CRIM

CENTRE DE
RECHERCHE INFORMATIQUE
DE MONTRÉAL

Apache Spark MLlib applied to geospatial imagery for flood indication

APACHECON 2018

Jean-Francois Rajotte, Tom Landry, Cédric Noiseux, Martin Sotir, Ayoub Tlili, Mario Beaulieu, Samuel Foucher

Montréal, Québec, Canada
SEPTEMBER 26TH, 2018

crim.ca

PRINCIPAL PARTENAIRE FINANCIER

Économie, Science et Innovation
Québec

CRIM | Centre de recherche informatique de Montréal

- Applied research center
- 4 teams : **Vision**, NLP & Speech, **Data Science**, Model
- More about us
  - www.crim.ca
  - https://www.linkedin.com/company/crim/
  - https://medium.com/crim

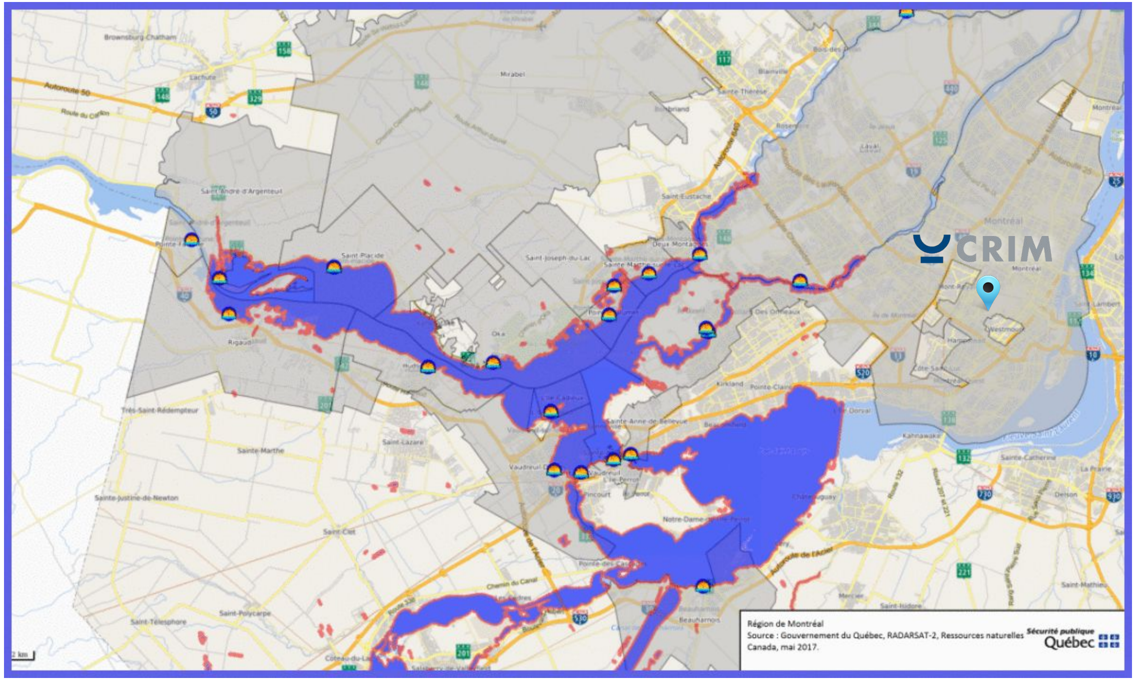
Overview

- Motivation
  - “NextGen” project at CRIM to advance compute and Machine Learning
  - Participation to OGC Testbed-14, Earth Observation and Machine Learning threads/tasks
  - Domain expert method of water detection from satellite images in flooding scenarios
- Our approach
  - Apache Spark
  - Deep learning and transfer learning
  - Deep Learning Pipelines
- Water detection at scale
  - Deep learning approach
  - Distributed detection with DLP

Motivation : OGC Testbed-14

- CRIM editors of OGC 18-038 - ML Engineering Report (**ER**). Other contribs:
  - Training DL models to detect features on Very High Resolution (VHR) imagery
  - **Processing SAR (radar) images in ML systems**
- CRIM as participants in Earth Observation & Clouds (**EOC**)
  - Implement application packaging, workflows and app deployment
  - **Application for flood detection using optical & SAR imagery**
- Same remote sensing techniques and data used in EOC and ML ER
- Both considers a disaster scenario: flooding

Motivation : disaster in Québec!



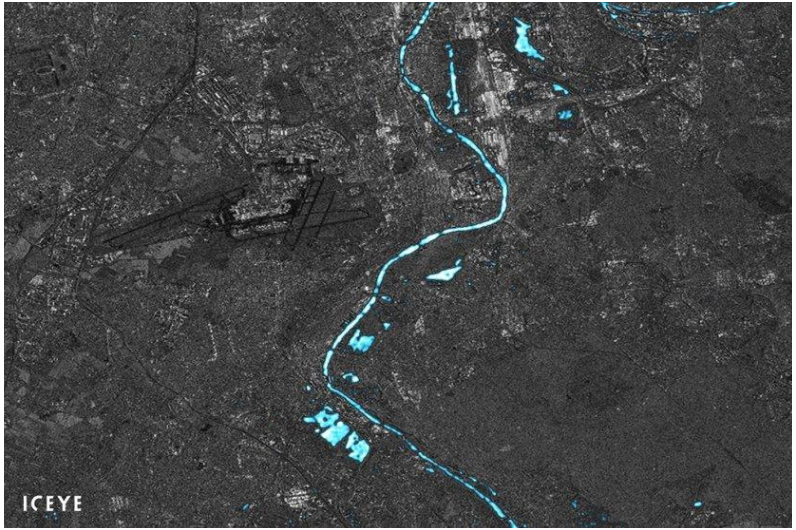
2017 flood map in Montreal area, made from RADARSAT-2 images.
Source: Quebec Ministry of Public Safety

Lac Saint-Pierre, Québec

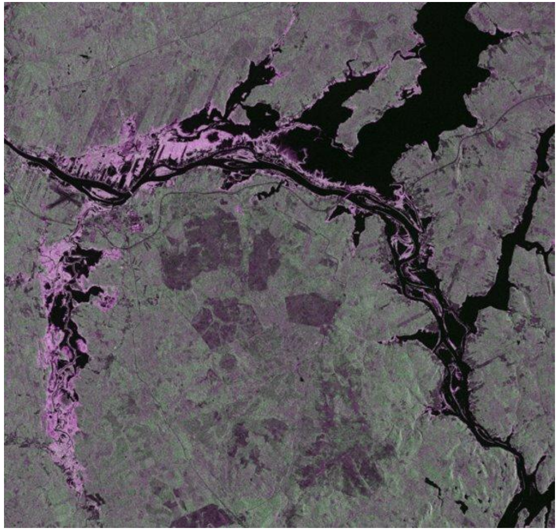
Deux-Montagnes, Québec

Satellite Radar Imagery

- Seeing large regions through clouds valuable in disasters.
- Detect water from floods, but also ice that clogs rivers.
- Great contrast between water and non-water.



