CRIM

CENTRE DE
RECHERCHE INFORMATIQUE
DE MONTRÉAL

Deliverable 4

Establishment of a drone-acquired imaging processing pipeline for the recognition of forest species by deep learning techniques

Report prepared for the Canadian Wood Fibre Centre of the Department of Natural Resources

Version 1

David Byrns
Advisor

Pierre-Luc St-Charles
Post-doctoral researcher

Francis Charette-Migneault
Research agent

Mario Beaulieu
Senior Research agent

Vision and Imaging Team

March 15, 2019

TABLE OF CONTENTS

INTRODUCTION **4**

1. REQUIREMENT DEFINITIONS AND PERFORMANCE INDICATORS **4**

2. DATA **4**

2.1 Petawawa experimental forest **4**

2.2 Drone acquisition mission **5**

2.2.1 Study site 5

2.2.2 Imaging sensors 7

2.2.3 Flight planification 7

2.2.4 The importance of overlaps between images 10

2.2.5 Completed flights 10

2.2.6 Photogrammetric processing 11

2.2.6.1 Initial processing 11

2.2.6.2 Densification 12

2.2.6.3 Orthomosaic generation 12

2.2.7 Limitations 14

2.2.7.1 Changing light conditions 14

2.2.7.2 Automatic exposures 15

2.2.8 Availability 16

2.3 Ground truth data generation **16**

3. MODEL TRAINING BASED ON THE PETAWAWA EXPERIMENTAL FOREST **18**

3.1 Neural network architecture tests **19**

4. FORÊT-OUAREAU TREE SPECIES CLASSIFICATION EXPERIMENTS **23**

4.1 Radiometry correction **23**

4.2 Input data variations **24**

4.3 Data limitations **24**

4.4 Classification model architecture variations **25**

4.5 Final results **25**

5. CLOUD INFRASTRUCTURE **27**

5.1 Platform architecture **28**

5.2 Processing **29**

5.2.1 Framework functionalities of the cloud service 31

5.3 Visualization **32**

6. TESTS 35
6.1 Cloud service requests 35
6.2 Additional experiments 40
7. OUTREACH 43
8. FINAL DISCUSSION AND CONCLUSION 43
9. REFERENCES 43
APPENDIX **45**

INTRODUCTION

This fourth and final report covers all the activities of the project and therefore includes sections of previous reports as well as the fourth and fifth activities, which are tests, outreach and final report.

1. REQUIREMENT DEFINITIONS AND PERFORMANCE INDICATORS

The main requirements of this project were the training of neural network models on the Petawawa experimental forest dataset, the exploration and testing of different neural network architectures, and the validation of our tree species classification results. Moreover, the requirements involved the acquisition of new imagery with a drone, the production of an orthomosaic with this imagery, and the testing of our models over the produced orthomosaic. Finally, the image processing pipeline had to be packaged and validated in a cloud infrastructure.

The performance (or progression) of our work was measured according to these indicators:

- The number of models and architectures tested;
- The number of cloud-based services deployed on the CRIM infrastructure;
- The number of tests done over the cloud infrastructure;
- The number of conference article written and published.

2. DATA

2.1 Petawawa experimental forest

An agreement with the Canadian Wood Fibre Centre helped us to obtain an imagery dataset from the Petawawa experimental forest used to train our models. Here, a summary of the dataset is provided:

- 1.16 TB of raw images and binary files
- Lidar and optical data obtained in 2009, 2012 and 2013 with resolutions ranging from 40 cm/pixel to 15 cm/pixel
- A shapefile containing 10k polygons that are defined manually or with a segmentation assistance tool, alongside tree species annotations (based on 2009 RGB-NIR data at 40 cm/pixel)

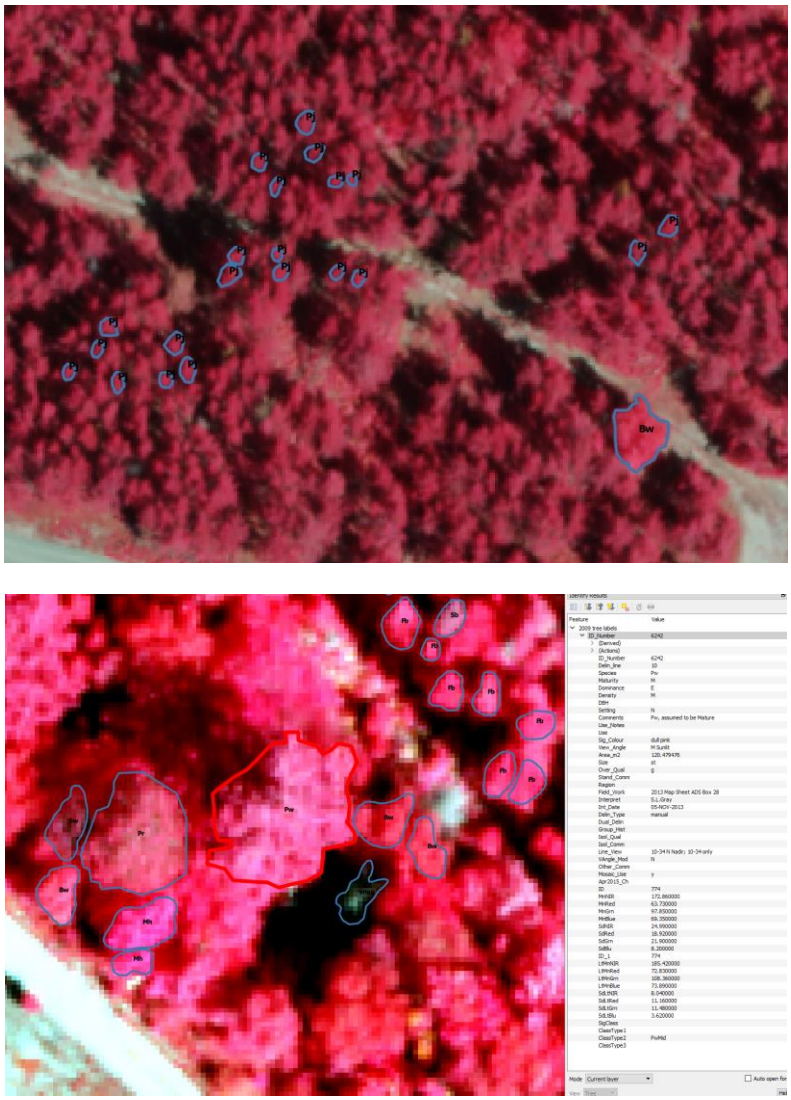

Figure 1 - Example of annotated crowns on the 2009 Petawawa orthomosaic.

2.2 Drone acquisition mission

2.2.1 Study site

The mission took place on Thursday, August 16, 2018, in the regional park of Forêt-Ouareau near the suspension bridge area. Our initial goal was to find an area with a distribution of tree species comparable to the one of the Petawawa forest. The site was chosen by FP Innovations and Pegasus a week before the scheduled flight.

The Forêt-Ouareau authorities were contacted by Pegasus's pilot in order to obtain flight authorizations for the selected area (Figure 2). The Minister of Forests, Wildlife and Parks was also contacted to present the research project.

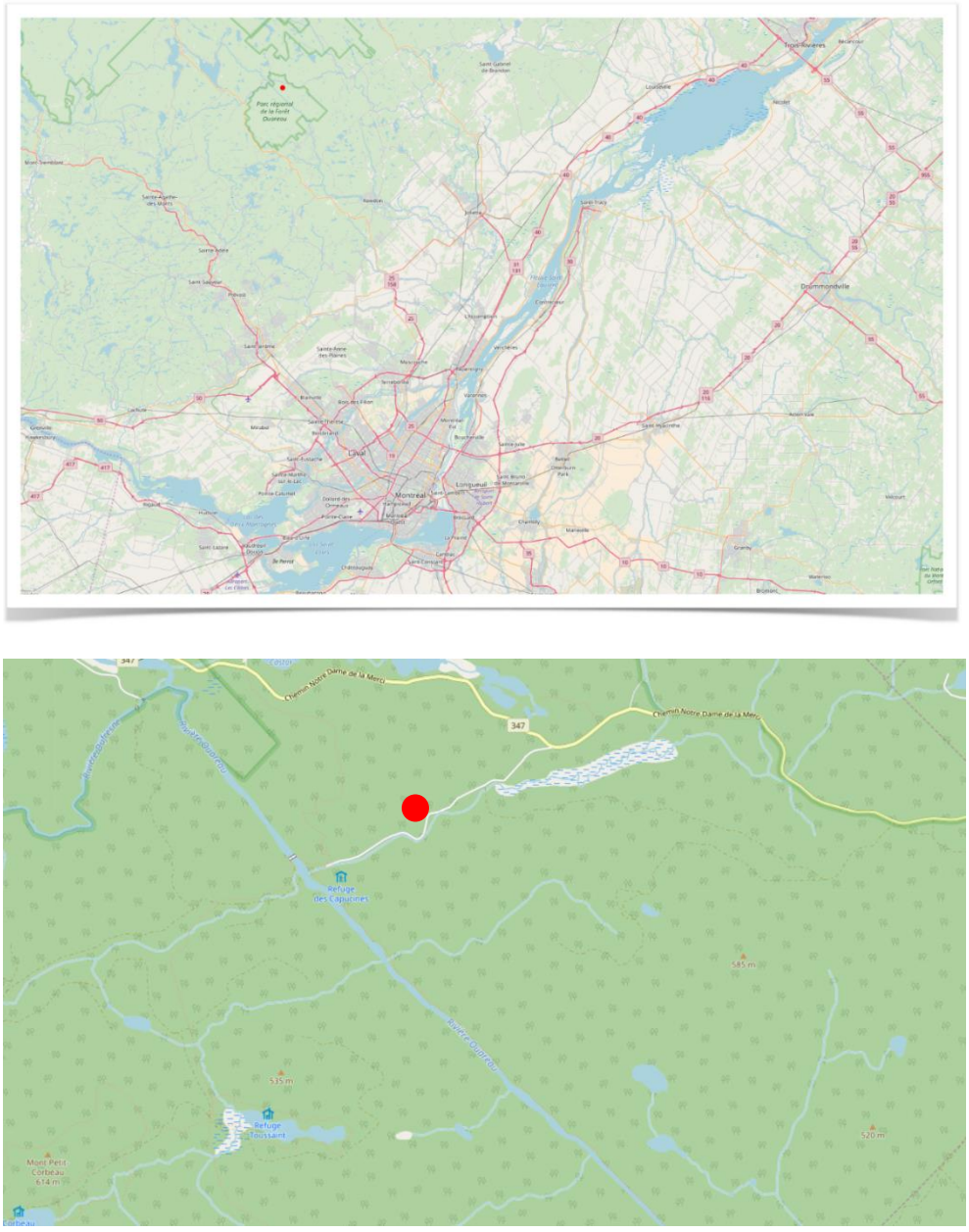

Figure 2 - Location maps of the study area in the regional park of Forêt-Ouareau

Figure 3 - The DJI Matrice 600 drone with its two visible and near infrared sensors

2.2.2 Imaging sensors

Two sensors were integrated to the drone. The first was a stabilized DJI X5S RGB camera (18 MPx) controlled by the drone, and the second a MicaSense RedEdge 3 (1.2 MPx) multispectral camera. The latter is autonomous and was set to acquire an image every two seconds. The images of both cameras are acquired with the drone's corresponding geographical position (latitude, longitude and altitude).

2.2.3 Flight planification

Two applications were used: DJI Ground Station Pro (GSP) and Maps Made Easy Map Pilot (MP). Map Pilot has a terrain tracking option, which allows the drone to keep its height 90 meters above the ground all along the flight even if the terrain is moving in altitude. Unfortunately, the software was not able to handle both sensors. It was therefore necessary to manually trigger the image every two seconds during flight number 3 (Figure 8).

Figure 4 - Illustration of the Map Pilot application tracking based on the SRTM 30 meters

Flights number 1, 2 and 4 were successfully completed with the GSP application but without field monitoring. The areas covered by these flights were generally low relief, and the impact on the images was limited.

At an altitude of 90 meters, the spatial resolution of visible imagery was 3.9 cm/pixel on the ground (and about 2.7 cm/pixel at the treetops). For near-infrared imaging, the spatial resolution was 6.3 cm/pixel on the ground (and about 4.2 cm/pixel at the treetops). Examples of the RGB and multispectral imagery acquired are presented in Figure 5 and Figure 6.

