodes in a layer. The first layer in Keras

models (figure 5) need to specify the input dimensions and number of nodes (512). In this case our input dimensionality is 9, (number of Sentinel-2bands). Since we're constructing the output layer, and we have 17 classes so this value for nodes in output layer is 17. This means we are working with multiclass classification. However activation function for output layer is '*softmax*' which is suitable for categorical output. Then beside input and output layer we have three hidden layers. This model (figure 5) was chosen after trial and error approach and using Grid Search. "Grid Search" (scikit-learn Python library) was used to find the optimal set of parameters that yields the best performance for our model.

2.3.1. TRAIN THE MODEL AND MAKE PREDICTIONS

Once the network architecture is defined, and before model is compiled one still has to choose "*Loss function"*. For loss function we choose '*categorical_crossentropy'* which is default for multiclass classification. For *Optimizer* parameter we use *Adam* which determines how the network will be updated based on the loss function. In our model we used EarlyStopping technique to reduce overfitting. Early stopping prevents overtraining of our model by terminating the training process if it's not really learning anything. We also applied ModelCheckpoint callback to save our model as a checkpoint file (in hdf5 or h5 format) to disk after each successful epoch.

```
new_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               5632

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 17)                1105

=================================================================
Total params: 179,217
Trainable params: 179,217
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. A summary of our sequential model.

The architecture of the model takes weeks for the making, therefore the first model will never make it to predictions. I started with only the input and output layer to check the base accuracy. I kept on adding the layers and changing the number of convolutions in each layer to reach somewhere near the desired results. After defining all parameters we need to fit the model, while running it for certain number of *epochs*

2.4 EVALUATE MODEL ESTIMATION OF PERFORMANCE

The three main metrics used to evaluate a classification model are accuracy, precision, and recall. Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions. We typically plot the accuracy and loss plots between training and validation data for the one last time (figure 8, 9) before we predict model onto a not seen image.

2.5 PREDICTION

Once we have a model that we are happy with, then we can use it on a new image for prediction. The key is that we have to make sure that we work with the same bands. If we used 9 bands to train a model, then we need to use the same bands for prediction as well. Also, if we trained model on normalized data we have to use the same parameters as we normalize new unknown image. Therefore, our image for prediction needs to be scaled to the same standardized normal distribution parameters as well.

3. RESULTS AND DISCUSSION

It is clear that the accuracies of land cover classification vary across different neural models associated with different internal parameter settings. Our results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Stochastic machine learning algorithms use randomness during learning, ensuring a different model is trained each run. Stochastic does not mean random. Stochastic machine learning algorithms are not learning a random model. They are learning a model conditional on the historical data you have provided. Instead, the specific small decisions made by the algorithm during the learning process can vary randomly. The impact is that each time the stochastic machine learning algorithm is run on the same data, it learns a slightly different model. In turn, the model may make slightly different predictions, and when evaluated using error or accuracy, may have a slightly different performance.