Source: IceEye, ESA



Source: MDA

Active sensing: Synthetic Aperture Radar (SAR)

- Optical sensors (CCD, CMOS) are passive, only measuring reflected sunlight
- Radar sensors are *active*
  - microwave beam is sent down towards Earth
  - reflections read by sensor at different polarizations
- Sentinel-1 satellite:
  - 20m x 20m pixel resolution (after geometric corrections)
  - Up to 250km swath

Sentinel-1 radar vision. Credit: ESA/ATG medialab

Water Detection on Satellite Radar Images

Motivation :

- Simple use case to showcase the **scalability** of satellite image analysis.
- Scientists working on water detection at CRIM

Approach :

Replace domain expert method with neural network in **Apache Spark** to reproduce its results.

Raw images → Neural Network → Water detection

Note: we are not the first to use deep learning on satellite images for water detection


Sentinel-1 radar IW image - ESA

Water Detection : Domain Expert Method

- Acquire raw images
- Geometric correction (satellite-earth interactions)
- Handcrafted feature : Structural Feature Set (related to local homogeneity)
- Mean filter (reduce noise)
- Thresholding (water definition)

→ Water segmentation

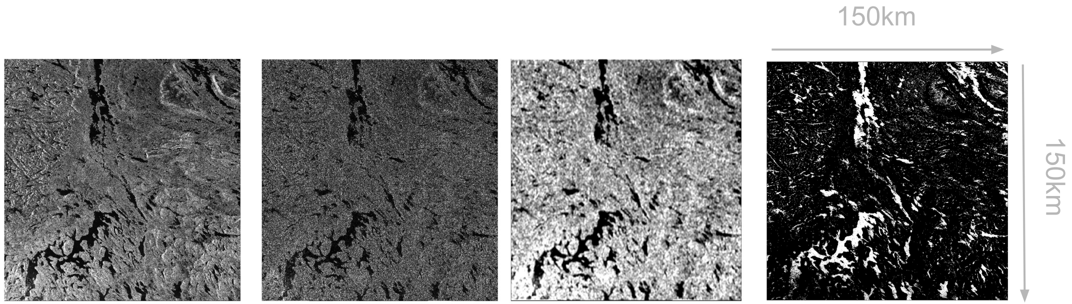
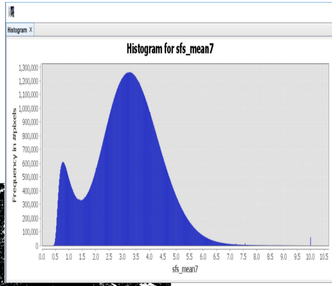


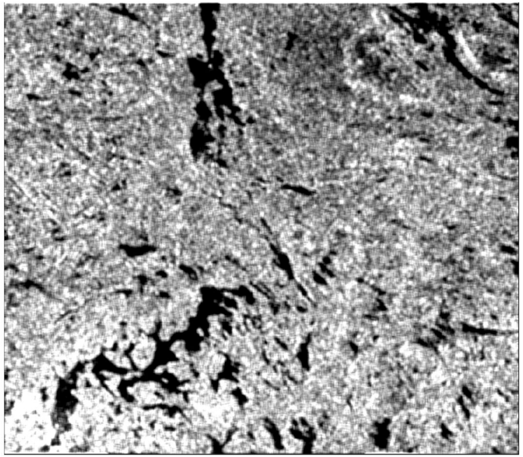
Image Sentinel-1 en polarisation VV | Descripteur de Texture SFS | Filtre moyen 7*7 | Seuillage par la méthode Ostu's

Black ≠ Water
White = Water

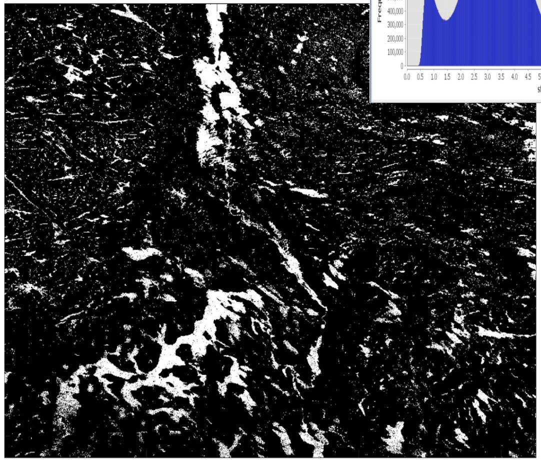
Water Detection : Domain Expert Method

Thresholding on SFS image





SFS + Filtre moyen 7*7



Seuillage par la méthode Ostu's

How to scale this method?

Currently : domain expert workstation using Sentinel Application Toolbox (SNAP).
Contains most remote sensing processes to build apps, but runs locally.

Processing performance issues:

- Storage
- Data locality

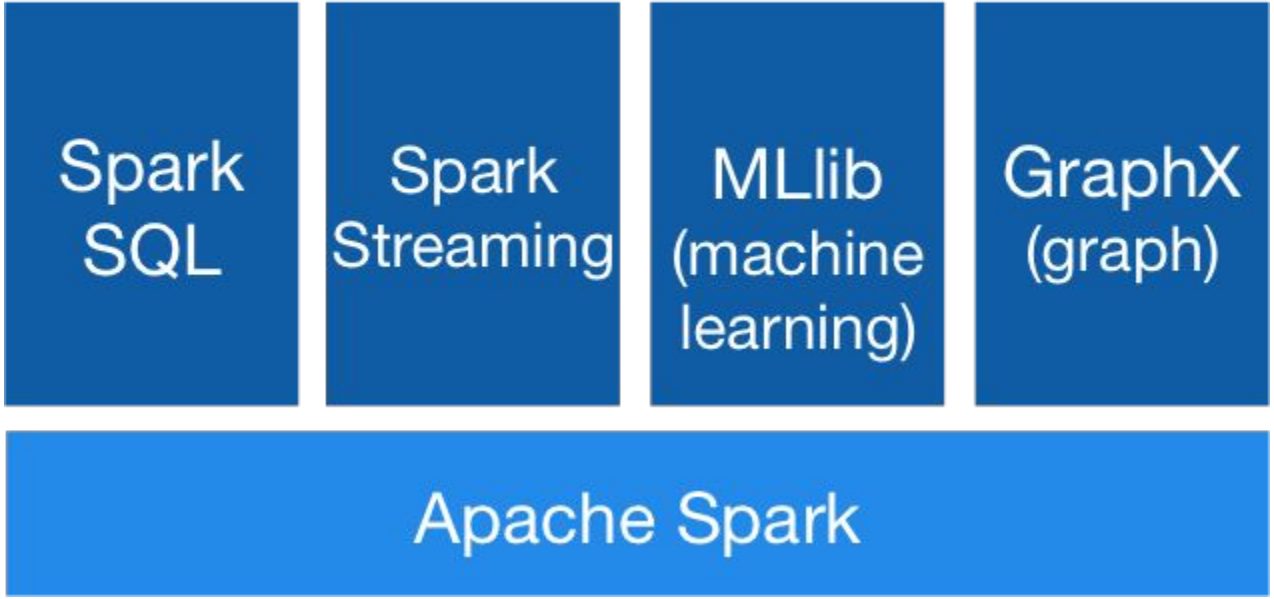
Other solutions:
- Cloud
- Microservices
- Lamda/server

Our solution

- Deep learning model **target** = domain expert water detection
- Distributed parallel computing
- → Apache Spark + HDFS + YARN

Apache Spark

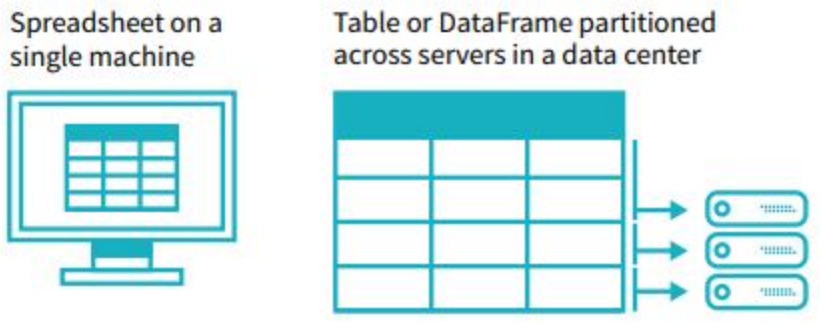
Open source distributed “in-memory” computing framework



Apache Spark

Dataframe

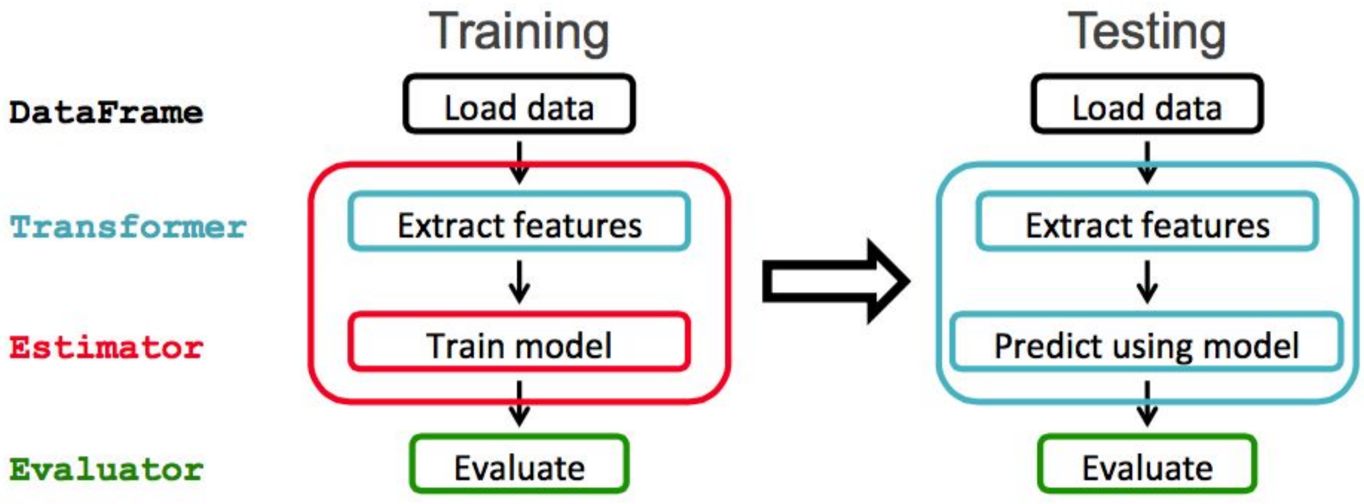
- Spark’s most common data structure API
- Concept inspired by R but distributed.



