Packaging, deployment and interfacing of machine learning applications in scientific workflow environments

Tom Landry
Coordinator, geospatial expertise

IS-ENES3/ESGF Virtual Workshop on Compute and Analytics
December 2nd 2019

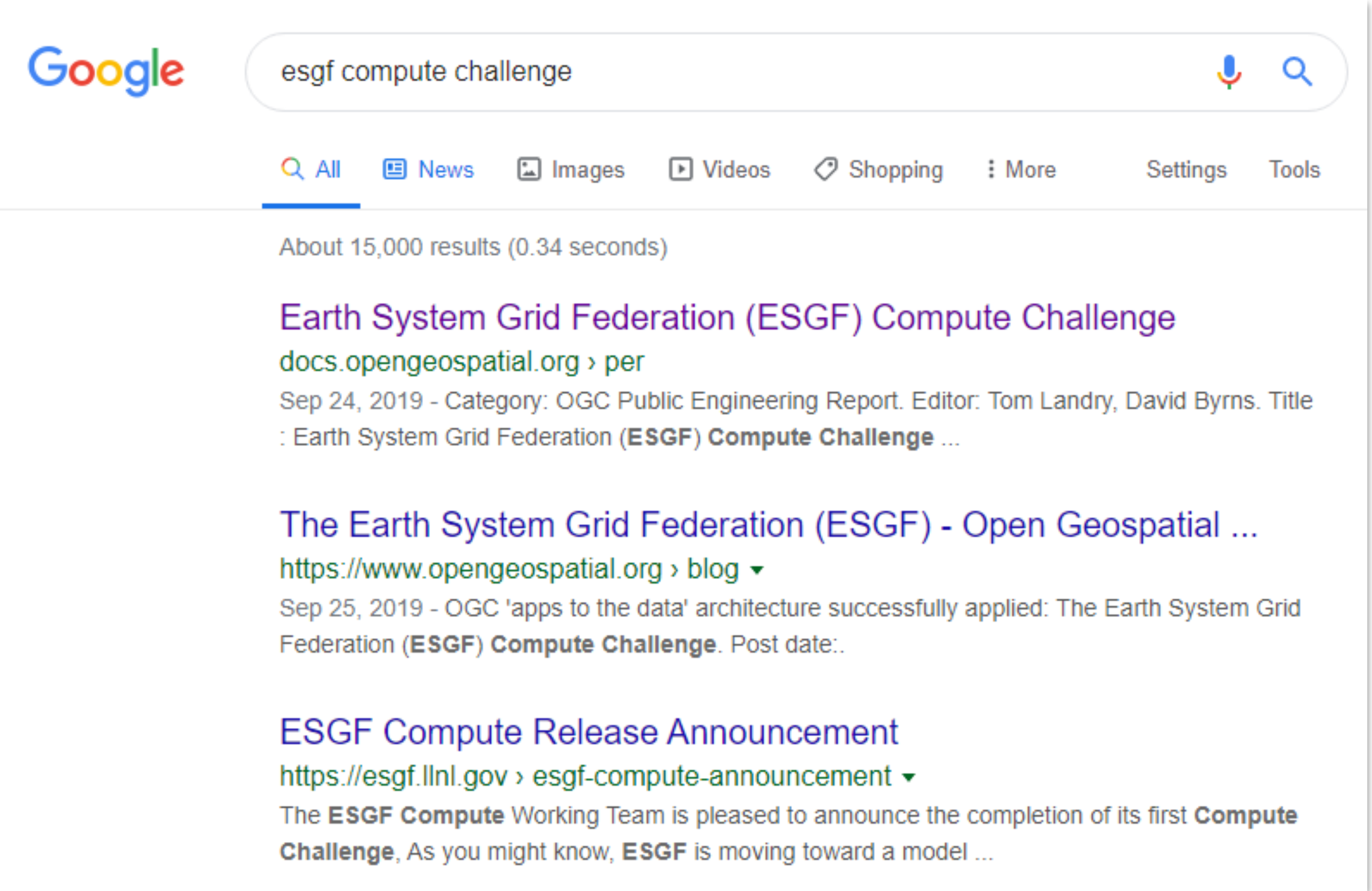
OGC®
Making location count.