```
Epoch 00197: val_loss improved from 0.44476 to 0.44064, saving model to my_best_np_CVSearch_model_NORMal_3.hdf5
48/48 [==============================] - 0s 10ms/step - loss: 0.3936 - accuracy: 0.8276 - val_loss: 0.4406 - val_accuracy: 0.8274
Epoch 198/200
47/48 [============================>.] - ETA: 0s - loss: 0.3854 - accuracy: 0.8408
Epoch 00198: val_loss did not improve from 0.44064
48/48 [==============================] - 0s 9ms/step - loss: 0.3854 - accuracy: 0.8407 - val_loss: 0.4637 - val_accuracy: 0.8147
Epoch 199/200
47/48 [============================>.] - ETA: 0s - loss: 0.3801 - accuracy: 0.8436
Epoch 00199: val_loss did not improve from 0.44064
48/48 [==============================] - 0s 8ms/step - loss: 0.3805 - accuracy: 0.8435 - val_loss: 0.4567 - val_accuracy: 0.8303
Epoch 200/200
43/48 [=========================>....] - ETA: 0s - loss: 0.4005 - accuracy: 0.8287
Epoch 00200: val_loss did not improve from 0.44064
48/48 [==============================] - 0s 9ms/step - loss: 0.4021 - accuracy: 0.8265 - val_loss: 0.4557 - val_accuracy: 0.8260
```

Figure 6. Model accuracy using only Sentinel 2 data

```
Epoch 167/200
89/93 [===========================>..] - ETA: 0s - loss: 0.2226 - accuracy: 0.9223
Epoch 167: val_loss did not improve from 0.38014
93/93 [==============================] - 1s 8ms/step - loss: 0.2252 - accuracy: 0.9206 - val_loss: 0.3974 - val_accuracy: 0.8721
Epoch 168/200
86/93 [==========================>...] - ETA: 0s - loss: 0.2180 - accuracy: 0.9209
Epoch 168: val_loss did not improve from 0.38014
93/93 [==============================] - 1s 8ms/step - loss: 0.2186 - accuracy: 0.9213 - val_loss: 0.3903 - val_accuracy: 0.8721
Epoch 169/200
91/93 [============================>.] - ETA: 0s - loss: 0.2236 - accuracy: 0.9194
Epoch 169: val_loss did not improve from 0.38014
93/93 [==============================] - 1s 8ms/step - loss: 0.2240 - accuracy: 0.9191 - val_loss: 0.4022 - val_accuracy: 0.8707
Epoch 170/200
87/93 [===========================>..] - ETA: 0s - loss: 0.2207 - accuracy: 0.9245
Epoch 170: val_loss did not improve from 0.38014
Restoring model weights from the end of the best epoch: 150.
93/93 [==============================] - 1s 8ms/step - loss: 0.2198 - accuracy: 0.9236 - val_loss: 0.3841 - val_accuracy: 0.8714
Epoch 170: early stopping
```

Figure 7. Model accuracy utilizing Sentinel 2 and Touzi decomposition (Phi) data

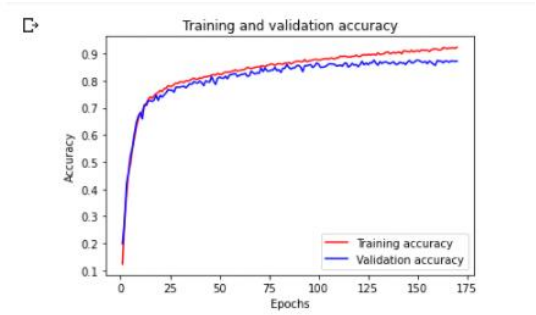

Fig 8 a). Accuracy for Sentinel 2 & Touzi phase $\phi\alpha s1$

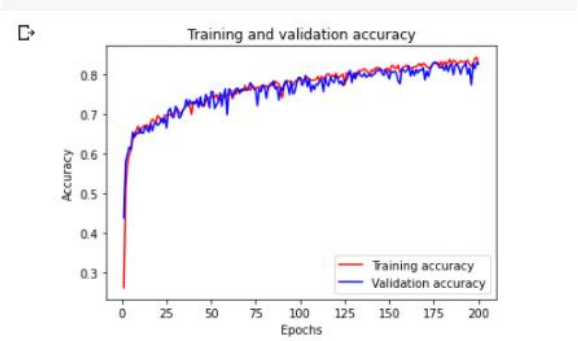

Fig 8.b. Accuracy for Sentinel 2 model

The accuracy of our model using multisource data showed higher accuracy of 87% comparing to model output for Sentinel-2 data only -83%. Reviewing the learning curves, we can see that the overfitting has become an issue (figure 8a). Although the learning curves suggested that the model for multisource data had overfit the training dataset. Learning continues well past 100 epochs, although may show signs of leveling out towards the end of the run. The results suggest that further augmentation or other types of regularization added to this configuration may be helpful. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. In another word an overfitted model performs well on the training set but poorly on the test set. This means that the model can't seem to generalize when it comes to new data. It is noticeable that our model in fig. 8a exhibits overfitting.