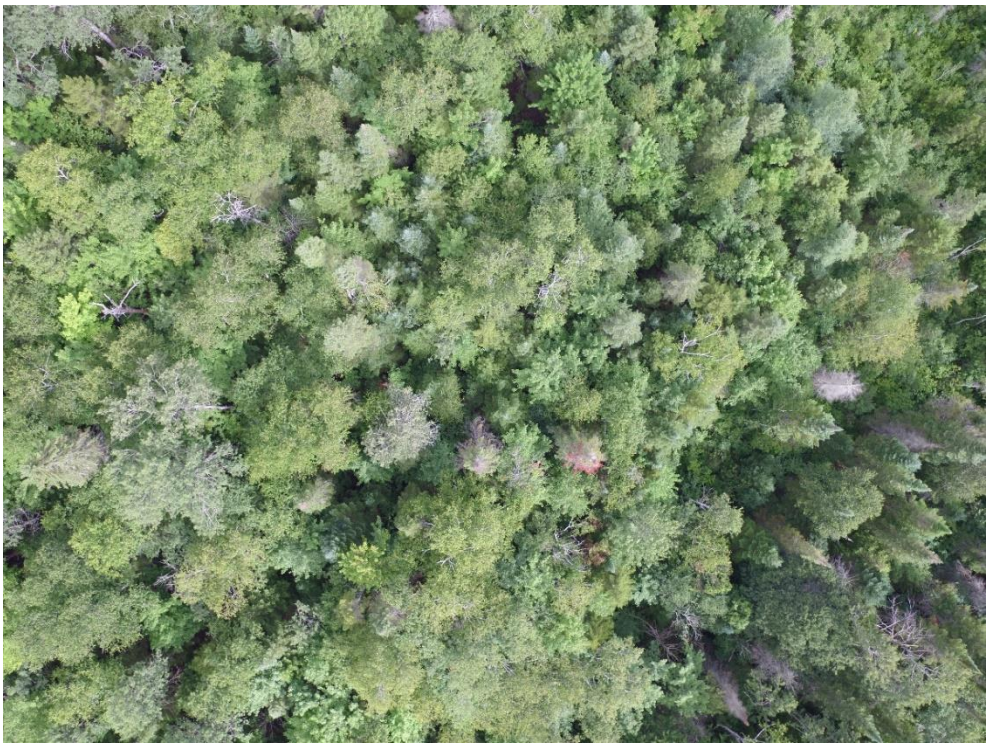

Figure 5 – Visible (RGB) image acquired by the DJI X5S

Here is an example of multispectral images (Blue 475 nm, Green 560 nm, Red 668 nm, PIR 840 nm, RE 717 nm):

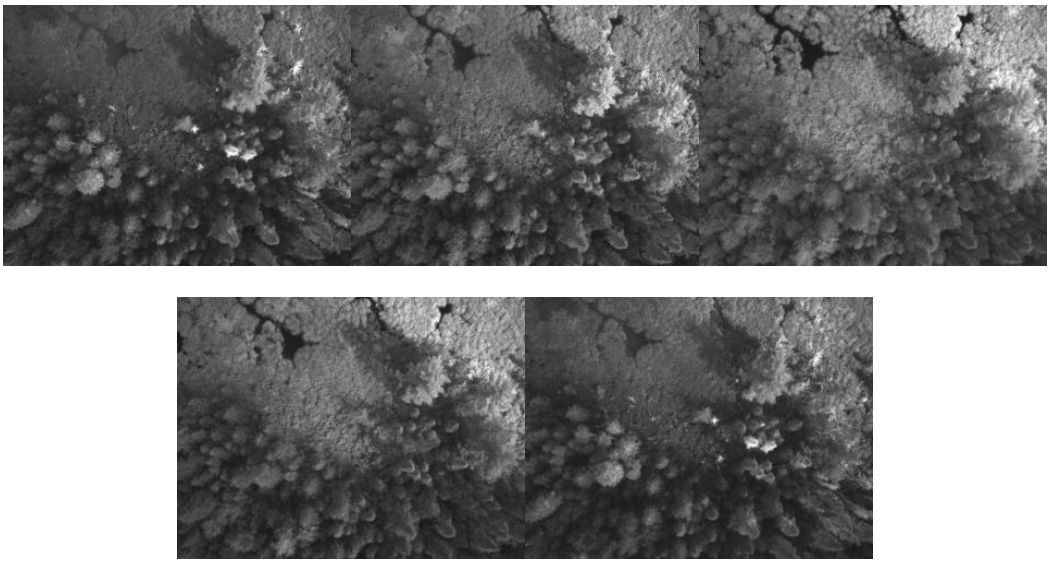

Figure 6 – Five-band image acquired by the MicaSense RedEdge 3

2.2.4 The importance of overlaps between images

When planning a nadir imaging acquisition flight, it is recommended that 80% longitudinal overlap be provided between the images (on the flight line) and 60% lateral overlap (between the flight lines). The forest environment, consisting of a multitude of vertical elements (trees), is one of the most difficult land uses to model by photogrammetry. In order to better model the study area, recoveries were increased to 85% of longitudinal and frontal recovery at the level of the crowns, which represents about 8 meters between each picture and 10.5 meters between each flight line.

The trees in the study area measure an average of 30 meters in height, which corresponds to one-third of the flying height of the drone relative to the ground. The distance between the crowns and the drone being reduced (60 m), it was necessary to compensate for this loss by a greater recovery. The recoveries in flight planning applications were therefore defined at 90% longitudinal and frontal coverage.

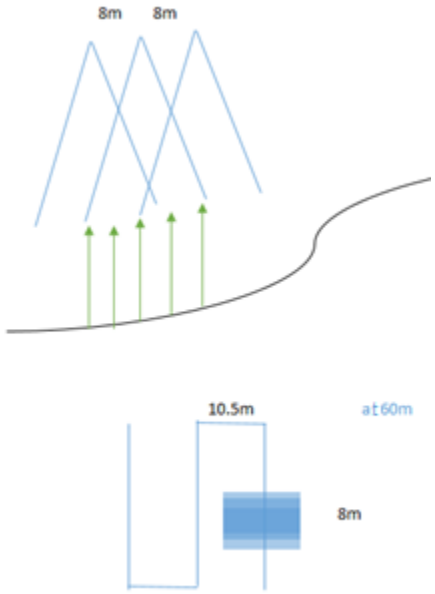

Figure 7 - Front and side overlapping based on tree heights

2.2.5 Completed flights

The four flights were conducted in this area at an altitude of 90 meters above ground level and generated 20118 photos, or 56.1 GB of visible and multispectral imagery.

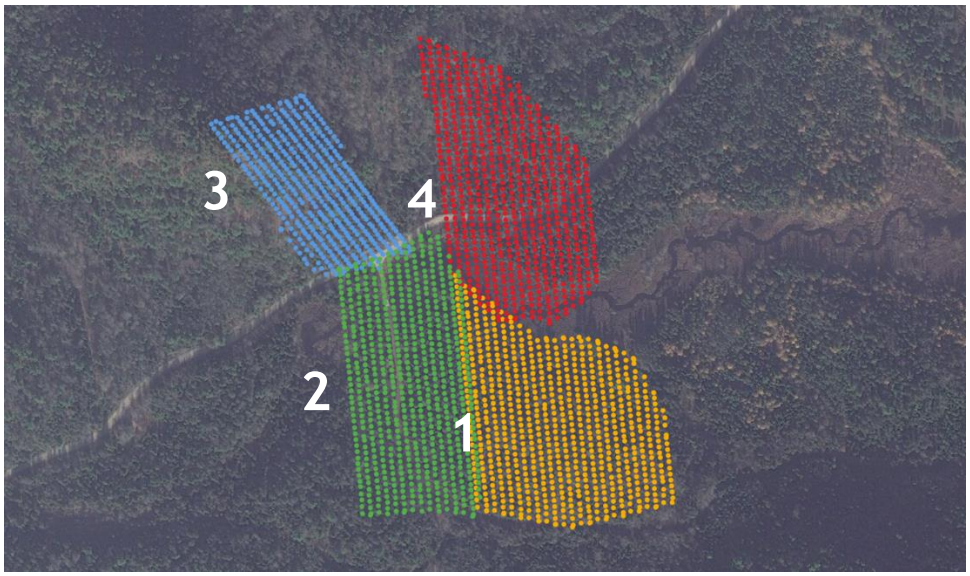

Figure 8 - Positions of images acquired for each flight

Table 1 - Flight statistics

FLIGHT NUMBER	NUMBER OF IMAGES (VISIBLE / MULTISPECTRAL)	FLIGHT TIME (MIN)	FLIGHT SPEED	TERRAIN FOLLOWING	WEATHER CONDITIONS
1	743 / 4125	26	3 m/s	No	Scattered clouds
2	652 / 4030	22	3 m/s	No	Scattered clouds
3	470 / 4875	18	2 m/s	Yes	Clear sky
4	683 / 4540	23	3 m/s	No	Clear sky

2.2.6 Photogrammetric processing

The processing of the imagery was done in two stages, first the visible imagery, then the multispectral images. The software used was Agisoft's Photoscan.

The main goal of the imagery processing was to produce an orthomosaic and a point cloud in visible and infrared wavelengths. The processing is divided in three steps: initial processing, densification and orthomosaic generation.

2.2.6.1 Initial processing

All the images acquired during the four flights were loaded in the software. The initial processing step was to use the images to compute the keypoints and find the matches between the images. An Automatic Aerial Triangulation (AAT) was used at this step.

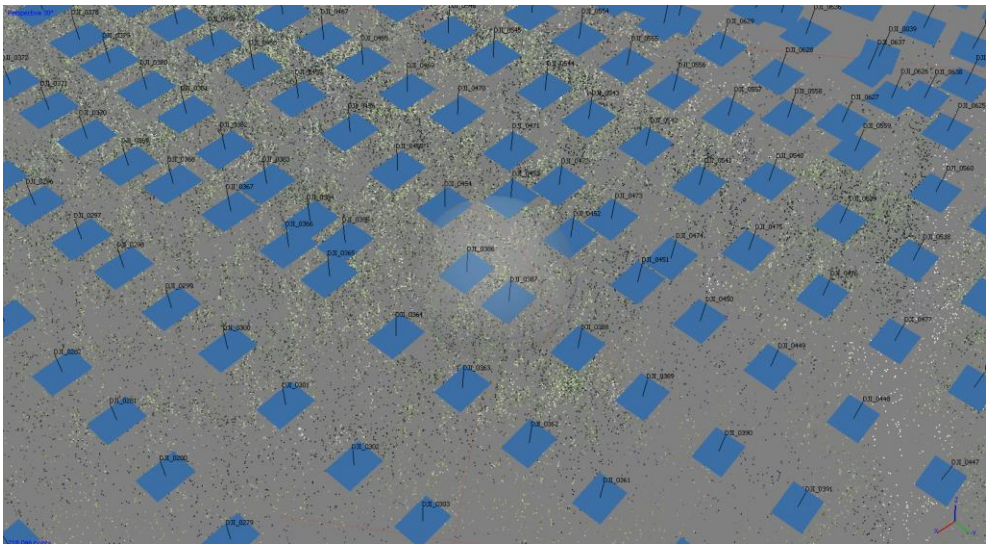

Figure 9 - Result of the initial processing step with visible images

2.2.6.2 Densification

Once the initial processing was done, the software knew the position of each picture and could densify the point cloud with 3D points that reconstructed the surface model. Each point has the X, Y, Z position and color information stored.

Figure 10 - Result of the 3D point cloud with visible imagery

2.2.6.3 Orthomosaic generation

The orthomosaic is the result of the orthorectification of the stitched images. The perspective distortions are removed from the images using the 3D Point cloud. The

orthomosaic can have a few artifacts, especially when the surveyed area is covered by a lot of trees. Even if the drone can deliver very high-resolution imagery, the trees have an infinite amount of details (trunk, branches and leaves) and they can only be approximately modeled.

Figure 11 – Sample crop of the visible spectrum orthomosaic generated by Agisoft PhotoScan

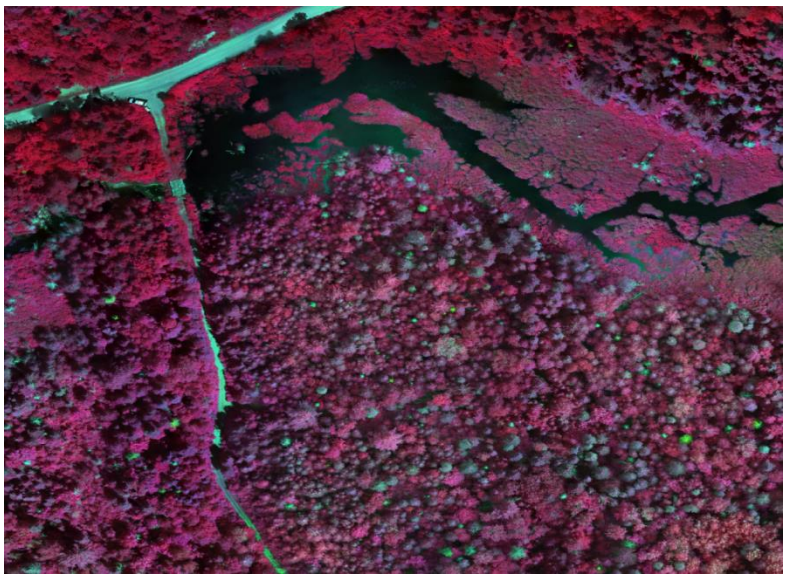

Figure 12 - Extract of the NIR-RGB orthomosaic generated by Agisoft PhotoScan (only the near infrared, red, and green channels are shown)

However, the artifacts are less present on the multispectral orthomosaic. This can be explained by the fact that the multispectral camera is a global shutter camera, which

guarantees an image quality superior to the DJI X5S (rolling shutter), even if it has a lower resolution.

Figure 13 - Extract of the colored compound of the multispectral orthomosaic

2.2.7 Limitations

2.2.7.1 Changing light conditions

The weather conditions changed during the day of the flight: cloudy in the morning and clear in the afternoon. The flights done in the morning were impacted by the changing light conditions. Unfortunately, we did not have access to the DLS sensor (DownLight Sensor) to measure the incident radiation for each picture. Figure 14 shows the impact of the clouds on the same area on two images taken three minutes apart.