CRIM

Agenda

- ESGF Compute challenge 2019
- EO Exploitation Platform open architecture
- Problem statement
- Technical approach
- Results
- Conclusion

A challenging task



About 15,000 results (0.34 seconds)

Earth System Grid Federation (ESGF) Compute Challenge
docs.opengeospatial.org › per
Sep 24, 2019 - Category: OGC Public Engineering Report. Editor: Tom Landry, David Byrns. Title : Earth System Grid Federation (**ESGF**) **Compute Challenge** ...

The Earth System Grid Federation (ESGF) - Open Geospatial ...
https://www.opengeospatial.org › blog ▾
Sep 25, 2019 - OGC 'apps to the data' architecture successfully applied: The Earth System Grid Federation (**ESGF**) **Compute Challenge**. Post date:.

ESGF Compute Release Announcement
https://esgf.llnl.gov › esgf-compute-announcement ▾
The **ESGF Compute** Working Team is pleased to announce the completion of its first **Compute Challenge**, As you might know, **ESGF** is moving toward a model ...

OGC®

CRIM

Platforms and ML in OGC Testbeds

Open architecture for Thematic Exploitation Platforms (**TEP**) relying on Mission Exploitation Platforms (**MEP**) for data and computing.

Application of geospatial ML on Earth Observation data to advance standards.

- Use of Common Workflow Language (CWL) for application chaining (TB-14)
- Use of an EMS and ADES pair on TEP and MEP (TB-14)
- WPS 2.0 REST interfaces includes quoting, billing, visibility, etc.(TB-14)
- Integration with ESGF Compute Working Team API for analytics (TB-14+)
  - see ESGF Compute Challenge Engineering Report
- Application discovery (TB-15 EO)
- Machine Learning pipelines (TB-15 ML)