https://databricks.com/glossary/what-are-dataframes

MLlib

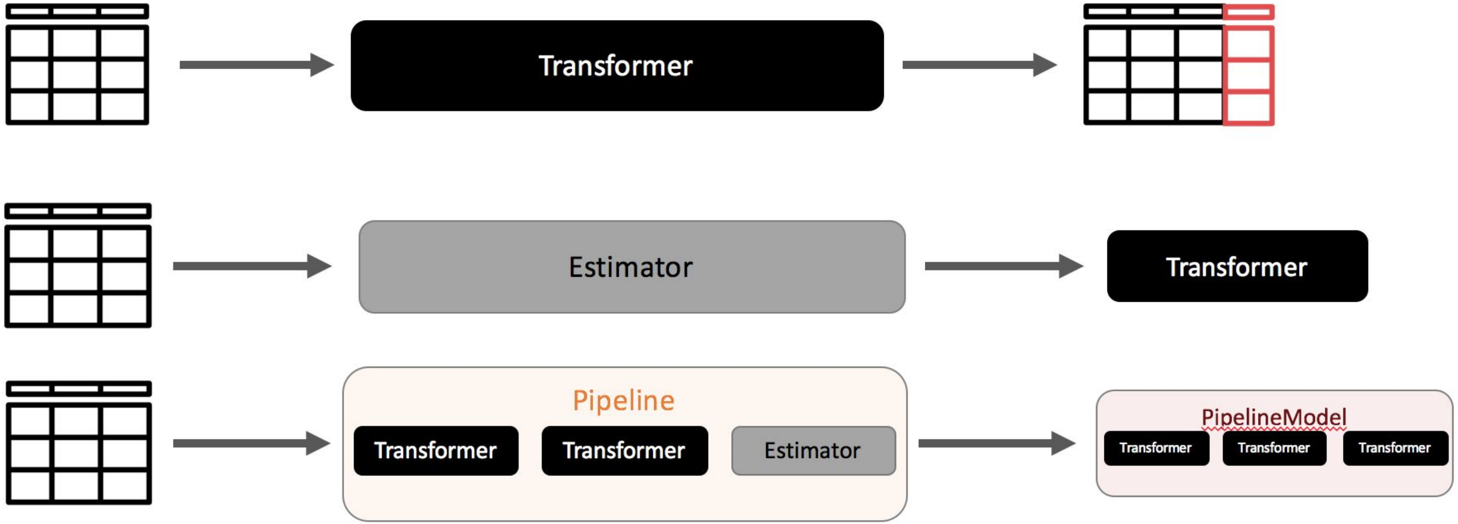
Machine learning library



https://databricks.com/session/deep-dive-into-deep-learning-pipelines

MLlib

Pipelines

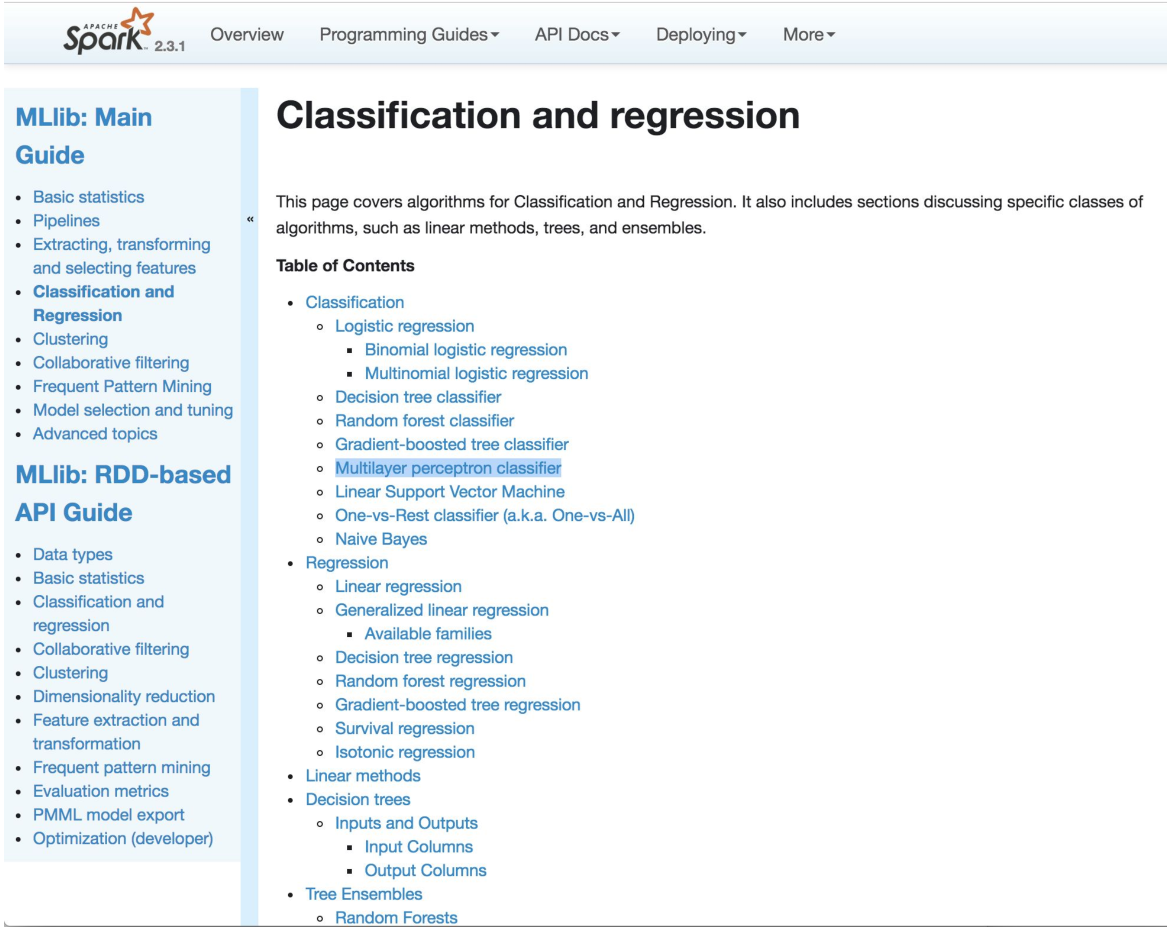


https://medium.com/rv-data/my-first-foray-into-spark-mllib-2907dde75f73

MLlib

No deep learning

…except Multilayer perceptron classifier



Spark 2.3.1 Overview Programming Guides API Docs Deploying More

MLlib: Main Guide

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- **Classification and Regression**
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

MLlib: RDD-based API Guide

- Data types
- Basic statistics
- Classification and regression
- Collaborative filtering
- Clustering
- Dimensionality reduction
- Feature extraction and transformation
- Frequent pattern mining
- Evaluation metrics
- PMML model export
- Optimization (developer)

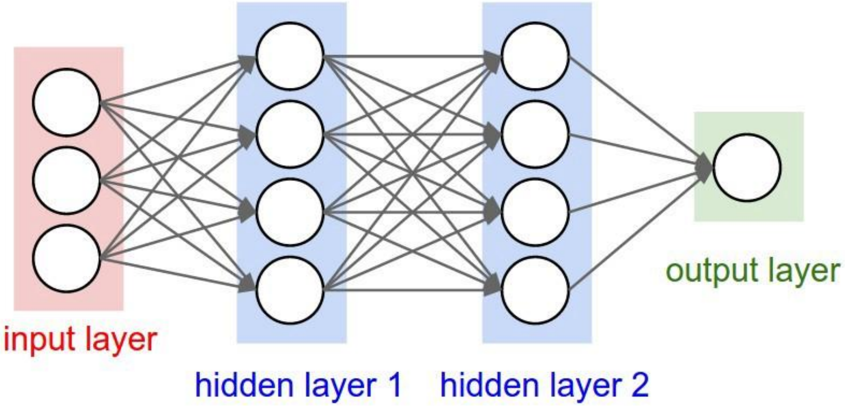
«

Classification and regression

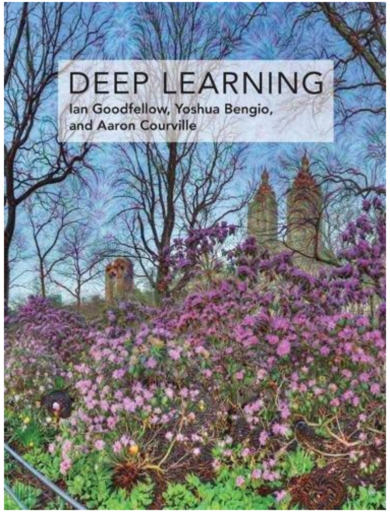
This page covers algorithms for Classification and Regression. It also includes sections discussing specific classes of algorithms, such as linear methods, trees, and ensembles.