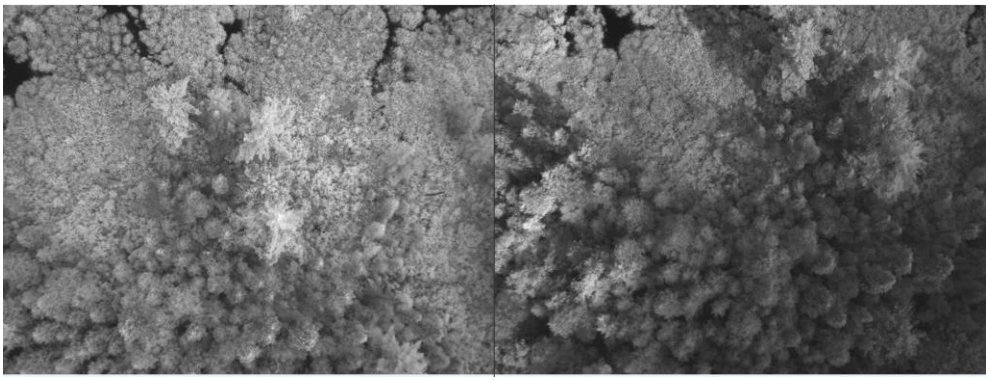

Figure 14 - Changes of light conditions on the same area between two flight lines

We can also observe the impact on the final orthomosaic of the radiometric variations between the flight lines (Figure 15).

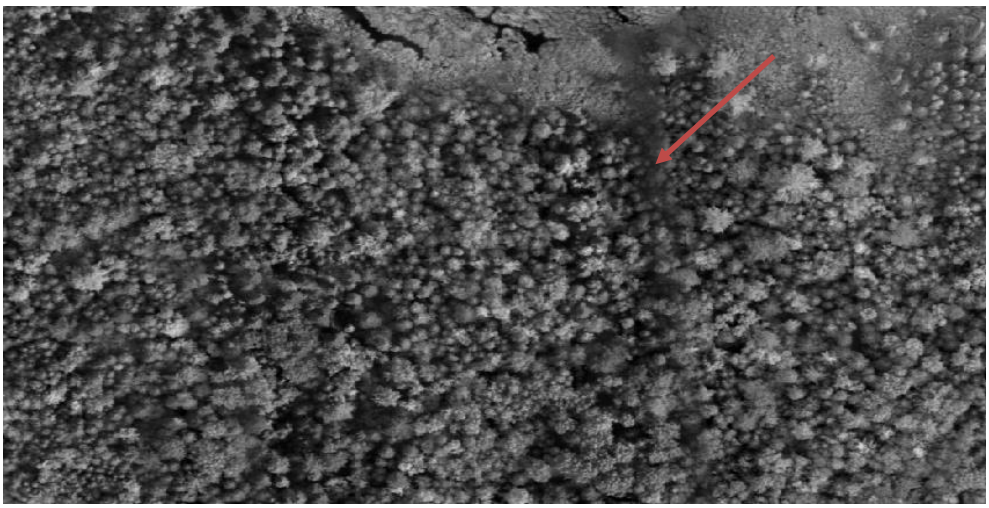

Figure 15 - Radiometric variations between flight lines

2.2.7.2 Automatic exposures

MicaSense, the manufacturer of the RedEdge camera, recommends leaving the exposure setting in automatic mode. There is a way to set the exposure of the camera, but it is very difficult to manually define these parameters and, in most cases, an under-exposed or over-exposed (saturated) image will result.

So, we decided to fly the multispectral camera in automatic exposure mode. In general, the camera worked very well over the trees to capture well-exposed images. But the surveyed area is crossed by a forest road that is made of beige sand. Over this road, the camera chose to expose correctly the road and under expose the trees areas around (Figure 16).

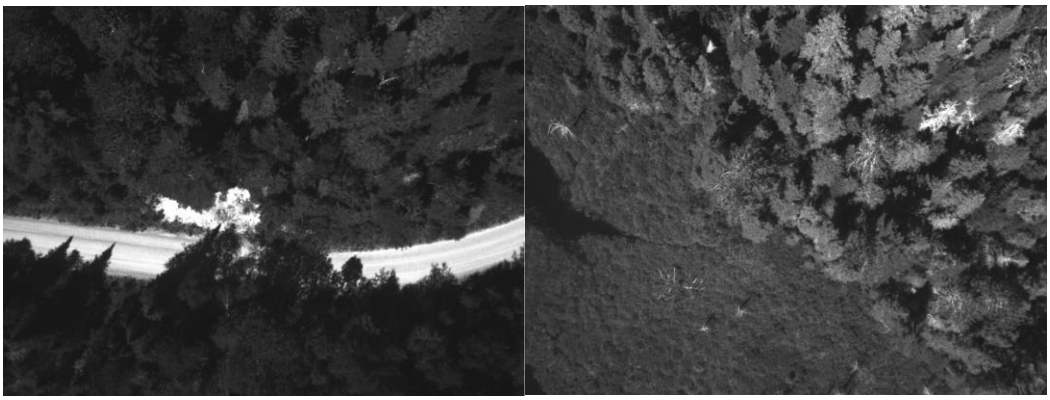

Figure 16 - Difference between an under-exposed image (left) and a well-exposed image (right)

2.2.8 Availability

As an effort of contribution to the domain, we have decided upfront to make the acquired data as well as the produced orthomosaics available. They can be downloaded by following this link: https://bit.ly/2NKXH2h[1].

2.3 Ground truth data generation

In order to produce ground truth data (i.e. manually identify tree species) related to the new drone-acquired imagery, we visited our acquisition site a second time in September with a hand-held camera and high-precision GPS/GNSS measuring equipment. We rented a Trimble Geo7X along with a Zephyr 3 antenna to acquire geolocalization of selected trees with a high accuracy (i.e. below one meter) despite a variable canopy cover. We captured photos of the bark and crown of roughly 100 trees that were also simultaneously geotagged over several hours. These trees were specifically selected in the forest for their size (or for the presence of a nearby tree cluster). The goal was to avoid the possibility that they might not show up correctly in the aerial imagery. An overview of the distribution of the tagged trees is shown in Figure 17. This data was then post-processed, cleaned, and manually analyzed to improve its accuracy before been used in tests.

[1] Full url in case of shortened link's expiry:
https://crimrd-my.sharepoint.com/:f:/g/personal/byrnsda_crim_ca/EnVQM5cEAcFClNVaHwcrj-cBlynJ44kdBY96jHXSpL6A9A?e=jLiF99

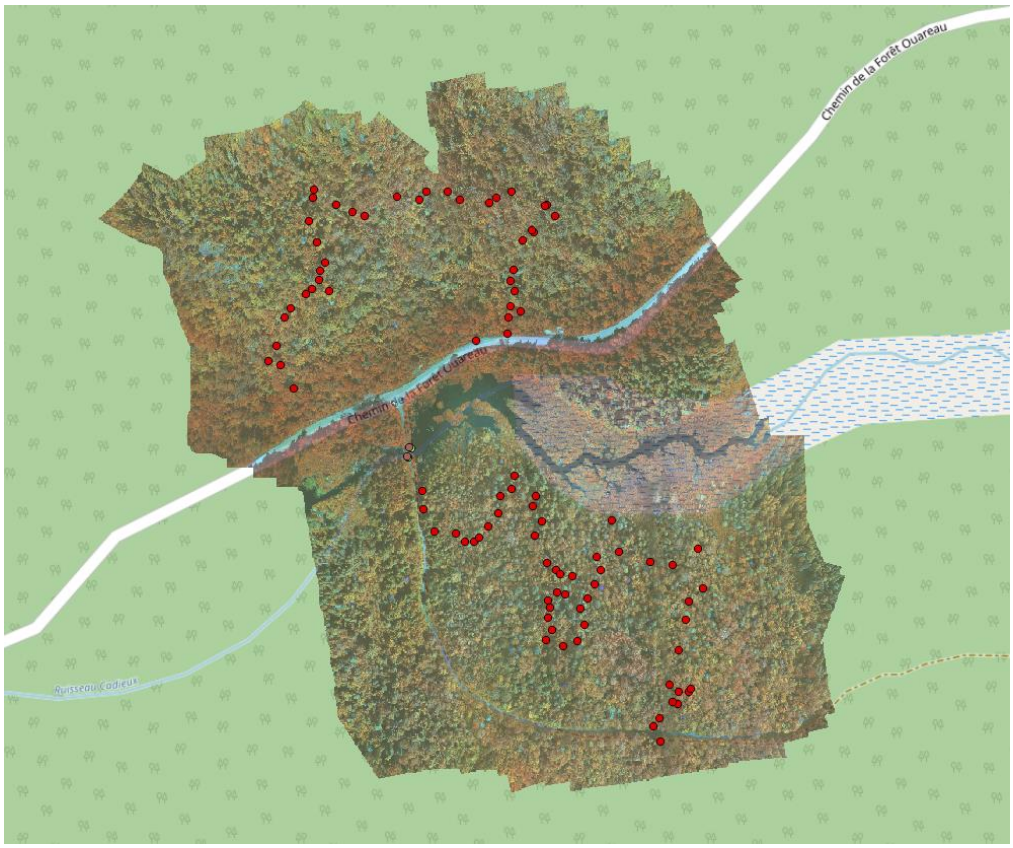

Figure 17 - Overview of the distribution of geotagged trees (red points) at the Ouareau site

For the identification of the tree species, we relied on a two-pronged approach. First, we trained a new classifier based on the bark images gathered by Carpentier et al. (2018) using our training framework presented in section 3.1. Their bark image dataset ("BarkNet") encompasses all tree species present in both the Forêt-Ouareau and Petawawa target regions, and it contains more than 23,000 samples. Using the same ResNet architecture than for the aerial imagery, also presented in section 3.1, and including new data augmentation techniques like random tiling and cropping of training images, we achieved classification accuracies of about 98% on a subset of bark images reserved for evaluation. These accuracies are similar to the accuracy reported by Carpentier et al. (2018) for a similar network architecture (97.81%). This level of performance is however not achieved when translating the problem to the images we captured ourselves in the Forêt-Ouareau region (see Figure 18 for an example). In fact, more than 20% of our images were clearly mislabeled by the bark classifier: for example, it would give beech or maple labels to trees that were conifers.

Figure 18 - Example of an image captured by Carpentier et al. (2018) for the BarkNet dataset (left), and an image captured for our test set in Forêt-Ouareau (centre). In order to isolate the main trunk in our images, we had to manually draw an approximate mask (shown on the right) for each image to control the region of interest provided to the bark classifier.

We believe these errors are caused mainly by the fact that our pictures depict the trunks at widely different scales than those in the BarkNet dataset. Furthermore, trunks at the Forêt-Ouareau site were generally not as clean as those in the BarkNet dataset, as many of them were covered in moss and/or fungi that sometimes changed the trunks' colour and texture.

To obtain a second opinion for the labeling of tree species, we also asked one of our colleagues who is more knowledgeable than us on local tree species to annotate our geotagged images. Using a set of reference samples as well as the picture of each trunk and crown, he annotated as best he could all of our geotagged samples. We intersected his predictions to the ones produced by the classifier trained on BarkNet, and manually validated all ambiguous cases by comparing the photos to more samples taken online and in other datasets. In the end, we kept 87 of the 102 originally tagged trees from our test set. The distribution of labels for these trees is shown in appendix A.

3. MODEL TRAINING BASED ON THE PETAWAWA EXPERIMENTAL FOREST

Model training is based on the imagery collected on April 21, 2009. More specifically, we first loaded the orthorectified 5x5 km tiles saved as NRGB GeoTIFF images via GDAL, and extracted individual patches for each targeted tree crown from the annotation shapefile. Some examples of extracted patches are illustrated in Figure 19. Resulting patch dimensions have been found to vary anywhere between 2 and 42 pixels (on average 10±7 pixels), making tree signatures very dynamic and irregular.

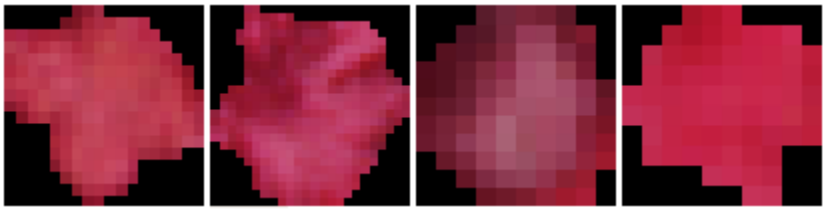

Figure 19 – Segmented tree patches extracted from NRGB GeoTIFF images

From the annotation shapefile, a total of 10918 features (polygons) could be extracted. After cleaning up out-of-bounds (323, ~2%), badly shaped (1, <1%), and uncertain or unlabeled features (1108, ~10%), a total of 9482 (87%) clean features remained for the training of a classifier. Of these, 101 were eliminated at training time due to minimal patch crop size constraints. In terms of tree species labeling, the training data possesses 25 unique classes. The distribution of these classes is presented in Table 2, which shows great imbalance between different classes.

Table 2 – Ground truth data labeling distribution for the April 21, 2009 annotations.

<unset>	'Abl'	'Alt'	'At'	'Aw'	'Ba'	'Be'	'Bw'	'By'	'Ce'	'Fb'	'He'	'Iw'
0	156	1202	1028	11	111	158	412	182	19	400	21	1
0.00%	1.66%	12.82%	10.96%	0.12%	1.18%	1.68%	4.39%	1.94%	0.20%	4.27%	0.22%	0.01%

'Lt'	'Mh'	'Mr'	'Or'	'Ow'	'Pb'	'Pj'	'Pr'	'Ps'	'Pw'	'Sb'	'Sw'	'XP'
193	439	351	698	1	71	570	1426	26	549	318	1005	41
2.06%	4.68%	3.74%	7.44%	0.01%	0.76%	6.08%	15.21%	0.28%	5.85%	3.39%	10.72%	0.44%

Supervised training from patches was accomplished using three distinct image sets. First, the “training” set defined images used to automatically modify the model’s internal weights via backpropagation so that it learnt to recognize and differentiate between various tree species based on their assigned labels. Second, a “validation” set was used to evaluate predictions resulting from the model’s adjusted weights. Since these “validation” images were never used to adjust classifier weights directly, this set helped make sure that the model generalizes to new images. This offered a verification mechanism that quickly indicated if a model’s performance was too strongly related to specific training images, which would have made it less accurate when presented with new cases. The training and validation sets were used alternately to adjust weights and re-evaluate the model’s accuracy over multiple iterations, until convergence was reached. Finally, a “testing” set containing yet again more unseen images was employed for overall performance quantification. This final set usually consisted of persisting reference images to compare performances of various model configurations.