Sponsor Testbed-13,14

Sponsor Testbed-13,14,15

Sponsor TB14+ ESGF

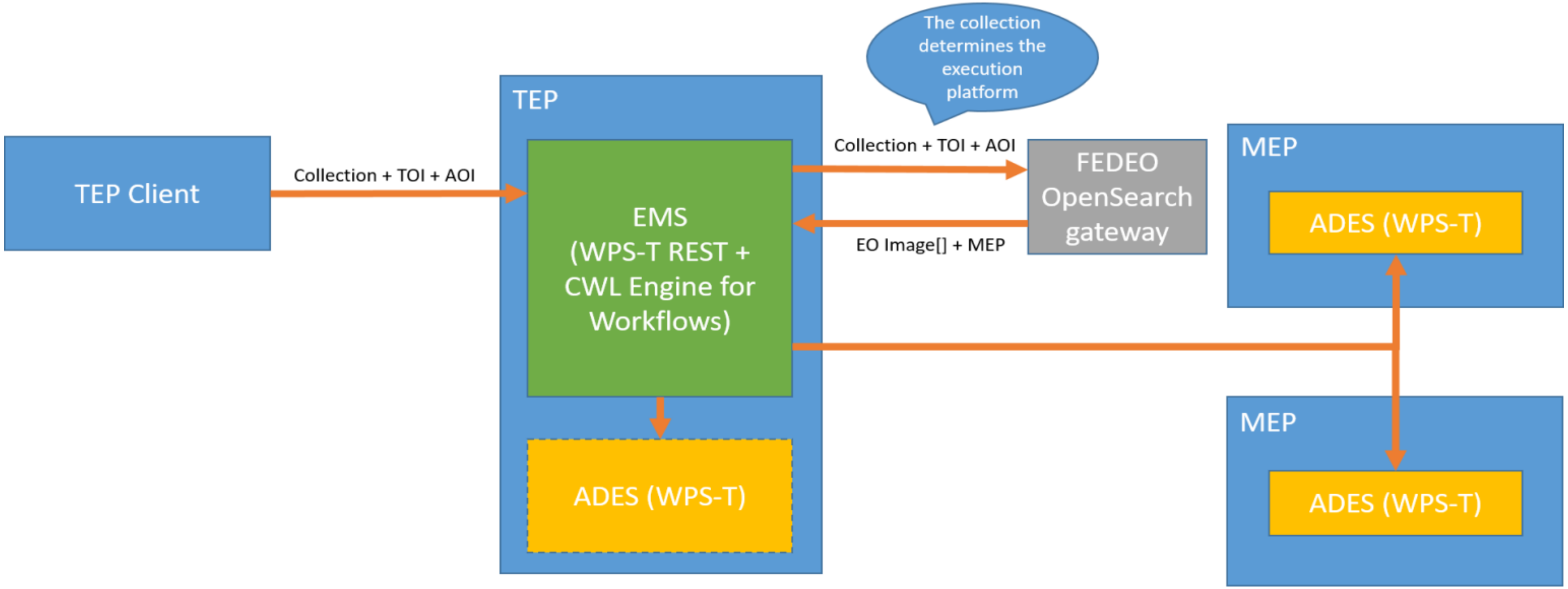
Sponsor Testbed-15

Testbed-14 EOC: ADES/EMS architecture

ADES - Application Deployment and Execution Service (application runner)

EMS - Execution Management Service (workflow orchestrator)

Our implementation: https://github.com/crim-ca/weaver



OGC® esa CRIM

The deliverable: lake-river differentiation model

Objective: Train model to recommend waterbody splits into lake and river features

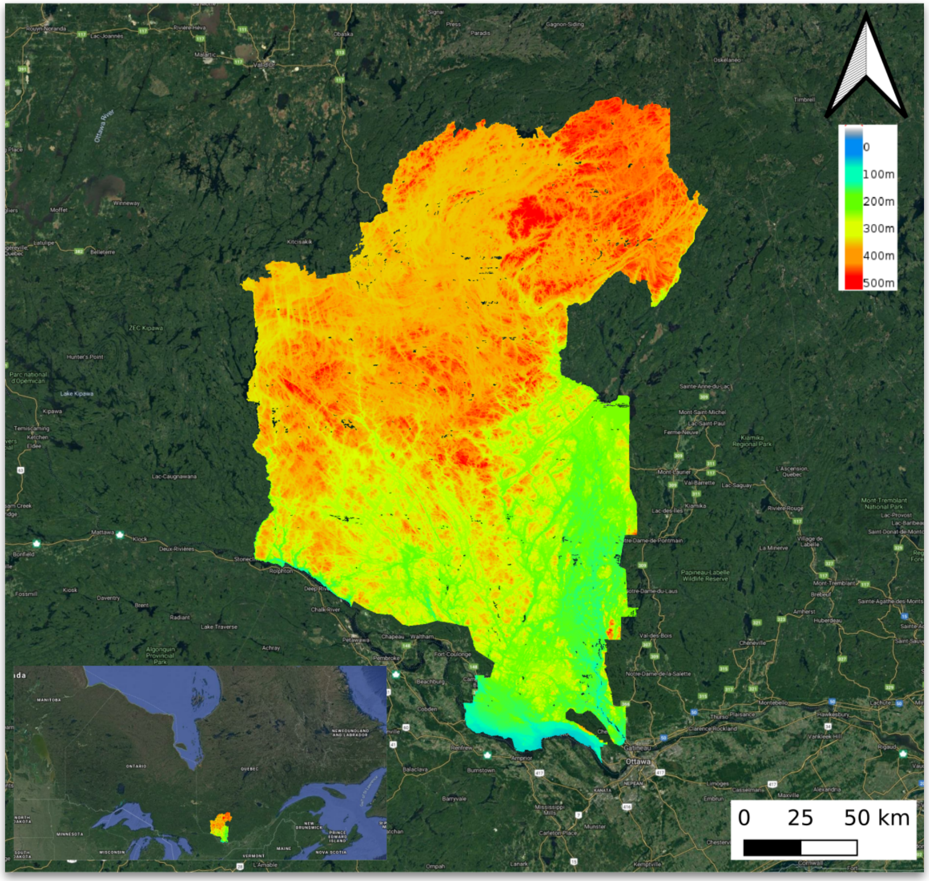
- If no split, determine if lake or river
  - Detect lakes!
- If split, determine division between the features
  - Hydro - lakes = rivers!