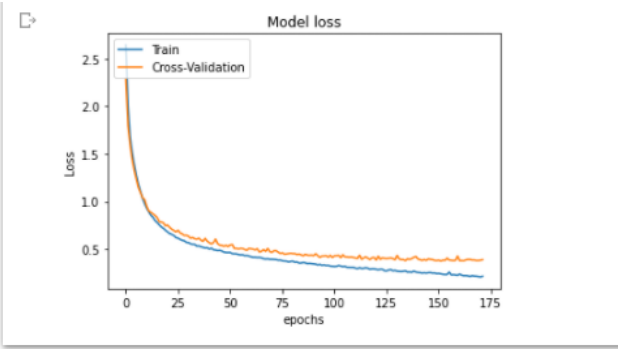

Fig 9 a). Loss function for Sentnel-2 & Touzi phase $\phi\alpha s1$.

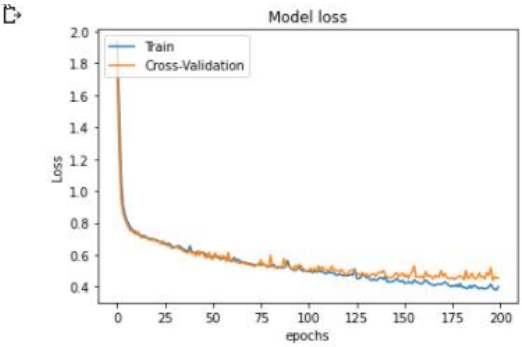

Fig. 9.b. Loss function for Sentinel 2

As you can see (Fig. 9 a) after the early stopping state the validation-set loss start stagnating or level off, but the training set value keeps on decreasing. In an accurate model both training and validation, accuracy must be decreasing.

Confusion matrix helps us evaluate how our model performed, where it went wrong and offers us guidance to correct our path. So, we use it in combination with other evaluation metrics which gives us a complete picture of the result. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making. Precision tells us how many of the correctly predicted cases actually turned out to be positive. One of the observations was that initial model that did not take in consideration Normalizing and balancing data, produced significantly lower accuracy for these 17 landcover classes.

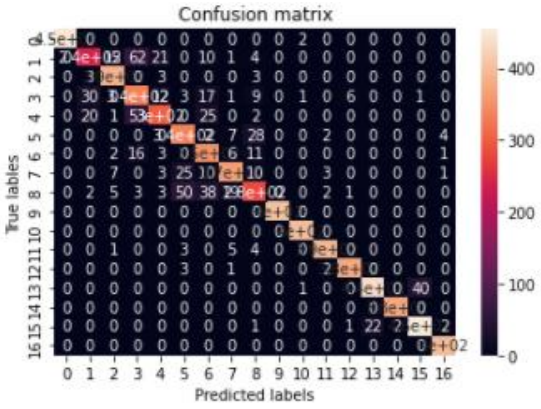

Figure 10. Confusion matrix for Sentinel-2 and Touzi phase $\phi\alpha s1$ component

While observing classification report or confusion matrix we can find out which class the model performed bad out of the given 17 classes. We see that the classifier is underperforming for class 2, 5 and 9. These are Black Spruce (ASLC- 51), Mixed Conifer (ASLC- 54) and Mixedwood (58). Tipically Black Spruce get mixed with White Spruce and they easily get mixed. After closer inspection our product looks really good; model perfectly finds all these shruby wetlnads around water bodies, Black Spruce bogs and burned scars.