Data splits of approximately 0.8/0.1/0.1 were employed for training, validation and testing in this project. Therefore, a total of 939 image patches are used for performance comparison of any produced models, while the remaining patches are employed for training and evaluation of different model architectures.

3.1 Neural network architecture tests

In the project time frame, we built a training framework for the exploration of various neural network model configurations and architectures. It was then used to train baseline species classifiers with raw tree patch data. We should also note that this framework is entirely open source[2] and can be used at will.

The initial model evaluated was a deep image classifier based on the ResNet-34 architecture of He et al. (2015) with an input resolution of 224x224. Due to the relatively

[2] https://github.com/plstcharles/thelper

small number of samples that compose the training dataset, this relatively shallow architecture was deemed enough for an initial baseline. Besides, because of the small size of training patches with respect to the model's input resolution, an upscaling operation was applied alongside the usual image preprocessing stages. The classic architecture was also modified to accept input images with four channels (RGB-NIR) instead of the traditional three (RGB). The results presented in Table 3 and Table 4 have been achieved with this architecture.

Table 3 - Confusion matrix of tree species classification using the baseline ResNet-34 architecture.

t/p	<unset>	Abl	Alt	At	Aw	Ba	Be	Bw	By	Ce	Fb	He	Iw	Lt	Mh	Mr	Or	Ow	Pb	Pj	Pr	Ps	Pw	Sb	Sw	XP	total
<unset>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Abl	0	3	0	0	0	1	0	1	1	0	0	0	0	0	2	4	1	0	0	0	2	0	0	0	1	0	16
Alt	0	0	71	19	0	0	0	4	3	0	0	0	0	0	0	5	14	0	0	0	4	0	0	0	0	0	120
At	0	0	17	62	0	0	0	7	1	0	1	0	0	0	0	2	7	0	0	0	1	0	5	0	0	0	103
Aw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Ba	0	0	0	0	0	4	0	2	1	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	11
Be	0	1	0	0	0	0	5	1	4	0	0	0	0	0	2	2	1	0	0	0	0	0	0	0	0	0	16
Bw	0	1	4	4	0	3	0	16	3	0	0	0	0	0	0	4	3	0	0	0	3	0	0	0	0	0	41
By	0	0	2	0	0	0	2	2	9	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	18
Ce	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	2
Fb	0	0	0	1	0	0	0	1	0	0	28	0	0	1	0	0	0	0	0	0	1	0	0	1	7	0	40
He	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2
Iw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lt	0	0	0	0	0	0	0	0	0	0	1	0	0	15	0	0	0	0	0	0	0	0	0	0	3	0	19
Mh	0	1	0	0	0	1	2	3	3	0	0	0	0	0	26	7	0	0	0	0	0	0	1	0	0	0	44
Mr	0	1	0	1	0	2	1	3	0	0	0	0	0	0	3	20	2	0	0	0	0	0	2	0	0	0	35
Or	0	1	7	6	0	0	1	2	3	0	0	0	0	0	0	6	44	0	0	0	0	0	0	0	0	0	70
Ow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pb	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	4	0	0	0	0	0	0	0	7
Pj	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	3	0	0	1	3	0	57
Pr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	139	0	2	0	0	0	143
Ps	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3
Pw	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	2	0	46	0	2	0	55
Sb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	3	1	32
Sw	0	0	0	0	0	0	0	0	0	0	3	0	0	1	0	0	0	0	0	4	3	0	0	1	88	0	100
XP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	4
total	0	9	101	95	0	11	11	43	28	0	34	0	0	19	35	53	75	0	6	56	159	1	59	33	108	3	939

Table 4 - Precision, recall and F_1-score of tree species classification using the baseline ResNet-34 architecture.

Test Set	Abl	Alt	At	Aw	Ba	Be	Bw	By	Ce	Fb	He	Lt	Mh	Mr	Or	Pb	Pj	Pr	Ps	Pw	Sb	Sw	XP	avg / total
precision	0.33	0.70	0.65	0.00	0.36	0.45	0.37	0.32	0.00	0.82	0.00	0.79	0.74	0.38	0.59	0.67	0.89	0.87	1.00	0.78	0.85	0.81	0.67	0.7049
recall	0.19	0.59	0.60	0.00	0.36	0.31	0.39	0.50	0.00	0.70	0.00	0.79	0.59	0.57	0.63	0.57	0.88	0.97	0.33	0.84	0.88	0.88	0.50	0.7039
f1-score	0.24	0.64	0.63	0.00	0.36	0.37	0.38	0.39	0.00	0.76	0.00	0.79	0.66	0.45	0.61	0.62	0.89	0.92	0.50	0.81	0.86	0.85	0.57	0.7001
support	16	120	103	1	11	16	41	18	2	40	2	19	44	35	70	7	57	143	3	55	32	100	4	939

Results from previous tables show that an average F_1-score of 0.7001 was achieved on the testing set. On the other hand, the training and validation sets generated F_1-scores of 0.9889 and 0.7116 respectively. These metrics show that the model considerably overfits the training data which consists of 7509 images. Therefore, the model was sufficiently deep to almost perfectly learn all training instances, but still had difficulty generalizing to new unseen instances of trees from the test set. This validated the assumption that a ResNet-34 architecture was enough for this baseline task. When taking a closer look at each classes' performance in Table 4 with respect to their support count, we can definitively notice that the overall system performance is hindered by classes that simply have too few samples to let the model learn variable representations of their descriptive characteristics. To avoid overfitting, the first approach attempted was to

apply stochastic transform (data augmentation) to random 90º rotations and flips. Such transformations force the neural network to search for more generic characteristics that fundamentally describe the targeted classes. Following this approach, the results of Table 5 and Table 6 were obtained.

Table 5 - Confusion matrix of true-positive tree species classification using the second (with stochastic transformations) ResNet-34 architecture.

t/p	<unset>	Abl	Alt	At	Aw	Ba	Be	Bw	By	Ce	Fb	He	Iw	Lt	Mh	Mr	Or	Ow	Pb	Pj	Pr	Ps	Pw	Sb	Sw	XP	total
<unset>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Abl	0	5	0	0	0	0	0	2	0	0	0	0	0	0	1	4	1	0	0	0	3	0	0	0	0	0	16
Alt	0	2	77	16	0	0	0	6	0	0	0	0	0	0	1	3	14	0	0	0	1	0	0	0	0	0	120
At	0	0	18	65	0	0	1	4	1	0	1	0	0	0	0	0	6	0	0	0	0	0	5	0	2	0	103
Aw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Ba	0	0	0	1	0	4	0	2	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	11
Be	0	0	0	0	0	1	6	0	6	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	16
Bw	0	1	3	2	0	0	1	22	4	0	0	0	0	0	2	4	0	0	0	0	2	0	0	0	0	0	41
By	0	1	1	0	0	2	0	4	5	0	0	0	0	0	1	3	1	0	0	0	0	0	0	0	0	0	18
Ce	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
Fb	0	0	0	1	0	0	0	1	0	0	25	0	0	1	0	0	0	0	0	0	1	0	0	4	7	0	40
He	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2
Iw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lt	0	0	0	0	0	0	0	0	0	0	2	0	0	14	0	0	0	0	0	0	0	0	0	0	2	1	19
Mh	0	1	2	0	0	1	3	2	3	0	0	0	0	0	28	3	0	0	0	0	0	0	1	0	0	0	44
Mr	0	2	4	1	0	0	0	1	2	0	1	0	0	0	5	16	2	0	0	0	0	0	1	0	0	0	35
Or	0	2	7	9	0	0	2	3	4	0	0	0	0	0	0	2	40	0	0	0	1	0	0	0	0	0	70
Ow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pb	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	4	0	0	0	0	0	0	0	7
Pj	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	1	0	0	1	2	0	57
Pr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	138	0	2	0	0	0	143
Ps	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	3
Pw	0	0	0	2	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	2	0	45	0	2	0	55
Sb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	3	1	32
Sw	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	3	4	0	0	3	86	1	100
XP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4
total	0	15	112	97	0	8	13	49	26	1	32	0	0	18	41	38	65	0	5	59	153	2	57	36	105	7	939

Table 6 - Precision, recall and F_1-score of tree species classification using the second (with stochastic transformations) ResNet-34 architecture.

Test Set	Abl	Alt	At	Aw	Ba	Be	Bw	By	Ce	Fb	He	Lt	Mh	Mr	Or	Pb	Pj	Pr	Ps	Pw	Sb	Sw	XP	avg / total
precision	0.33	0.69	0.67	0.00	0.50	0.46	0.45	0.19	1.00	0.78	0.00	0.78	0.68	0.42	0.62	0.80	0.90	0.90	1.00	0.79	0.78	0.82	0.57	0.7122
recall	0.31	0.64	0.63	0.00	0.36	0.38	0.54	0.28	0.50	0.63	0.00	0.74	0.64	0.46	0.57	0.57	0.93	0.97	0.67	0.82	0.88	0.86	1.00	0.7114
f1-score	0.32	0.66	0.65	0.00	0.42	0.41	0.49	0.23	0.67	0.69	0.00	0.76	0.66	0.44	0.59	0.67	0.91	0.93	0.80	0.80	0.82	0.84	0.73	0.7096
support	16	120	103	1	11	16	41	18	2	40	2	19	44	35	70	7	57	143	3	55	32	100	4	939

Again, with the second model, overfitting of the training set occurred. An F_1-score of 0.7096 (an absolute increase of 0.95% over the previous model) was reached on the testing set, while values of 0.9967 and 0.7200 were obtained on the training and validation sets.

Because performances slightly increased compared to the previous test, but overfitting remained, further increasing the amount of transformations was considered. Instead of only two transforms, six were applied. Amongst these transforms, random zooms were also applied to mitigate the highly variable dimensions of extracted patches previously mentioned. Results presented in Table 7 and Table 8 were produced using this methodology.

Table 7 – Confusion matrix of true-positive tree species classification using the third (with 6 stochastic transformations) ResNet-34 architecture.

t/p	<unset>	Abl	Alt	At	Aw	Ba	Be	Bw	By	Ce	Fb	He	Iw	Lt	Mh	Mr	Or	Ow	Pb	Pj	Pr	Ps	Pw	Sb	Sw	XP	total
<unset>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Abl	0	2	0	1	0	0	0	2	0	0	0	0	0	0	3	4	1	0	0	0	3	0	0	0	0	0	16
Alt	0	0	67	28	0	1	1	6	3	0	0	0	0	0	0	1	9	0	0	0	4	0	0	0	0	0	120
At	0	0	20	63	0	0	0	4	0	0	0	0	0	0	0	1	6	0	1	0	3	0	4	0	1	0	103
Aw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Ba	0	0	0	2	0	3	1	2	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	11
Be	0	1	0	0	0	1	6	2	3	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	16
Bw	0	0	3	3	0	3	0	24	2	0	0	0	0	0	0	3	2	0	0	0	1	0	0	0	0	0	41
By	0	0	2	0	0	1	1	2	7	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	0	0	18
Ce	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	2
Fb	0	0	0	0	0	0	0	1	0	0	26	0	0	2	0	0	0	0	0	0	1	0	0	1	9	0	40
He	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2
Iw	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lt	0	0	0	1	0	0	0	0	0	1	1	0	0	12	0	0	0	0	0	0	0	0	1	0	0	2	19
Mh	0	0	0	1	0	0	2	4	4	0	0	0	0	0	27	4	1	0	0	0	0	0	1	0	0	0	44
Mr	0	0	2	1	0	0	0	4	1	0	2	0	0	0	3	16	4	0	0	0	0	0	2	0	0	0	35
Or	0	2	5	11	0	0	0	2	3	0	0	0	0	0	0	4	42	0	0	0	1	0	0	0	0	0	70
Ow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pb	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	4	0	0	0	0	0	0	0	7
Pj	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	0	0	0	0	1	2	57
Pr	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	138	0	1	0	0	0	143
Ps	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	3
Pw	0	0	0	6	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	3	0	38	1	3	0	55
Sb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	3	1	32
Sw	0	0	0	1	0	0	0	0	0	1	5	0	0	1	0	0	0	0	0	5	2	0	0	0	85	0	100
XP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4
total	0	6	99	119	0	9	12	53	24	2	34	0	0	18	36	39	69	0	7	63	156	1	48	31	107	6	939

Table 8 – Precision, recall and F_1-score of tree species classification using the third (with 6 stochastic transformations) ResNet-34 architecture.