Data

- Hydrography network
- High Resolution DEM
- Imagery, if possible/necessary

Study area

- north of Gatineau/Ottawa
- near from Petawawa experimental forest



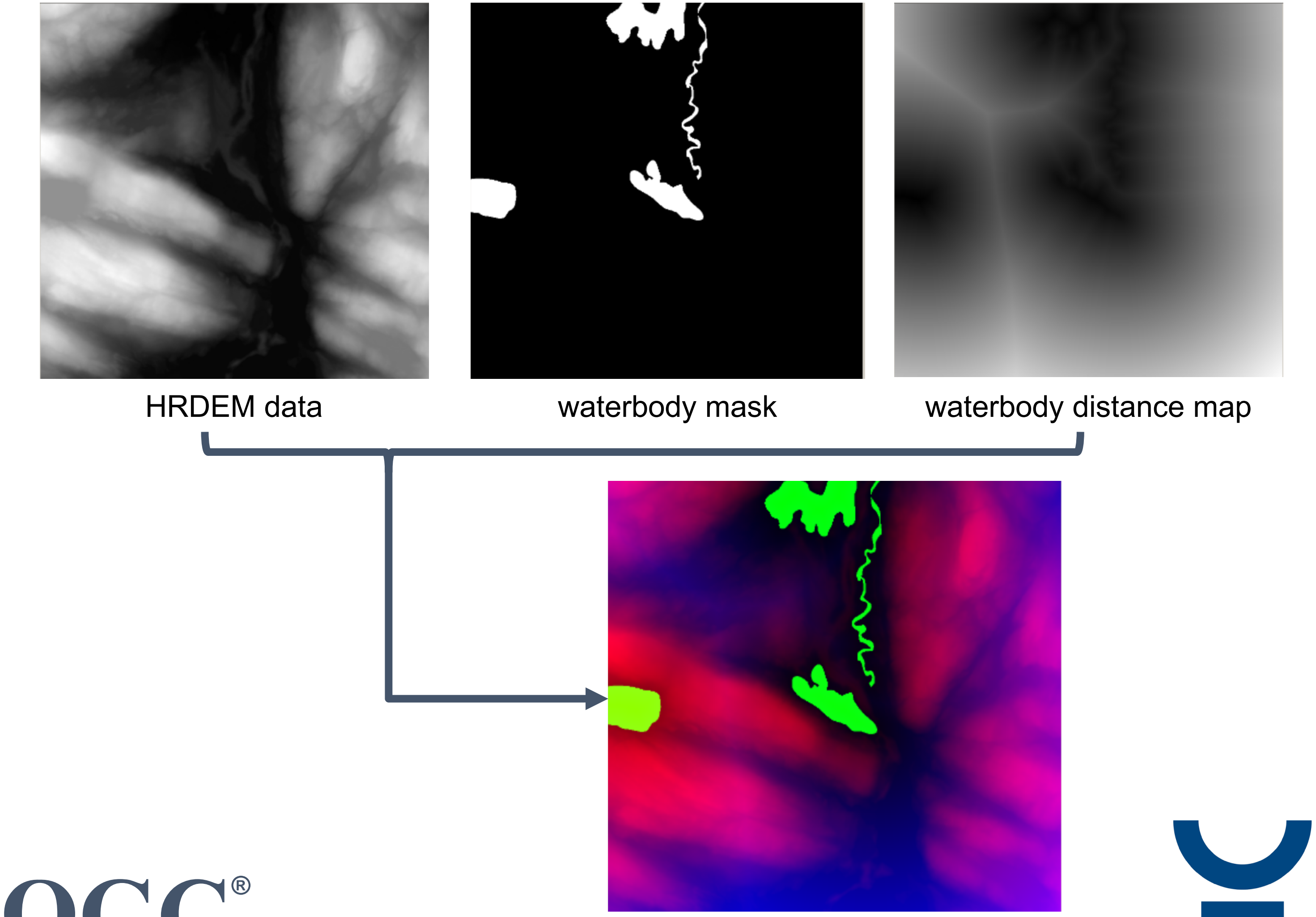
OGC®

Natural Resources Canada
Ressources naturelles Canada
Canada

CRIM

Input data for training