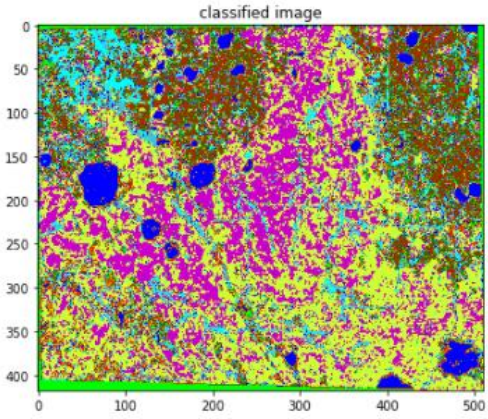

Figure 10. Predicted output

4. SUMMARY AND CONCLUSIONS

The accuracy of our model using multisource data showed higher accuracy (87%) than model output for Sentinel-2 data exclusively (83%). Classifier is underperforming for Black Spruce class, mixed coniferous and mixedwood cover types which was expected. Touzi scattering type phase $\phi\alpha s1$ appears to be improving our classification accuracy while increasing accuracy of some wetland types such as bogs and fens.

REFERENCES

1. Bischof, H., Schneider, W., Pinz, A.J.,1992. Multispectral classification of Landsat-images using neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 30(3), pp. 482-490.

2. Chen, K., Yen, S., Tsay, D.,1997. Neural classification of SPOT imagery through integration of intensity and fractal information. *International Journal of Remote Sensing*, 18(4), pp. 763-783.

3. Civco, D.L.,1993. Artificial neural networks for land-cover classification and mapping. *International Journal of Geographical Information Systems*, 7(2), pp. 173-186.

4. Foody G.M., Arora, M.K.,1997. An evaluation of some factors affecting the accuracy of classification by an artificial network. *International Journal of Remote Sensing*, 18(4), pp. 799-810.

5. Jensen, J. R.,2005. *An Introductory Digital Image Processing: A Remote Sensing Perspective* (3rd). Prentice Hall, New Jersey,pp. 296-301.

6. Kanellopoulos, I., Wilkinson, G.G.,1997. Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, 18(4), pp. 711-725.

7. Kavzoglu, T., Mather, P.M.,2003. The use of backpropagating artificial neural networks in land cover classification.*International Journal of Remote Sensing*, 24(23), pp. 4907-4938.

8. Serpico, S.B., Bruzzone, L., Roli, F.,1996. An experimental comparison of neural and statistical non-parametric algorithmsfor supervised classification of remote sensing images. *Pattern Recognition Letters*, 17(13), pp. 1331-1341.

9. Shupe, S.M., Marsh, S.E.,2004. Cover- and density-based vegetation classifications of the Sonoran desert using Landsat TM and ERS-1 SAR imagery. *Remote Sensing of Environment*, 93(2004), pp. 131-149.

10. Yang, X.,2002. Satellite monitoring of urban spatial growth in the Atlanta metropolitan area. *Photogrammetric Engineering &Remote Sensing*, 68(7), pp. 725-734.

11. François Chollet MANNING SHELTER ISLAND, Deep Learning with Python,

12. **https://realpython.com/python-ai-neural-network/**

13. Ian Goodfellow, Yoshua Bengio, Aaron Courville .Cambridge, MA:MIT Press, 2017. Deep Learning (Adaptive Computation and **Machine Learning series) Illustrated Edition**

14. R. Touzi1, K. Omari1, B. Sleep2, and X. Jiao Scattered and received wave polarization optimization for enhanced peatland classi_cation and fire damage assessment using polarimetric PALSAR

15. [18] R. Touzi. Target scattering decomposition in terms of roll invariant target parameters. *IEEE Trans. Geoscience Rem. Sens.*,45, No. 1:73–84, 2007.

16. [19] R. Touzi, A. Deschamps, and G. Rother. Phase of target scattering for wetland characterization using polarimetric C-band SAR. *IEEE Trans. Geoscience Rem. Sens.*, Vol. 47, No. 9:3241–3261,September, 2009