Test Set	Abl	Alt	At	Aw	Ba	Be	Bw	By	Ce	Fb	He	Lt	Mh	Mr	Or	Pb	Pj	Pr	Ps	Pw	Sb	Sw	XP	avg / total
precision	0.33	0.68	0.53	0.00	0.33	0.50	0.45	0.29	0.00	0.76	0.00	0.67	0.75	0.41	0.61	0.57	0.86	0.88	0.00	0.79	0.90	0.79	0.67	0.6855
recall	0.13	0.56	0.61	0.00	0.27	0.38	0.59	0.39	0.00	0.65	0.00	0.63	0.61	0.46	0.60	0.57	0.95	0.97	0.00	0.69	0.88	0.85	1.00	0.6880
f1-score	0.18	0.61	0.57	0.00	0.30	0.43	0.51	0.33	0.00	0.70	0.00	0.65	0.68	0.43	0.60	0.57	0.90	0.92	0.00	0.74	0.89	0.82	0.80	0.6828
support	16	120	103	1	11	16	41	18	2	40	2	19	44	35	70	7	57	143	3	55	32	100	4	939

In this case, F_1-scores of 0.9918, 0.7152 and 0.6828 were attained correspondingly for training, validation and test sets. Yet again, overfitting remains a concern, and in this specific case, additional transforms did not seem to further improve the performance. Specifically, it decreased by 1.73% compared to the baseline configuration.

Early conclusions of these tests are that the selected architecture suffices to learn very specific patterns for tree species recognition, but the limited amount of data makes generalization without overfitting quite difficult. Future considerations and testing should therefore revolve around (1) improving data balancing and variety using additional data augmentation techniques, and (2) removing or combining the predictions of data classes with limited support.

All the previous model architectures were implemented using the PyTorch library. Preprocessing steps for synthetic image generation were accomplished using a combination of OpenCV, PIL (pillow) and NumPy libraries. Extraction operations to retrieve tree segmentations from shapefiles were executed using GDAL. Every model was trained over 40 epochs. Including the overall data preprocessing, training epochs and post-training evaluation and tests, each model was trained in approximately 2 hours using a single Nvidia GTX1080Ti GPU.

4. FORÊT-OUAREAU TREE SPECIES CLASSIFICATION EXPERIMENTS

In this section, we resume the experiments conducted in relation to the classification of tree species on our own acquired dataset, and present our most up-to-date results.

4.1 Radiometry correction

The NIR-RGB orthomosaic obtained following our acquisitions at the Forêt-Ouareau site (shown in Figure 12) is actually quite different from the Petawawa orthomosaic. The difference in radiometry might be related to the exposure settings used on the sensors, or based on how PhotoScan normalizes and blends the acquired images together to build the orthomosaic. In any case, a colour correction is required for the Ouareau test data to become compatible with the Petawawa training/validation data. We accomplished this by rectifying the cumulative histogram of each band in the Ouareau orthomosaic to match the corresponding bands in the Petawawa orthomosaic. The cumulative histograms before and after rectification are shown in Figure 20. A comparison of canopy patches taken from both datasets is shown in Figure 21. We can observe in this latter figure that the patches from both datasets are now quite similar.

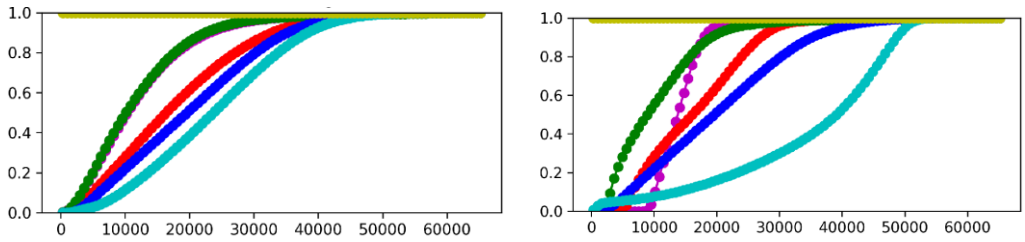

Figure 20 - Cumulative 16-bit band histograms of the Ouareau orthomosaic before applying colour correction (left) and after applying colour correction (right).

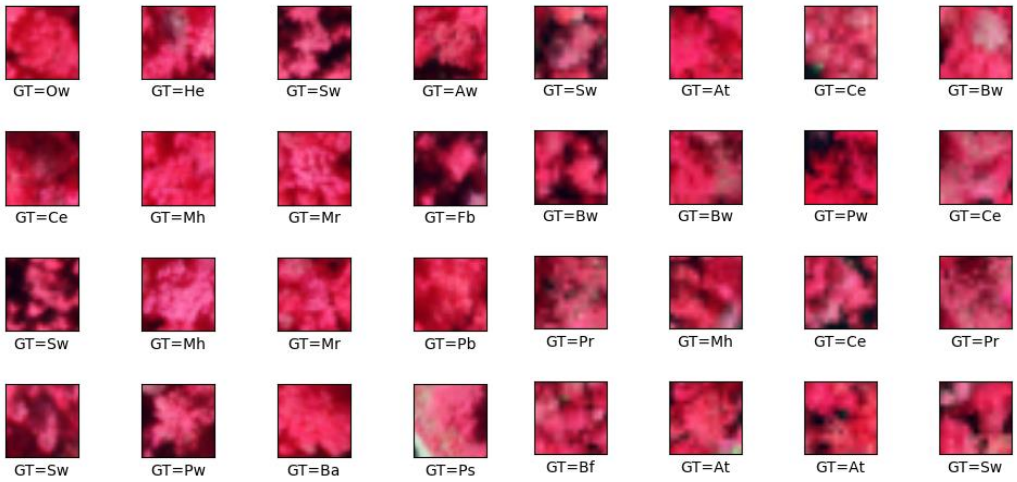

Figure 21 - Examples of random tree patches taken from the Petawawa orthomosaic (left four columns) and the Ouareau orthomosaic (right four columns). Each sample has its ground truth label underneath it.

4.2 Input data variations

Based on the new dataset, we changed our crown patch sampling strategy. At first, we extracted image patches using the axis-aligned bounding box of the targeted crown geometries found in the annotation shapefile. The resulting patches were then normalized to our network's input resolution and processed following some geometric augmentation operations. Examples of these patches are shown in Figure 22. Our initial hypothesis was that the information surrounding the crown was not important for its classification, and that it might even mislead the classifier. However, subsequent experiments demonstrated that this contextual information was actually quite beneficial. In fact, using fixed-size patches of 5 m by 5 m centred on the centroid of each crown geometry outperformed the perfectly extracted crown images by over 3% in terms of mean classification accuracy. Such patches are shown in Figure 22. Increasing this fixed size to 10 m by 10 m further increased mean accuracy by another 2%, despite the fact that the very large majority of crowns in the Petawawa dataset cover a much smaller surface. This seems to indicate either that crown segments are often badly aligned to their respective trees, or that surrounding trees can be used to identify clusters more easily.

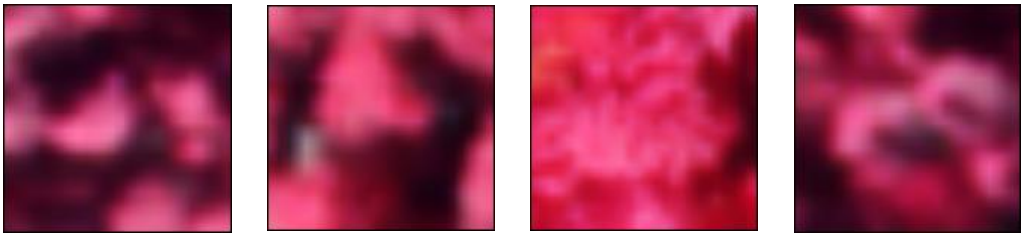

Figure 22 - Examples of fixed-size (10 m x 10 m) crown patches extracted from the Petawawa NRGB dataset.

As for the use of data augmentation operations, we observed that any form of radiometric transformations adversely affected the classification performance, while simple geometric transformations were mostly beneficial. The final set of augmentation operations we rely on thus consists of horizontal and vertical flips, lossless rotations, and random zooming and cropping around the target region of interest.

4.3 Data limitations

We had initially planned to study crown detection and/or instance segmentation pipelines alongside patch-based classification approaches. However, it became clear through our explorations of different solutions that these tasks required a form of annotation that differed from the one at our disposal. In other words, the sparsely annotated tree crowns contained in the Petawawa shapefile would not be adequate for the training of supervised detection or instance segmentation models. This is due to the fact that these models generate dense predictions on large-scale input images; any unlabelled instance in the input images might therefore be detected by the model, and the model would be punished for detecting them during the training phase. Due to this problem, we had to abandon the use of detection and instance segmentation models.

Interestingly, we could still adapt our current classification models with the approach introduced by Long et al. (2015) to generate semantic segmentation maps from images

of arbitrary resolutions. We experimented with this approach in the final phase of the project; the results are discussed in section 6.

4.4 Classification model architecture variations

As discussed in section 4.2, our classification models now process fixed-scale patches with a spatial coverage often much larger than the targeted crowns. We discovered that using contextual information helps, but forcing the algorithm to consider the structure of the patches would be ideal, as the middle of a patch is more likely to contain the crown of interest. Convolutional neural networks are translation-invariant by nature, a property that is very useful in unconstrained settings. However, this property can be substituted for better structural awareness in our application. To do so, we added three new channels to the NIR-RGB patches given as input to the model. We encoded the X and Y coordinates of each pixel in the patch in the first two channels, and the Euclidean distance to the middle of the patch (which corresponds to the middle of the crown) for each pixel in the third channel. These new channels could then be exploited by the model to learn visual features that were specific to some locations in the analyzed patches. As shown previously, a relatively shallow architecture such as the 34-layer ResNet can already overfit our training data due to its high capacity. The specialization of features to be location-aware should therefore not overburden this architecture. In fact, the 34-layer ResNet model used to obtain our latest results in Section 4.5 still overfited our training data with these extra channels, but yielded better validation and test results than its equivalent configuration without the new channels.

We also briefly studied whether using deeper architectures resulted in better overall performance despite even more risk of overfitting to the training data. Different configurations of ResNets (using 50 to 152 layers) as suggested by He et al. (2015) were tested alongside the Inception-ResNet v2 architecture of Szegedy et al. (2016) and the DenseNet architecture of Huang et al. (2016). In the end, none of these surpassed the performance of our baseline ResNet-34 configuration, and most of them even resulted in significant drops in performance. This seems to indicate that even with our current data augmentation operations, the number of annotations we possess is insufficient to train such high-capacity models.

Finally, we trained various models based on the ResNet-34 architecture using different layer block types, namely the basic and bottleneck blocks suggested by He et al. (2015), and the squeeze-excitation block proposed by Hu et al. (2017). While the first two seemed to result in roughly the same validation and test performance, the squeeze-excitation block significantly improved performance at test time (by over 5% mean accuracy).

4.5 Final results

We present our latest validation and test results for tree species classification in separate tables. First, Table 9 shows the best classification performance achieved on the Petawawa validation set (i.e. on the 10% portion of all annotations that were withheld from training) when all 23 classes are considered. Compared to the previous results, these new ones show an increase of more than 0.15 in absolute F_1-score (an improvement of 22% in relative terms). Besides, the overall accuracy of our best model on the validation set is currently 86.47%. This demonstrates the capacity of our model to

properly identify tree species using subtle cues that are beyond human grasp, as trying to visually identify tree species ourselves from the input patches is almost impossible.

Table 9 - Classification results for the Petawawa validation set.

Species	Precision	Recall	F_1-score	Sample Count
Abl	0.7143	0.6250	0.6667	16
Alt	0.9153	0.9000	0.9076	120
At	0.9048	0.9223	0.9135	103
Aw	0.5000	1.0000	0.6667	1
Ba	0.5385	0.6364	0.5833	11
Be	0.6471	0.6875	0.6667	16
Bf	n/a	n/a	n/a	0
Bw	0.6250	0.6098	0.6173	41
By	0.5833	0.3889	0.4667	18
Ce	0.5000	0.5000	0.5000	2
Fb	0.8974	0.8750	0.8861	40
He	1.0000	1.0000	1.0000	2
Lt	0.9444	0.8947	0.9189	19
Mh	0.7021	0.7500	0.7253	44
Mr	0.6857	0.6857	0.6857	35
Or	0.8056	0.8286	0.8169	70
Pb	0.7143	0.7143	0.7143	7
Pj	0.9643	0.9474	0.9558	57
Pr	0.9650	0.9650	0.9650	143
Ps	1.0000	1.0000	1.0000	3
Pw	0.9184	0.8182	0.8654	55
Sb	0.9412	1.0000	0.9697	32
Sw	0.9065	0.9700	0.9372	100
XP	1.0000	1.0000	1.0000	4
Overall	0.8645	0.8647	0.8636	939

Next, we show in Table 10 the best classification performance achieved by the same model on the Ouareau test set. Here, we see a drastic drop in overall performance, as the mean F_1-score is down to only 0.1561. This indicates that despite our best efforts at colour correction to make the patches of both datasets visually similar (c.f. Figure 21), there is still at least one variation factor that is unaccounted for. Possibilities include a lack of training samples in the Petawawa dataset for tree species that are more common in the Ouareau test set (e.g. *Thuja Occidentalis*), and the detrimental modification of subtle visual signatures via our colour correction approach.

Table 10 - Classification results for the Ouareau validation set.