- **Merge LiDAR + waterbody geometries into 3D tensors**



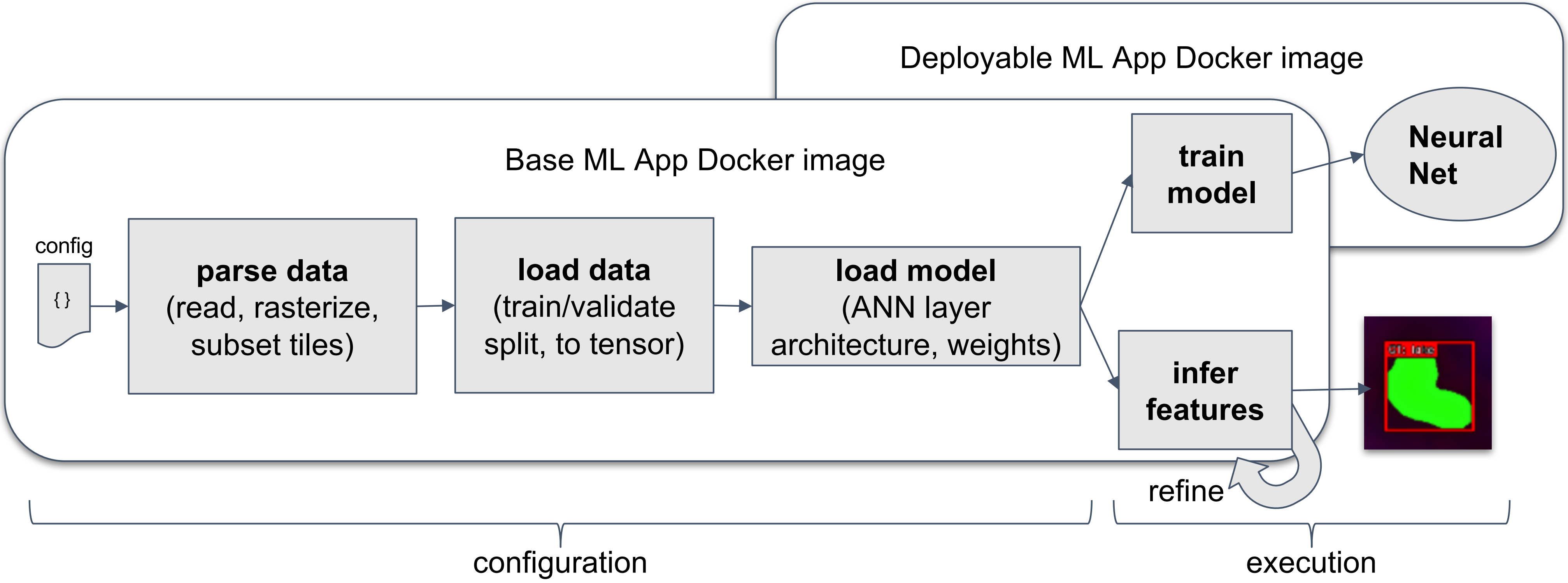
HRDEM data

waterbody mask

waterbody distance map

3-channel (RGB) tensor

TB-15 proposed ML workflow



Deployable ML App Docker image

Base ML App Docker image

config

{ }

parse data
(read, rasterize, subset tiles)