Table of Contents

Deep Learning



For more information, read:



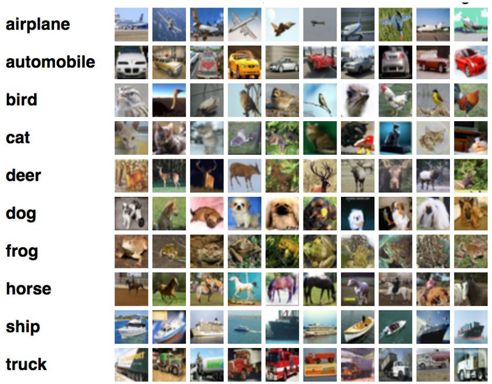
It is common practice to separate the data in two disjoint sets:

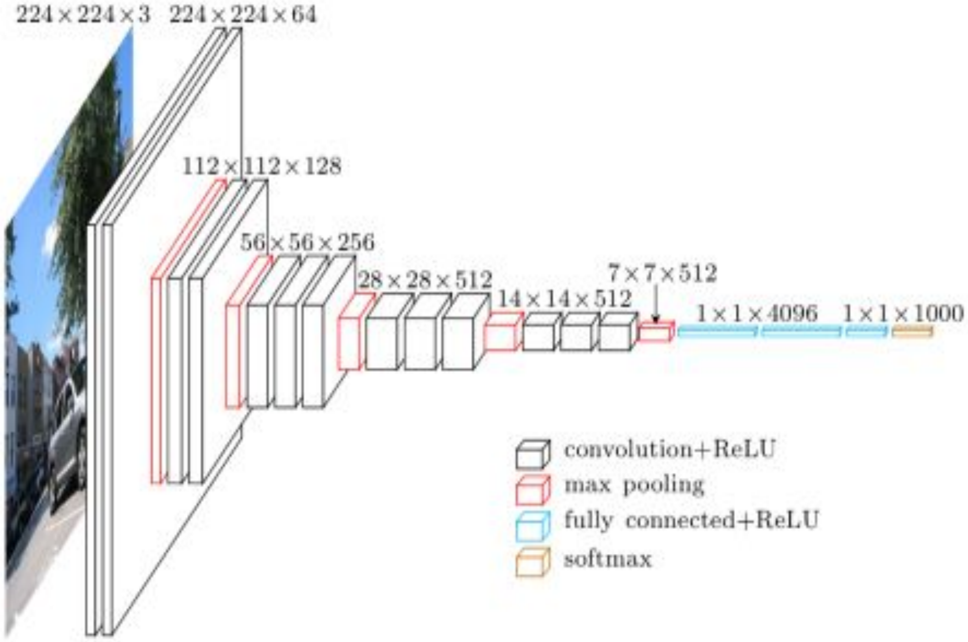
- **train set :** Used to teach the network a given task
- **test set :** To evaluate the network's performance on data never seen

Deep learning: vision

VGG-16

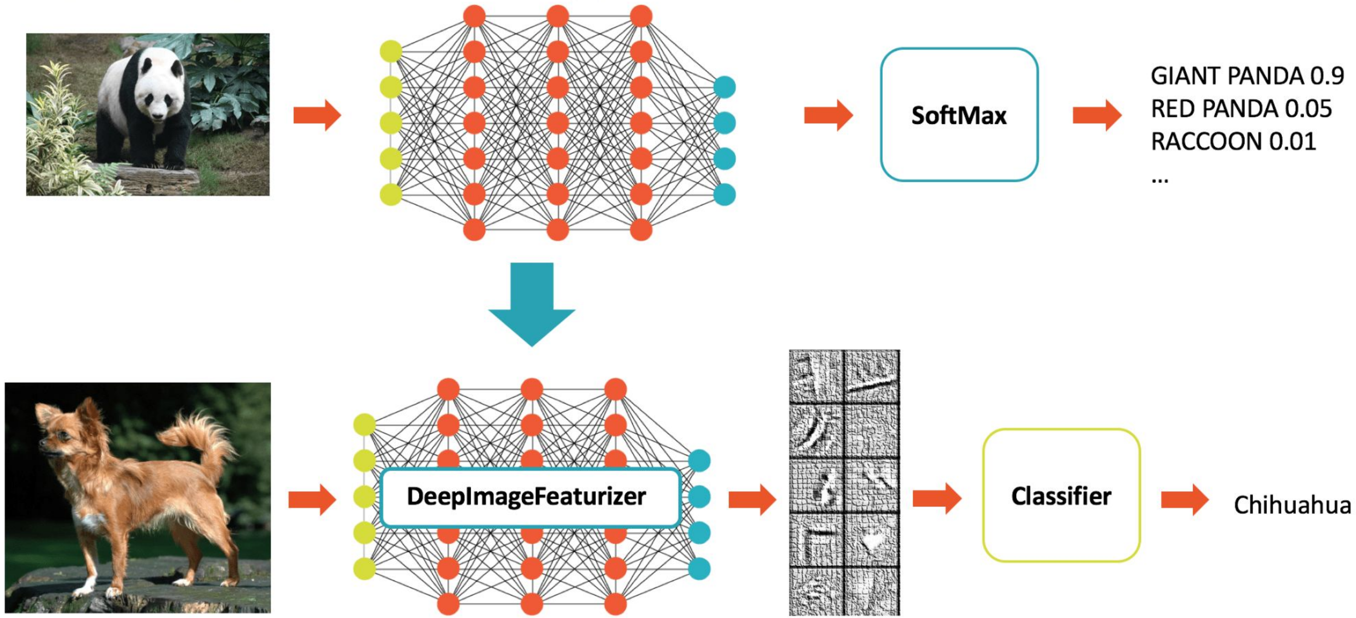
- Object recognition
- 2014 ImageNet competition winner
- Trained to classify images