Species	Precision	Recall	F_1-score	Sample Count
Abl	n/a	n/a	n/a	0
Alt	n/a	n/a	n/a	0
At	0.0000	0.0000	0.0000	18
Aw	n/a	n/a	n/a	0
Ba	n/a	n/a	n/a	0
Be	0.0000	0.0000	0.0000	8
Bf	n/a	n/a	n/a	4
Bw	0.5000	0.2857	0.3636	14
By	n/a	n/a	n/a	0
Ce	0.0000	0.0000	0.0000	8
Fb	n/a	n/a	n/a	0
He	n/a	n/a	n/a	0
Lt	n/a	n/a	n/a	0
Mh	1.0000	0.1250	0.2222	8
Mr	0.0000	0.0000	0.0000	1
Or	n/a	n/a	n/a	0
Pb	n/a	n/a	n/a	0
Pj	n/a	n/a	n/a	0
Pr	0.1739	0.5714	0.2667	7
Ps	n/a	n/a	n/a	0
Pw	0.5000	0.1818	0.2667	11
Sb	0.1667	1.0000	0.2857	1
Sw	0.3333	0.0769	0.1250	13
XP	n/a	n/a	n/a	0
Overall	0.3013	0.1494	0.1561	87

Finally, note that with the input data and structural improvements detailed earlier, training a model from randomly initialized weights (“from scratch”) using the developed framework takes approximately 5 hours on a Nvidia GTX 1080 Ti. Predicting the label codes for a new set of patches requires only a few milliseconds per patch.

5. CLOUD INFRASTRUCTURE

One of the achievements of the project was to deploy a cloud infrastructure offering some machine learning capabilities and results visualisation. Using a virtual machine host on our OpenStack cloud environment, we have been able to bring these services online at the following URL: https://ccfb.crim.ca.

The next sections focus on the architecture and details of the processing and visualization services.

5.1 Platform architecture

The architecture contains two services. The first one is the project API, which handles a project database, images, point clouds, annotations and models. The second one distributes job processing via the WPS (Web Processing Service) standard. The WPS wraps tasks such as image preprocessing or forest species classification. The last component, GeoServer, handles the visualization of the classification results via the WMS (Web Map Service) and WFS (Web Feature Service) standards. Figure 23 summarizes the updated cloud services architecture.

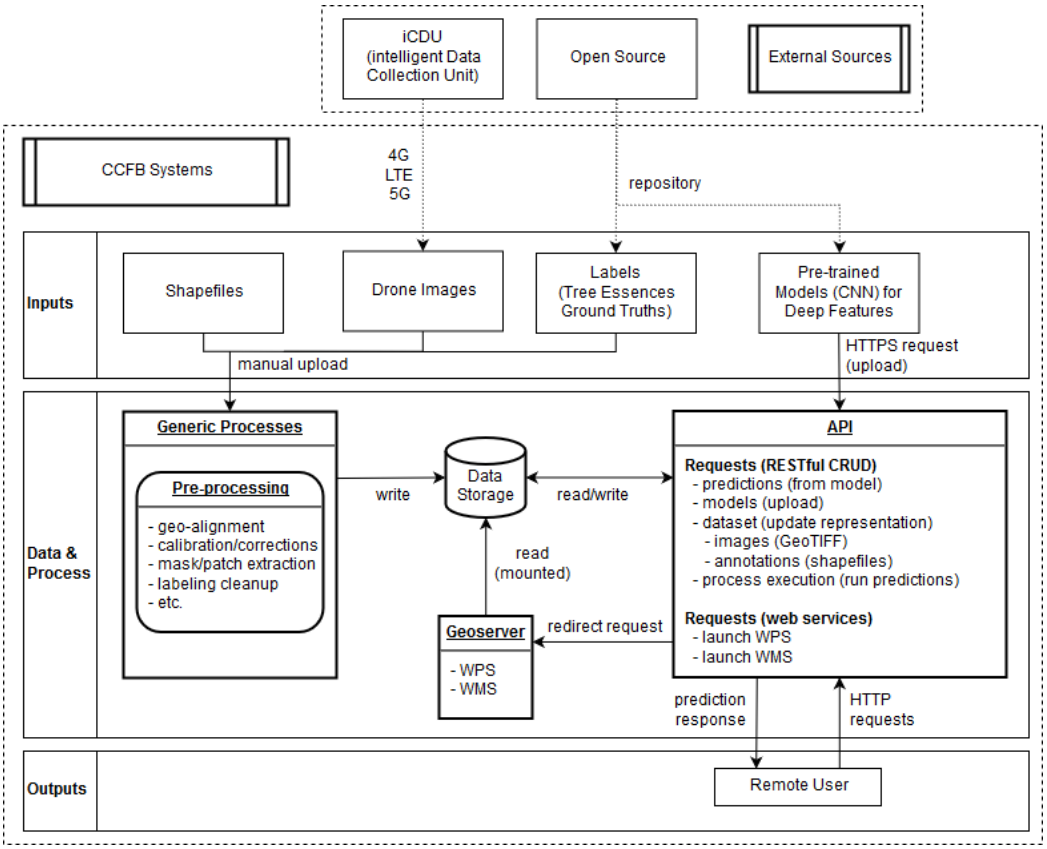

Figure 23 - Platform architecture

For convivial and robust deployment of new cloud infrastructure versions and functionalities, a *docker-compose* approach was employed. This method allows to define the multiple services required by the cloud server as well as helper services such as a database in a configuration file. The configuration also allows to set parameters and connection between servers. Finally, docker-compose allows to instanciate and configure all services in a single operation. Since each building block of the cloud infrastructure is delivered as independent Docker images, the versioning and status of very specific parts of the overall infrastructure can be controlled, which ensures more reliable deployments.

The database and GeoServer sections of the cloud infrastructure provide standardized storage implementation of the uploaded data. They are mostly employed out-of-the-box from existing images. The data processing, model prediction and REST-API offered to access them are custom implementations made for this application. The following sections will describe in more details the functionalities related to this project.

5.2 Processing

On the processing side, the implemented service handles datasets, models, processes and jobs. Functionalities include listing them, posting a new dataset or model definition to the cloud service and getting back their detailed information. The service also includes the processes entry point allowing us to describe available processes and execute them. A process allows to evaluate a dataset definition using an uploaded and pre-trained model from another dataset. Note that the model is trained prior to being uploaded on this cloud infrastructure.

The following table shows the most important routes of the service basic architecture currently deployed. More details are available by accessing the API page[3].

Table 11 – Service REST-API requests summary

HTTP Method	Request	Description
GET	`/api`	User interface listing all available routes hosted by the cloud service.
GET	`/datasets`	Get a list of available dataset unique identifiers from the service.
POST	`/datasets`	Upload a new dataset to server.

	Payload example:
	`{`
	`  "ccfb02_2018_rgbnir": {`
	`  "type": "ccfb02.ouareau.GeotagTreeDataset",`
	`  "params": {`
	`  "rasterfile": "/data/ouareau/foret_ouareau_multispectral.corrected.epsg26918.tif",`
	`  "shapefile": "/data/ouareau/raw_20180916/annotations_epsg26918/*.shp",`
	`  "crop_real_size": 10.0,`
	`  "scaling_factor": 0.08,`
	`  "category__INACTIVE": "something",`
	`  "category_field_name": "label_fred"`
	`  }`
	`  }`
	`}`	

GET	`/dataset/{dataset_uuid}`	Retrieve a dataset description using its unique identifier.
GET	`/dataset/{dataset_uuid}/download`	Download a dataset using its unique identifier.
GET	`/models`	Get a list of available models from the service.

[3] https://ccfb.crim.ca/api

<table>
<tr><td rowspan="2">POST</td><td>/models</td><td>Upload a new model definition to the server (server file path or remote URL).</td></tr>
<tr><td colspan="2">Payload example:
{
"model": "<model-checkpoint-file-path-or-url>",
}</td></tr>
<tr><td>GET</td><td>/model/{model_uuid}</td><td>Retrieve a model description using its unique identifier.</td></tr>
<tr><td>GET</td><td>/model/{model_uuid}/download</td><td>Download a model using its unique identifier.</td></tr>
<tr><td>GET</td><td>/processes</td><td>Get a list of available processes that can be run by the service.</td></tr>
<tr><td>GET</td><td>/processes/{process_uuid}</td><td>Describe a specific process using its unique identifier.</td></tr>
<tr><td rowspan="2">POST</td><td>/processes/{process_uuid}/jobs</td><td>Launch a new job for the specified process, such as a model testing.</td></tr>
<tr><td colspan="2">Payload example:
{
"inputs": [
{
"id": "model",
"value": "<model_uuid>",
},
{
"id": "dataset",
"value": "<dataset_uuid>",
},
{
"id": "metrics",
"value": [
{
"name": "class",
"type": "thelper.optim.ClassifLogger",
},
{
"name": "params",
"type": "top_k",
"value": 3
}
]
}
],
"mode": "async"
}

Response example (data only):
{
"jobID": "5de8e361-3ef1-43f4-9355-ecea7f4d3b51",
"status": "accepted",
"location":
"<server_url>/processes/<process_uuid>/jobs/<job_uuid>"
}</td></tr>
<tr><td>GET</td><td>/processes/{process_uuid}/jobs/<job_uuid></td><td>Retrieve the execution status of the submitted job.</td></tr>
<tr><td rowspan="2">GET</td><td>/processes/{process_uuid}/jobs/<job_uuid>/result</td><td>Retrieve the execution result of the submitted job when “succeeded”.</td></tr>
<tr><td colspan="2">Response example (data only):
"data": {
"outputs": [</td></tr>
</table>

```
      {
        "id": "classes",
        "value": [
          "Pj",
          <other class labels...>
          "Bf"
        ]
      },
      {
        "id": "predictions",
        "value": [
          {
            "predictions": [
              -10.086864471435547,
              <other class predictions for sample...>
              -13.59804916381836
            ],
            "gt": 4,
            "geo_tag": [
              580790.664009391,
              5122995.64044838,
              318.816854667974
            ],
            "geo_bbox": null,
            "id": "ouareau47"
          },
          <other processed samples...>
      },
  ],
}
```


5.2.1 Framework functionalities of the cloud service

Out of the complete offline framework for model training, evaluation and testing, only the testing procedure is ported to the cloud service. In other words, a model is expected to be trained offline using a powerful cluster and vast datasets of images to generate a robust model's checkpoint file. The trained model can then be uploaded to the cloud server (using `POST /models`), where it can be used for classification on another test dataset (using `POST /process/{process_uuid}/jobs`). Using the various test datasets definitions uploaded (using `POST /datasets`), the available models and versatile job execution inputs, many testing variations are possible to report performances of different training configurations on different testing conditions.

Images used in datasets are not expected to be transferred via the REST-API due to size limitations and transfer time that would be required otherwise. Instead, images are manually transferred into the server's storage by an administrator. These images are afterward available for use in any desired combination using the "dataset" object definitions uploaded to the server. This allows to reuse existing images according to multiple "sets" depending on the desired evaluation to be executed.

Because of the highly adaptable "process" definitions, jobs available for execution can define any desired operation. The most common example is when a process defines the dataset and model references (i.e. UUIDs), so that classification and prediction of tree species on corresponding images can be accomplished. The process execution methodology will entirely depend on the process definition, which will ensure monitoring and logging of the job's execution until successful completion.

Depending on the dataset's definition, the amount of samples to be processed by a given model can change drastically, and so does the job execution time. The major limitation in this regard is the available worker resources to process the data. Given a large amount of samples to be processed, the current cloud infrastructure will segment the images over multiple parallel worker threads which will all be monitored by a job executor. Sample predictions are gradually updated as the images' batch processing is completed. Multiple jobs can also be submitted for processing. The cloud infrastructure will automatically monitor various jobs simultaneously up to a certain threshold. Over this value, jobs will be put into "accepted" status until previous jobs in the queue are completed. This threshold is also limited by the server's specific resources. Monitoring of any given job's execution and available results can be accomplished using the provided routes from Table 11.

5.3 Visualization

The second goal of the cloud infrastructure is to visualize the classification result once that process is completed. To complete this task, we use GeoServer. It is a JAVA free and open-source server for sharing geospatial data, implemented, tested, and supported as a community-driven project by individuals and organizations around the world. Using open standards set forth by the Open Geospatial Consortium (OGC), GeoServer allows for great flexibility in map creation and data sharing. We provide links to visualize the results in any applications that can manipulate WMS (Web Map Service) transactions. Any map servers that meet OGG WMS standards provide an http request to parse the catalog of available layers to view.

Our GeoServer WMS endpoint can be accessed using the following link: https://ccfb.crim.ca/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities.

In our case, the QGIS (2.18) application is used to parse the catalog for the visualization, and three outputs from the Petawawa region are shown in the next figures.

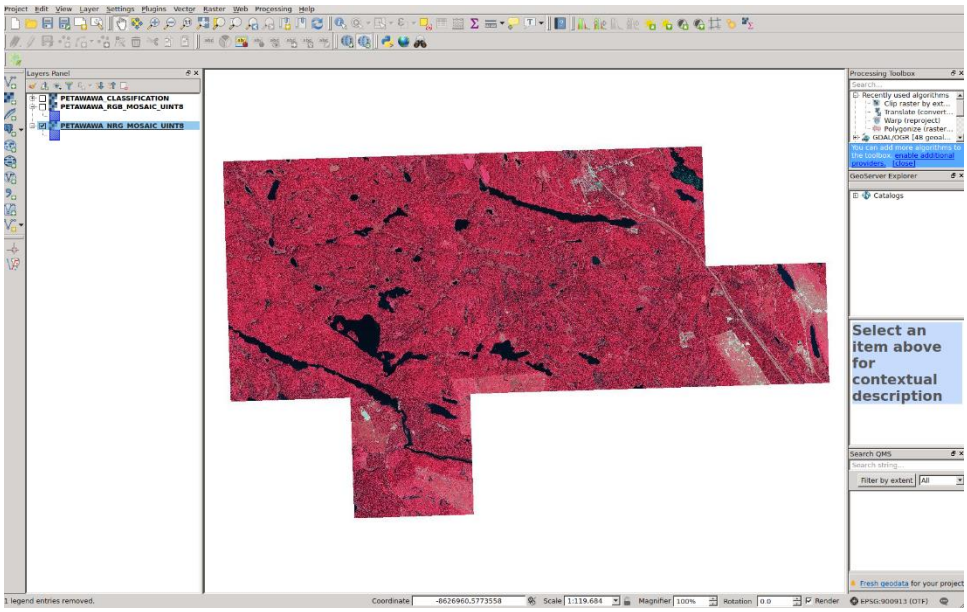

Figure 24 - The Petawawa NRG mosaic

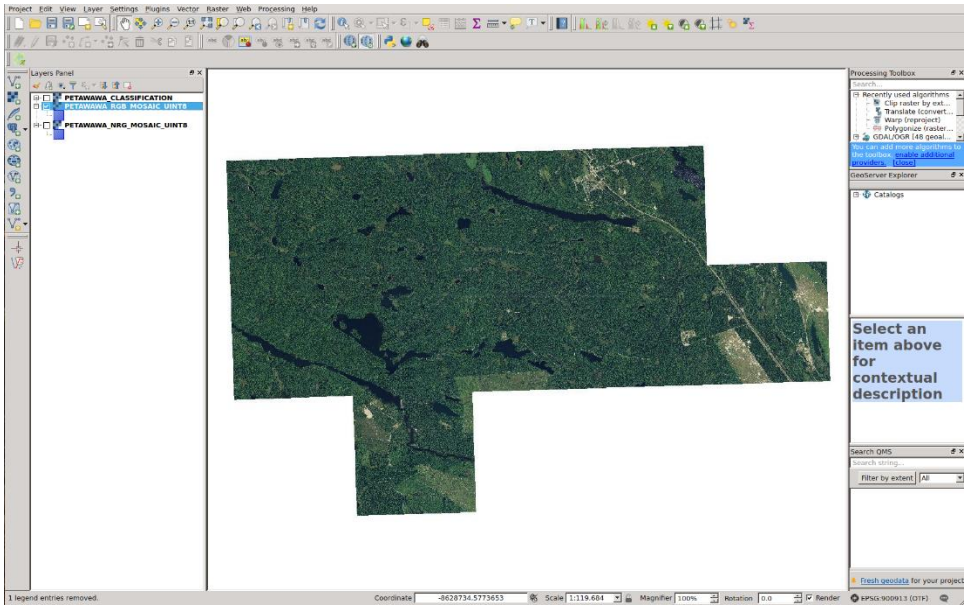

Figure 25 - The Petawawa RGB mosaic

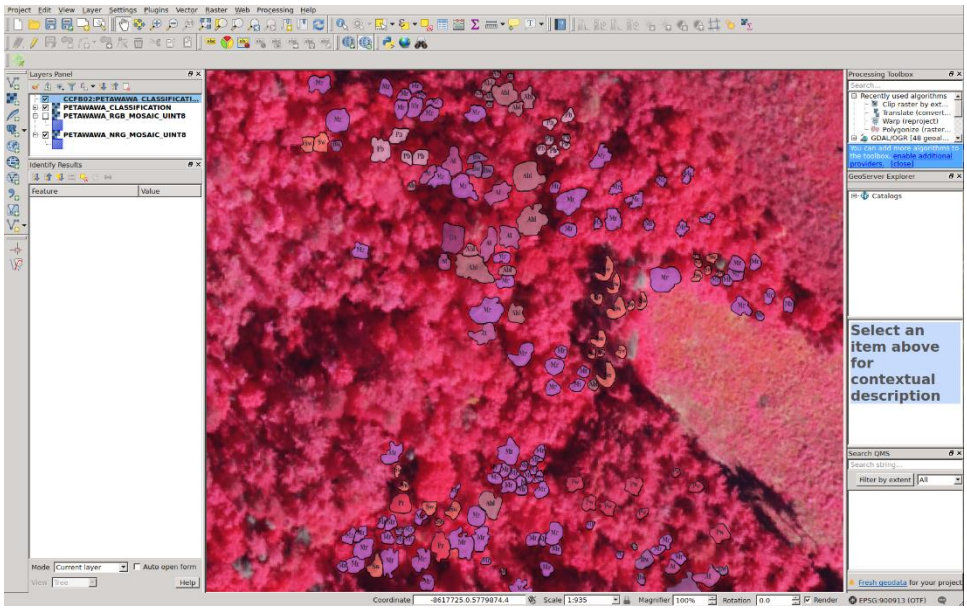

Figure 26 - The classification output

Geoserver provides access to the polygons directly with the WFS (Web Feature Service) transaction interface. This allows a user to do some analysis on the results with SQL requests. For example, the next figure shows misclassified polygons in yellow.

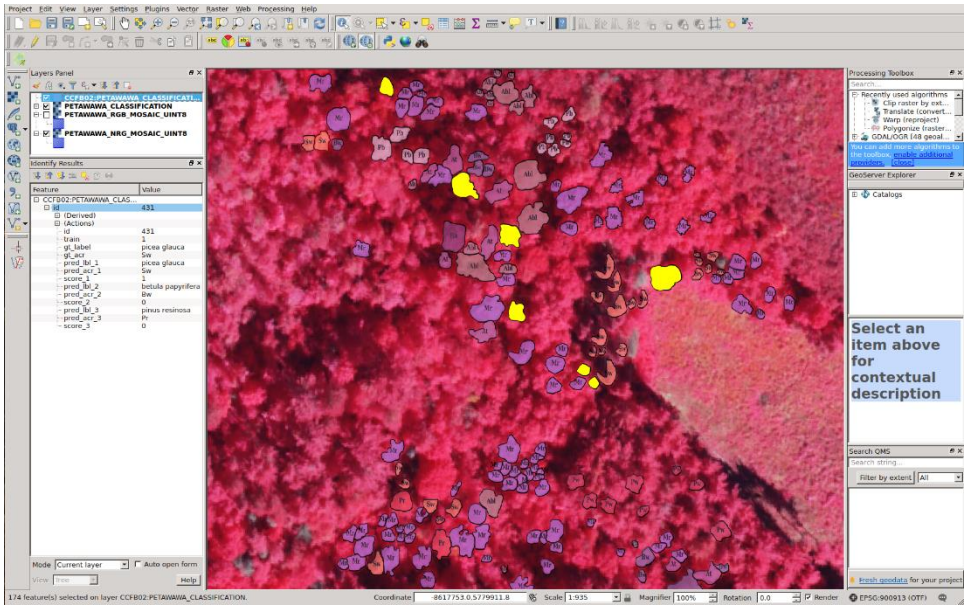

Figure 27 - Misclassified polygons in yellow

6. TESTS

The test activity was split in two sub-activities, one served to validate the cloud infrastructure and the other to make additional experiments trying to improve the training results.

6.1 Cloud service requests

This section provides the complete and detailed step-by-step guide specifying how the cloud infrastructure was employed to obtain prediction results from the “Ouareau” dataset from a model pre-trained on the “Petawawa” dataset and tree species.

1. Upload a pre-trained model checkpoint file.

REQUEST:
POST https://ccfb.crim.ca/models

Headers :
```
{"Content-Type": "application/json", "Accept": "application/json"}
```

Body:
```
{
    "model_name": "test_model",
    "model_path": "/data/outputs/ccfb02-petawawa-resnet34-se-10m-
allclasses.ckpt.best.pth"
}
```

RESPONSE :
Body:
```
{
    "meta": {
        "code": 201,
        "type": "application/json",
        "detail": "Create model successful.",
        "route": "/models",
        "uri": "http://ccfb.crim.ca/models",
        "method": "POST"
    },
    "data": {
        "model": {
            "uuid": "35440e4d-a02d-4e5c-924d-62d03e0d5199",
            "name": "test_model",
            "path": "/data/outputs/ccfb02-petawawa-resnet34-se-10m-
allclasses.ckpt.best.pth",
            "created": "2019-01-17T23:57:18.841348+00:00"
        }
    }
}
```

2. Define a dataset representation of available data on the server.

REQUEST:
POST https://ccfb.crim.ca/datasets

Headers :
```
{"Content-Type": "application/json", "Accept": "application/json"}
```

Body:
```
{
```

```
    "dataset_name": "ouareau",
    "dataset_path": "/data/ccfb/datasets/ouareau",
    "dataset_type": "ccfb.ml.ouareau.GeotagTreeDataset",
    "dataset_params": {
        "rasterfile":
"/data/ccfb/datasets/ouareau/foret_ouareau_multispectral.corrected.epsg2
6918.tif",
        "shapefile":
"/data/ccfb/datasets/ouareau/raw_20180916/annotations_epsg26918/*.shp",
        "crop_real_size": 10,
        "scaling_factor": 0.08,
        "category_field_name": "label_fred"
    }
}
```


RESPONSE :
Body:


```
{
    "meta": {
        "code": 201,
        "type": "application/json",
        "detail": "Create dataset successful.",
        "route": "/datasets",
        "uri": "http://ccfb.crim.ca/datasets",
        "method": "POST"
    },
    "data": {
        "dataset": {
            "uuid": "8d76da9c-39f9-46a0-9342-af91c76724f5",
            "name": "ouareau",
            "path": "/data/ccfb/datasets/ouareau",
            "type": "ccfb02.ouareau.GeotagTreeDataset",
            "params": {
                "rasterfile":
"/data/ccfb/datasets/ouareau/foret_ouareau_multispectral.corrected.epsg2
6918.tif",
                "shapefile":
"/data/ccfb/datasets/ouareau/raw_20180916/annotations_epsg26918/*.shp",
                "crop_real_size": 10,
                "scaling_factor": 0.08,
                "category_field_name": "label_fred"
            }
        }
    }
}
```


3. Create a process (as required) linking the model and datasets of previous steps.

REQUEST:
POST https://ccfb.crim.ca/processes

Headers :


```
{"Content-Type": "application/json", "Accept": "application/json"}
```


Body:


```
{
    "process_name": "test_model_dataset",
    "process_type": "ml"
}
```


RESPONSE :
Body:


```
{
    "meta": {
        "code": 201,
        "type": "application/json",
```

```
        "detail": "Create process successful.",
        "route": "/processes",
        "uri": "http://ccfb.crim.ca/processes",
        "method": "POST"
    },
    "data": {
        "process": {
            "uuid": "d8cfbd04-8c33-42f9-b169-7f959c10454e",
            "identifier": "test_model_dataset",
            "title": "test_model_dataset",
            "abstract": "",
            "keywords": [],
            "metadata": [],
            "version": null,
            "inputs": null,
            "outputs": null,
            "jobControlOptions": null,
            "outputTransmission": null,
            "executeEndpoint": "https://ccfb.crim.ca/processes/d8cfbd04-
8c33-42f9-b169-7f959c10454e/jobs"
        }
    }
}
```


4. Execute the created process and monitor the prediction job's status.

REQUEST:
POST
https://ccfb.crim.ca/processes/d8cfbd04-8c33-42f9-b169-7f959c10454e/jobs

Headers :

```
{"Content-Type": "application/json", "Accept": "application/json"}
```


Body:

```
{
    "inputs": [
        {
            "id": "dataset",
            "value": "8d76da9c-39f9-46a0-9342-af91c76724f5"
        },
        {
            "id": "model",
            "value": "35440e4d-a02d-4e5c-924d-62d03e0d5199"
        }
    ]
}
```


RESPONSE :
Body:

```
{
    "jobID": "0c542ebd-820a-4ed2-acda-b9a21638de3b",
    "status": "accepted",
    "location": "https://ccfb.crim.ca/processes/d8cfbd04-8c33-42f9-b169-
7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b"
}
```

5. Poll the job status until successful completion (status highlighted below).

REQUEST:
GET
https://ccfb.crim.ca/processes/d8cfbd04-8c33-42f9-b169-7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b

Headers :
{"Content-Type": "application/json", "Accept": "application/json"}

RESPONSE :
Body:

```
{
    "meta": {
        "code": 200,
        "type": "application/json",
        "detail": "Get process successful.",
        "route": "/processes/d8cfbd04-8c33-42f9-b169-
7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b",
        "uri": "http://ccfb.crim.ca/processes/d8cfbd04-8c33-42f9-b169-
7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b",
        "method": "GET",
        "process_uuid": "d8cfbd04-8c33-42f9-b169-7f959c10454e",
        "job_uuid": "0c542ebd-820a-4ed2-acda-b9a21638de3b"
    },
    "data": {
        "job": {
            "uuid": "0c542ebd-820a-4ed2-acda-b9a21638de3b",
            "task_uuid": "9c5828ea-7361-42cc-8ed7-83d31c844b68",
            "service_uuid": null,
            "process_uuid": "d8cfbd04-8c33-42f9-b169-7f959c10454e",
            "user_uuid": null,
            "inputs": [
                {
                    "id": "dataset",
                    "value": "8d76da9c-39f9-46a0-9342-af91c76724f5"
                },
                {
                    "id": "model",
                    "value": "35440e4d-a02d-4e5c-924d-62d03e0d5199"
                }
            ],
            "status": "succeeded",
            "status_message": "Job <0c542ebd-820a-4ed2-acda-
b9a21638de3b> done.",
            "status_location": "https://ccfb.crim.ca/processes/d8cfbd04-
8c33-42f9-b169-7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b",
            "execute_async": true,
            "is_workflow": false,
            "created": "2019-01-18T21:29:58.463000+00:00",
            "finished": "2019-01-18T21:30:13.905000+00:00",
            "duration": "0:00:15",
            "progress": 100,
            "tags": [
                "ml"
            ]
        }
    }
}
```