Transfer learning

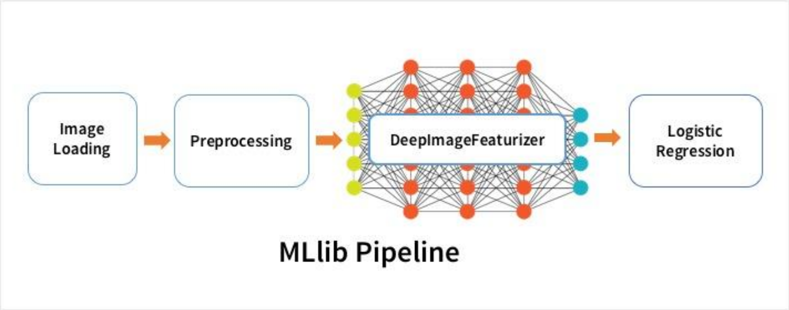
Needs much less training data than “original” training



Spark Deep Learning Pipelines

- From Databricks (released summer 2017)
- Open source
  https://github.com/databricks/tensorframes
  https://github.com/databricks/spark-deep-learning
- Based on Tensorframe (databricks) :
  "This package is experimental and is provided as a technical preview only. While the interfaces are all implemented and working, there are still some areas of **low performance**."
- Version 1.2

Transfer Learning as a Pipeline



Spark Deep Learning Pipelines

- Tensorflow and tensorflow-backed Keras
- Pre-trained model available through DeepImageFeaturizer

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from sparkdl import DeepImageFeaturizer

featurizer = DeepImageFeaturizer(inputCol = "image", outputCol = "features", modelName = "InceptionV3")
lr = LogisticRegression(maxIter = 20, regParam = 0.05, elasticNetParam = 0.3, labelCol = "label")
p = Pipeline(stages=[featurizer, lr])

p_model = p.fit(train_df)
```

- Transformers can also be created from custom TF/Keras models

Spark Deep Learning Pipelines

A native image support in Spark

Images loading (imageSchema) (Spark 2.3)

```
from pyspark.ml.image import ImageSchema
image_df = ImageSchema.readImages("/data/myimages")
```

Or, using a custom decoding function

```
from sparkdl.image import imageIO
image_df = imageIO.readImagesWithCustomFn(sample_img_dir, decode_f = imageIO.PIL_decode)
```

Spark Deep Learning Pipelines

Transfer learning with DeepImageFeaturizer

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml import Pipeline
from sparkdl import DeepImageFeaturizer

featurizer = DeepImageFeaturizer(inputCol="image", outputCol="features", modelName="InceptionV3")
lr = LogisticRegression(maxIter=20, regParam=0.05, elasticNetParam=0.3, labelCol="label")
p = Pipeline(stages=[featurizer, lr])

model = p.fit(train_images_df)    # train_images_df is a dataset of images and labels

# Inspect training error
df = model.transform(train_images_df.limit(10)).select("image", "probability",  "uri", "label")
predictionAndLabels = df.select("prediction", "label")
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
print("Training set accuracy = " + str(evaluator.evaluate(predictionAndLabels)))
```

Available pretrained models: InceptionV3, Xception, ResNet50, VGG16, VGG19

Spark Deep Learning Pipelines

Also available but **not** used here:

- Distributed hyperparameter tuning
- Deploying models as a SQL functions

```
from pyspark.ml.image import ImageSchema

image_df = ImageSchema.readImages(sample_img_dir)
image_df.registerTempTable("sample_images")
```

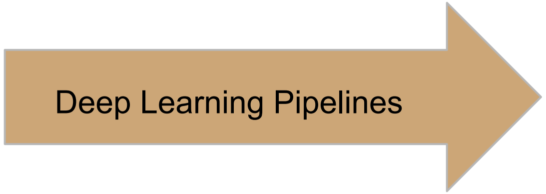
```
SELECT my_custom_keras_model_udf(image) as predictions from sample_images
```

Detection at scale

- Images are very large (dimension = 30k - 40k pixels)
- The whole conventional process takes about 3-4h of manipulation per radar image.

Goal : Leverage Deep Learning Pipelines shortcut the water detection process.

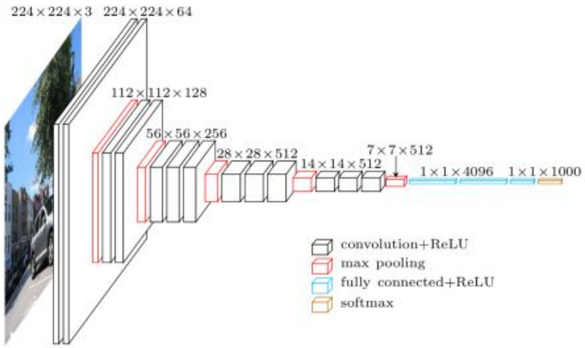
Image Sentinel-1
en polarisation VV

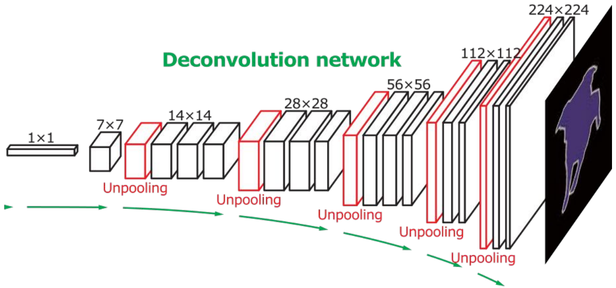


Seuillage par la méthode Ostu's

Detection at scale : model

Adding deconvolution layers to VGG-16 in order to make pixel-level prediction using KERAS.





http://cvlab.postech.ac.kr/research/deconvnet/

- Remove the top layers (Fully Connected + ReLu + softmax)
- Add **deconvolution layers** (Padding + Conv2D + Normalization + Upsampling) until desired output shape reached
- For pixel-level prediction, the output resolution equals the input image resolution (224 x 224)
- Image segmentation is then possible

Detection at scale

Our cluster

- 6 nodes: Total of 96 cores,
  750 Gb RAM for spark executors
- CPUs : 16 cores, 2.6 GHz Ivy Bridge
  Intel(R) Xeon(R) CPU E5-2650 v2 (2014)
- Spark 2.3, Yarn, HDFS
  (Cloudera distribution)
- Deep Learning Pipeline 1.2.0
  With Tensorframe 0.5 and Tensorflow 1.10

Our data

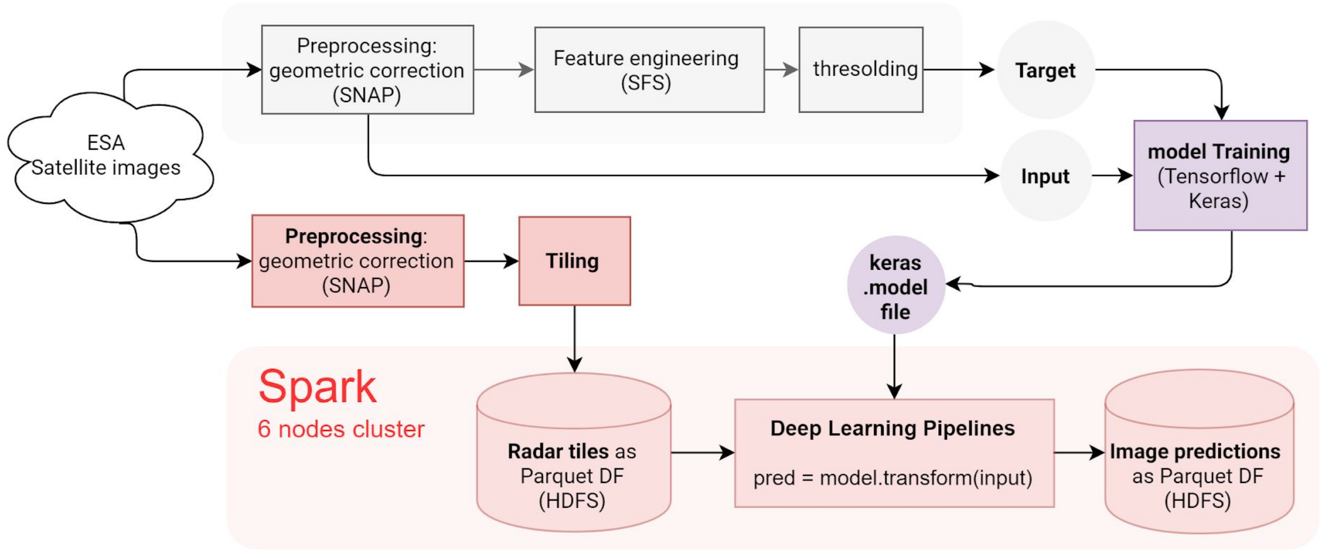
- Images from Québec (May ‘18)
- 130 satellite images (900Gb)
- 1.6M tiles of size 224x224
- 80 billions predicted pixels

- Model training set = 2000 tiles
- Test set = 500 tiles

Detection at scale

1. The domain expert workflow is used to create a training sets

Only a few images are used.



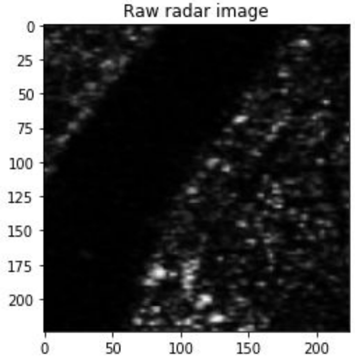
2. A convolutionnal neural network is trained to replicate the target water segmentation

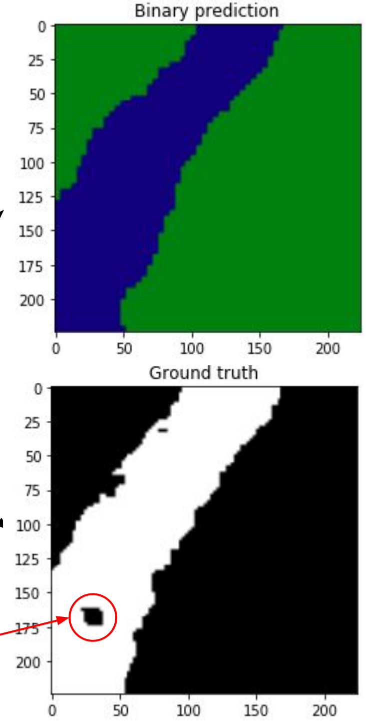
The model is trained with Keras on single machine (GPU recommeded).

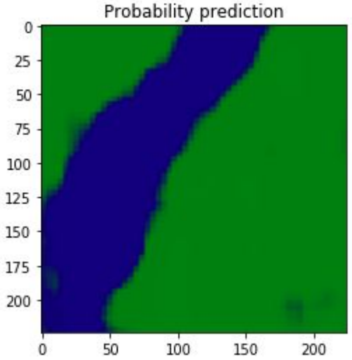
3. Batch segmentation are performed on the spark cluster

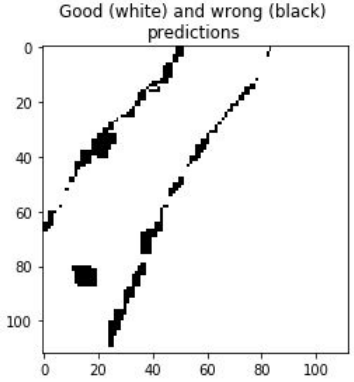
For now, geometric corrections and tiling are still performed out of the cluster.