6. Retrieve prediction results to be used with GeoServer or other visualization tool.

REQUEST:
GET
https://ccfb.crim.ca/processes/d8cfbd04-8c33-42f9-b169-7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b/result

Headers :
{"Content-Type": "application/json", "Accept": "application/json"}

RESPONSE :
Body:

```
{
    "meta": {
        "code": 200,
        "type": "application/json",
        "detail": "Get process job result successful.",
        "route": "/processes/d8cfbd04-8c33-42f9-b169-
7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b/result",
        "uri": "http://ccfb.crim.ca/processes/d8cfbd04-8c33-42f9-b169-
7f959c10454e/jobs/0c542ebd-820a-4ed2-acda-b9a21638de3b/result",
        "method": "GET",
        "process_uuid": "d8cfbd04-8c33-42f9-b169-7f959c10454e",
        "job_uuid": "0c542ebd-820a-4ed2-acda-b9a21638de3b"
    },
    "data": {
        "outputs": [
            {
                "id": "classes",
                "value": [
                    "Pj",
                    "Alt",
                    <...trimmed...>
                    "La",
                    "Bf"
                ]
            },
            {
                "id": "predictions",
                "value": [
                    {
                        "predictions": [
                            -21.072824478149414,
                            <...trimmed...>
                            -14.791350364685059
                        ],
                        "gt": 4,
                        "geo_tag": [
                            580649.226610959,
                            5123274.7094423,
                            310.577713521936
                        ],
                        "geo_bbox": null,
                        "id": "ouareau2"
                    },
                    <...trimmed...>
                    {
                        "predictions": [
                            -2.3589096069335938,
                            <...trimmed...>
                            -10.327459335327148
                        ],
                        "gt": 26,
                        "geo_tag": [
                            580634.814107597,
```

```
                                5123116.49235243,
                                315.395711425078
                            ],
                            "geo_bbox": null,
                            "id": "ouareau50"
                            "geo_bbox": null,
                            "id": "ouareau50"
                        }
                    ]
                }
            ]
        }
}
```


6.2 Additional experiments

During the project, we focused on training models for tree species classification at the patch (or image) level instead of focusing on the pixel level using semantic or instance-based segmentation architectures due to limitations in the available data. The technique proposed by Long et al. (2015) can be used to extend our classifications models to produce approximate pixel-level segmentation maps. In short, the authors proposed a technique to "rewire" the last densely connected layer of any typical convolutional model in order to reframe it as a fully convolutional model. That way, the modified model can be used to process large image patches and provide coarsely localized class predictions in those patches. While the upsampled class predictions are still quite rough and unlikely to be properly aligned with the image's original high-resolution content, this approach requires no additional training and leverages the model's weights that were learned solely for image classification.

We applied the fully convolutional transformation of Long et al. (2015) on the ResNet models we trained to assess their "out-of-the-box" segmentation capabilities despite having no annotations to evaluate their results quantitatively. Using the Petawawa orthomosaics as input, the segmentation results obtained with a 23-species classifier are hard to assess (an example for a 160 m^2 patch analyzed at roughly 5 cm/pixel is shown in Figure 28). Since the classifier does not contain a "background" class that encompasses non-forested areas (e.g. grass, roads), the results are quite noisy outside dense canopy regions. Nonetheless, these results may help determine the proportions of tree species over very large forested areas without having to resort to a sliding window approach.

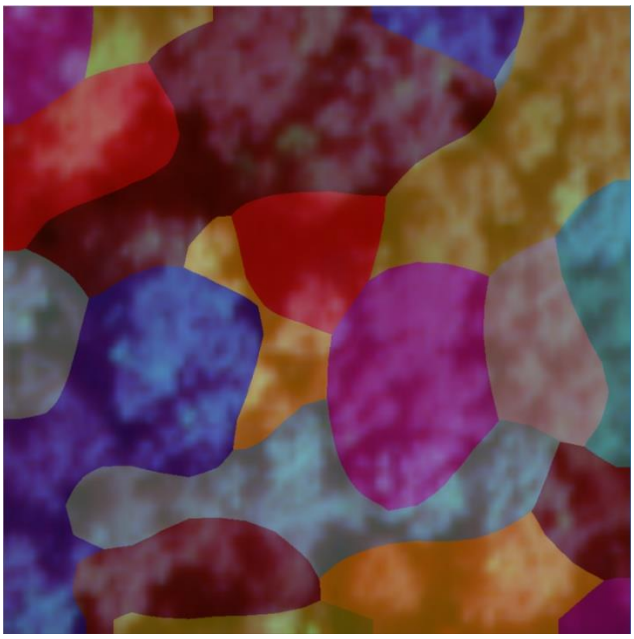

Figure 28 – Visualization of the segmentation map obtained for a 40 m x 40 m image by transforming a 23-species classifier into a fully convolutional network. The segmentation map is overlayed on top of the input image, and each color represents a different class.

As a further experiment, we retrained a ResNet classification model on the Petawawa dataset by combining the 23 original labels into two metacategories for deciduous and coniferous species. This yielded a classifier with an accuracy of more than 96% on a test set withheld from the training data according to a 90%-10% split. In this case, the segmentation model obtained using Long et al.'s technique produced maps that were much easier to interpret, and that seemed to fit the contours of trees more adequately (see Figure 29). The large-scale analysis of the proportion of deciduous-to-coniferous trees in a large region using this new model would be much more precise than with the 23-species model.

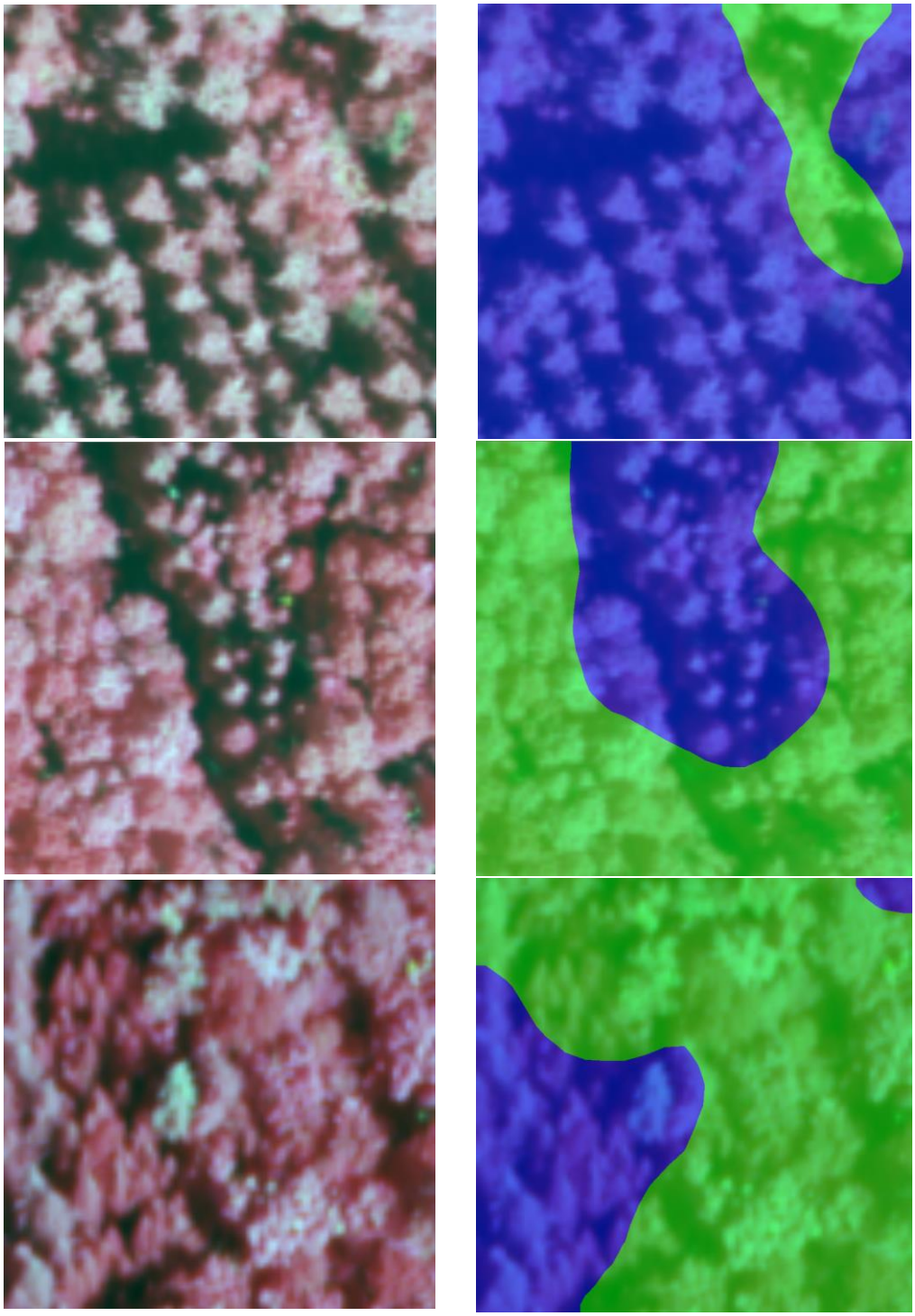

Figure 29 – Visualizations of segmentation maps obtained for 40 m x 40 m images by transforming a deciduous/coniferous classifier into a fully convolutional network. The segmentation map is overlayed on top of the input image on the right; green-tinted pixels indicate deciduous and blue coniferous.

7. OUTREACH

We have planned to present our findings in two different venues. The first is the 40th Canadian Symposium on Remote Sensing and Geomatics (CSRS), which will be held in Fredericton, N.B., June 4th to June 6th, 2019. An abstract has been submitted and accepted for an oral presentation. The second expected venue is the 87th Conference of the Association Francophone pour le Savoir (ACFAS), which will be held in Gatineau, QC, May 27th to May 31st, 2019. In this case, an abstract has been submitted, and we are still awaiting a decision as of early March.

8. FINAL DISCUSSION AND CONCLUSION

Over the span of this project, roughly 90 models were produced over 7 architectures. Multiple configurations and data augmentation techniques have been used to achieve better results. We even explored pixel-level segmentation in the last testing activity. We deployed two services over the cloud infrastructure, one to perform classification over a dataset using a selected model and the other to allow visualisation of the classification result using the WMS standard. We also provided a test sequence that has been used to validate the 6 steps required to run a classification. In the end, two conference presentations will be given, directly based on the work done in this project, two orthomosaics have been made available to download and we have open sourced our machine learning framework.

As we conclude this report, here we have the final values of the performance indicators:

Table 12 - Performance indicators

Performance indicators	Value
1. Number of models and architectures tested (activity 2)	Roughly 90 models over 7 architectures
2. Number of cloud-based services deployed on the CRIM infrastructure (activity 3)	2
3. Number of tests done over the cloud infrastructure (activity 4)	6
4. Number of conference article written and published (activity 5)	2

9. REFERENCES

M. Carpentier, P. Giguere, J. Gaudreault, “Tree species identification from bark images using convolutional neural networks”. arXiv (2018) 1803.00949

K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition”, arXiv (2015) 1512.03385

J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu, “Squeeze-and-Excitation Networks”, arXiv (2017) 1709.01507.

G. Huang, Z. Liu, L. Van der Maaten, K.Q. Weinberger, "Densely Connected Convolutional Networks", arXiv (2017) 1608.06993.

J. Long, E. Shelhamer, T. Darrell, “Fully convolutional networks for semantic segmentation”, in Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2015, pp. 3431-3440.

C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, “Inception-v4, Inception-ResNet and the impact of Residual Connections on Learning”, arXiv (2016) 1602.07261.

APPENDIX

A. DISTRIBUTION OF TREE SPECIES LABELS

Tree species (Code)	Petawawa	Ouareau
Unknown (Abl)	156 (1%)	0 (0%)
Populus Grandidentata (Alt)	1199 (12%)	0 (0%)
Populus Tremuloides (At)	1028 (10%)	18 (20%)
Fraxinus Americana (Aw)	11 (0%)	0 (0%)
Unknown (Ba)	111 (1%)	0 (0%)
Fagus Grandifolia (Be)	158 (1%)	0 (0%)
Abies Balsamea (Bf)	0 (0%)	4 (4%)
Betula Papyrifera (Bw)	412 (4%)	14 (16%)
Betula Alleghaniensis (By)	182 (1%)	0 (0%)
Thuja Occidentalis (Ce)	19 (0%)	8 (9%)
Unknown (Fb)	400 (4%)	0 (0%)
Tsuga Canadensis (He)	21 (0%)	0 (0%)
Unknown (Iw)	1 (0%)	0 (0%)
Larix Laricina (La)	0 (0%)	2 (2%)
Unknown (Lt)	193 (2%)	0 (0%)
Acer Saccharum (Mh)	439 (4%)	8 (9%)
Acer Rubrum (Mr)	351 (3%)	1 (1%)
Quercus Rubra (Or)	698 (7%)	0 (0%)
Quercus Alba (Ow)	1 (0%)	0 (0%)
Unknown (Pb)	71 (0%)	0 (0%)
Pinus Banksiana (Pj)	570 (6%)	0 (0%)
Pinus Resinosa (Pr)	1426 (15%)	7 (8%)
Unknown (Ps)	26 (0%)	0 (0%)
Pinus Strobus (Pw)	549 (5%)	11 (12%)
Picea Mariana (Sb)	318 (3%)	1 (1%)
Picea Glauca (Sw)	1005 (10%)	13 (14%)
Unknown (XP)	41 (0%)	0 (0%)
Total	**9386**	**87**