Results: promising



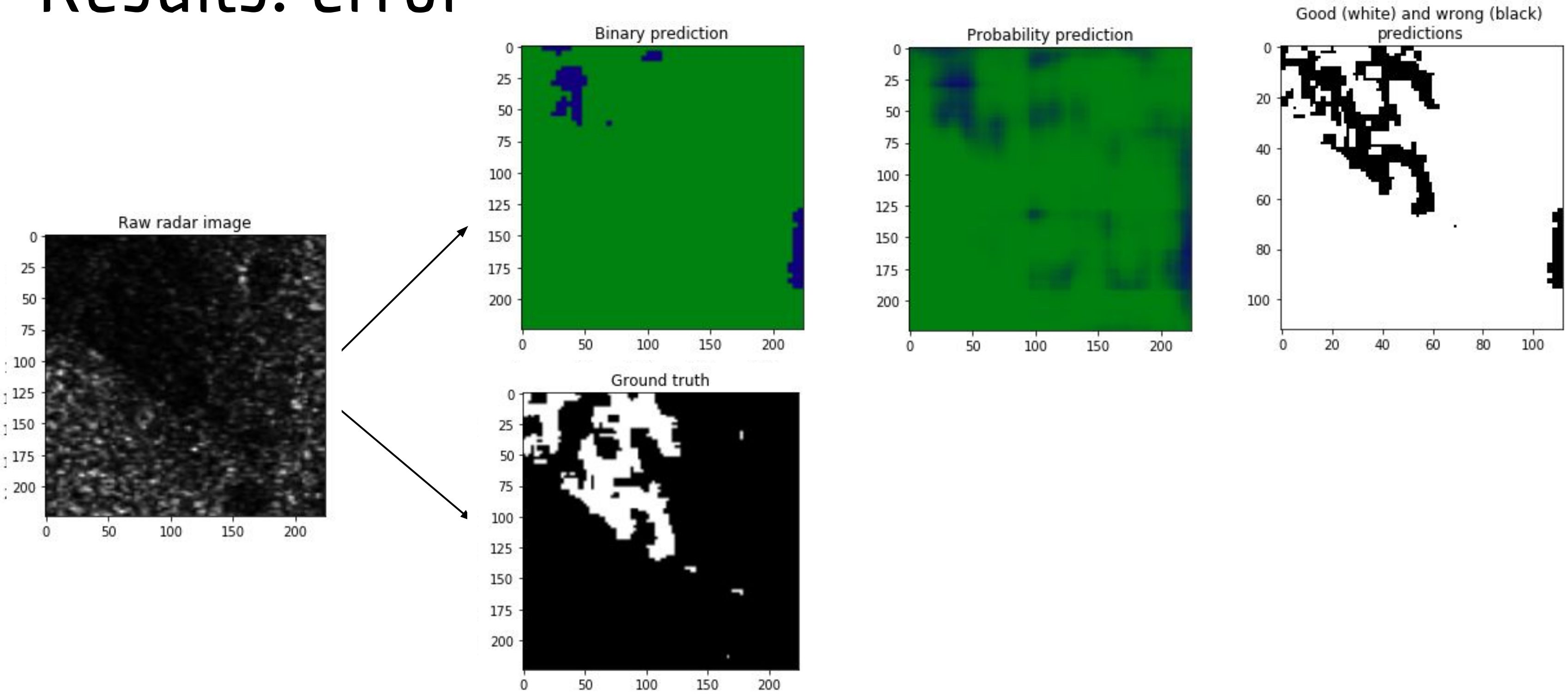






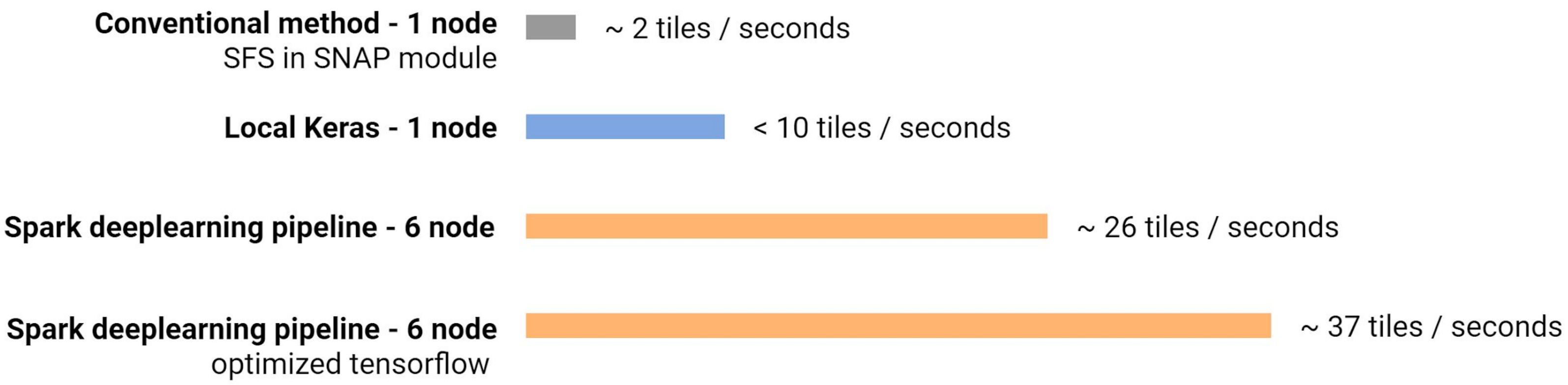
The domain expert method can introduce noise in the “detected” water, which is not detected in our predicted image for this example.

Results: error



Performance (preliminary results)

Our model: 100 MFlop
Input: 224 x 244 x 3



42% improvement on performance

Tensorflow java package compiled from sources using the **-march=native** flag (use SSE4.2 and AVX instructions sets)

Using GPUs would have been significatively faster, but Spark Deep Learning allows us to leverage our existing spark cluster and HDFS storage with no configuration modification.

Possible Improvements

Processing

- Add preprocessing on spark (GDAL, GeoTrellis, xarray)
  - Will allow to put images directly on HDFS
- Using GPU (Cloud, HPC clusters, ...)

Accuracy

- More training data
- Hyperparameter tuning
- Fully trained network

Lessons Learned

- DLP is a very simple way to predict with deep learning models at scale
- The technology is still very new and has many low hanging fruits of optimization
- Locally compiled tensorflow can lead to performance gain
- **Small images** storages on hdfs : serialized as parquet (spark ImageSchema) is a good option
- Looking for easy tool geospatial tools for Spark (to leverage more images info: time, geo...)
    - Gdal is difficult to use with spark + hdfs
    - Maybe geotrellis, geopyspark
    - Support of OGC standards to consume data, produce maps, store detections, find in catalogs, etc.
- Deconvolution method on VGG-16 gives reasonable results on prediction (WIP...)
- ImageNet transfer learning is applicable to non-optic images (cross-domain transfer)

Conclusion and Outlook

- DLP is a good solution to leverage existing Spark cluster into a prediction.
- DL is a promising solution to enhance existing “classical” geoscience models.
- Our approach can easily be adapted to
    - another (better) domain expert method
    - other types of detections (just need a target to train the network)

Further developments

- Flood monitoring with image time series provided by a data cube
- Use of other reference target domains (burn scars, ice, etc.)
- Advancement of ML best practices at OGC, towards standardisation

Backup

Some references

https://docs.databricks.com/applications/deep-learning/deep-learning-pipelines.html

https://databricks.com/blog/2017/06/06/databricks-vision-simplify-large-scale-deep-learning.html

https://towardsdatascience.com/deep-learning-with-apache-spark-part-2-2a2938a36d35

https://www.slideshare.net/databricks/build-scale-and-deploy-deep-learning-pipelines-using-apache-spark-90948963

Detection at scale

Processing flow