load data
(train/validate split, to tensor)

load model
(ANN layer architecture, weights)

train model

Neural Net

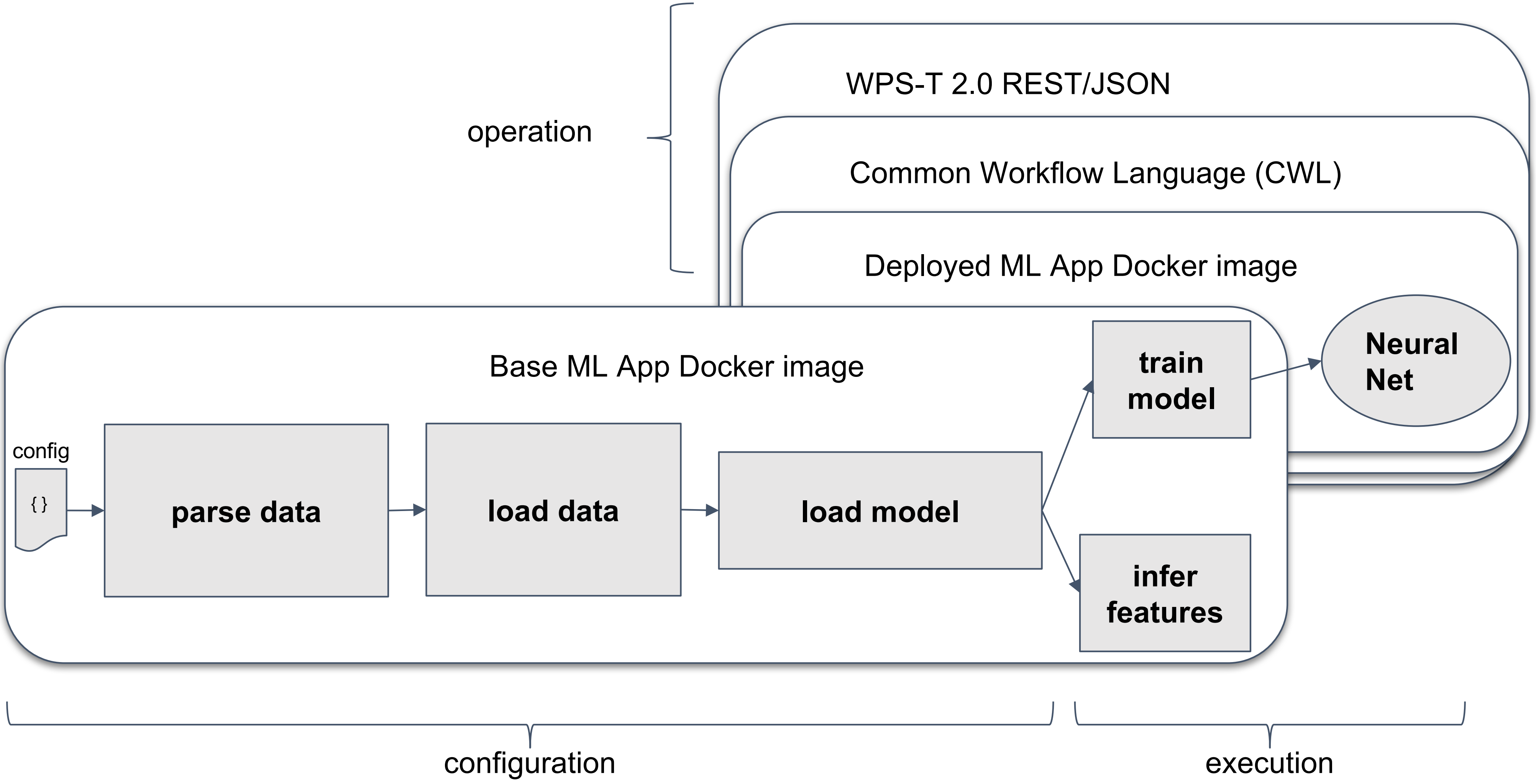
infer features

refine

configuration

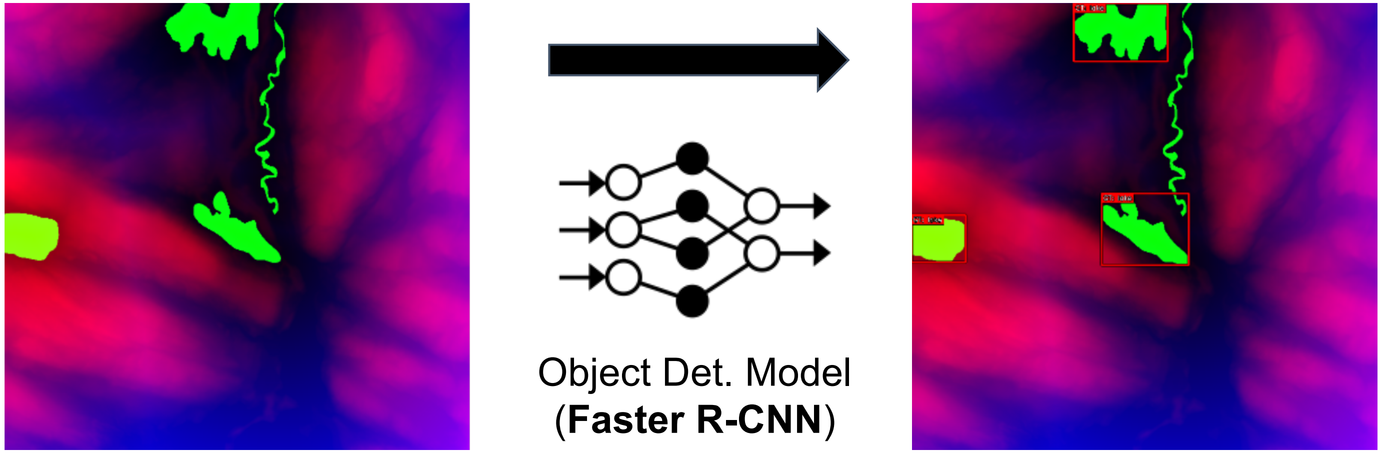
execution

Operations on ML App



Model I/O definitions

- **From input, infer bounding boxes for lakes (detection)**

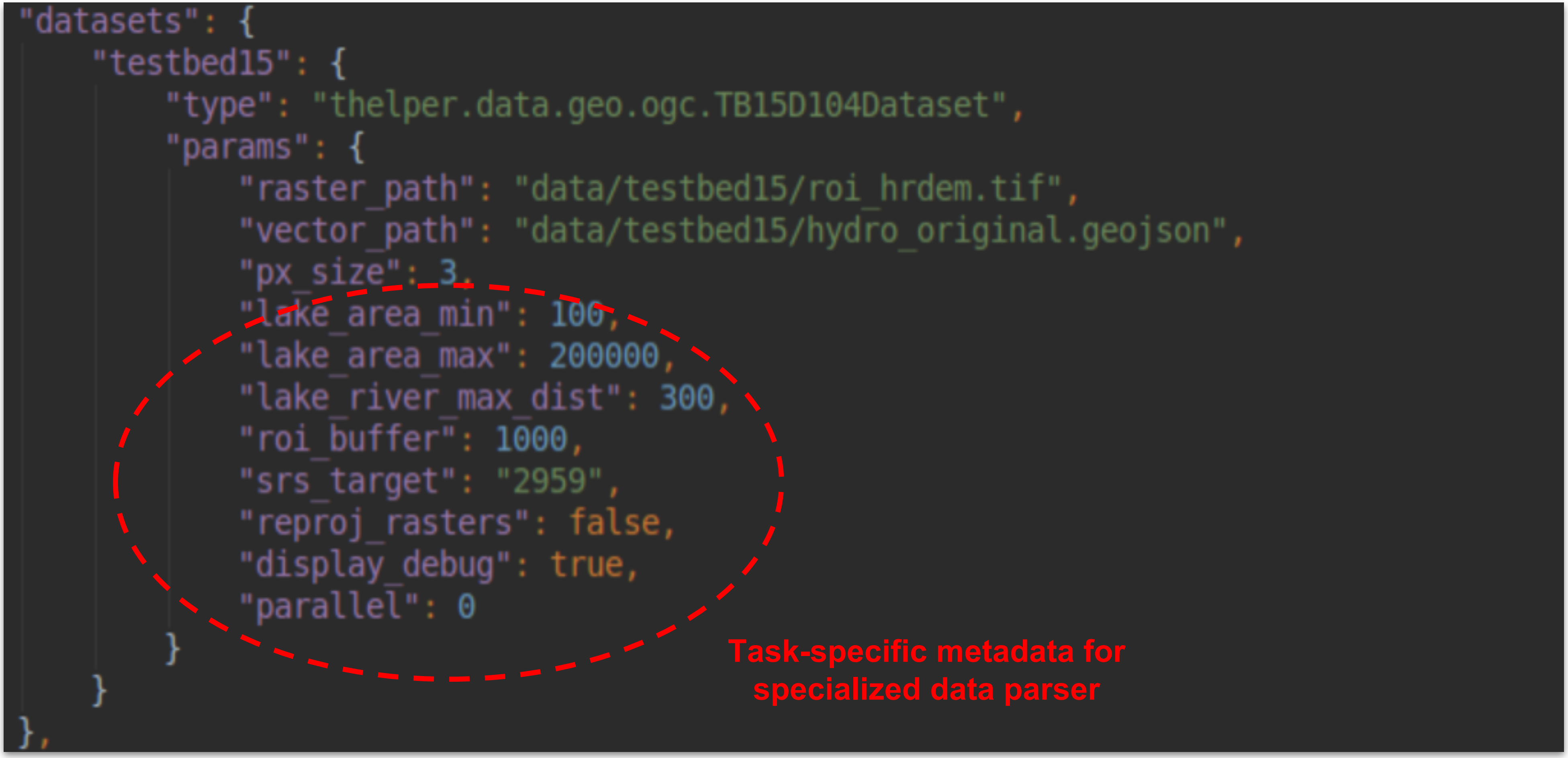


Object Det. Model
(**Faster R-CNN**)

- **Post-processing:**
  - **Reproject (predicted) pixel bounding boxes to geo system**
  - **Merge bounding boxes across overlapping tiles**
  - **“Cut out” lakes from original (pre-raster) waterbodies**

Model training - configure data parser

Setup data parser



```
"datasets": {
    "testbed15": {
        "type": "thelper.data.geo.ogc.TB15D104Dataset",
        "params": {
            "raster_path": "data/testbed15/roi_hrdem.tif",
            "vector_path": "data/testbed15/hydro_original.geojson",
            "px_size": 3,
            "lake_area_min": 100,
            "lake_area_max": 200000,
            "lake_river_max_dist": 300,
            "roi_buffer": 1000,
            "srs_target": "2959",
            "reproj_rasters": false,
            "display_debug": true,
            "parallel": 0
        }
    }
},
```

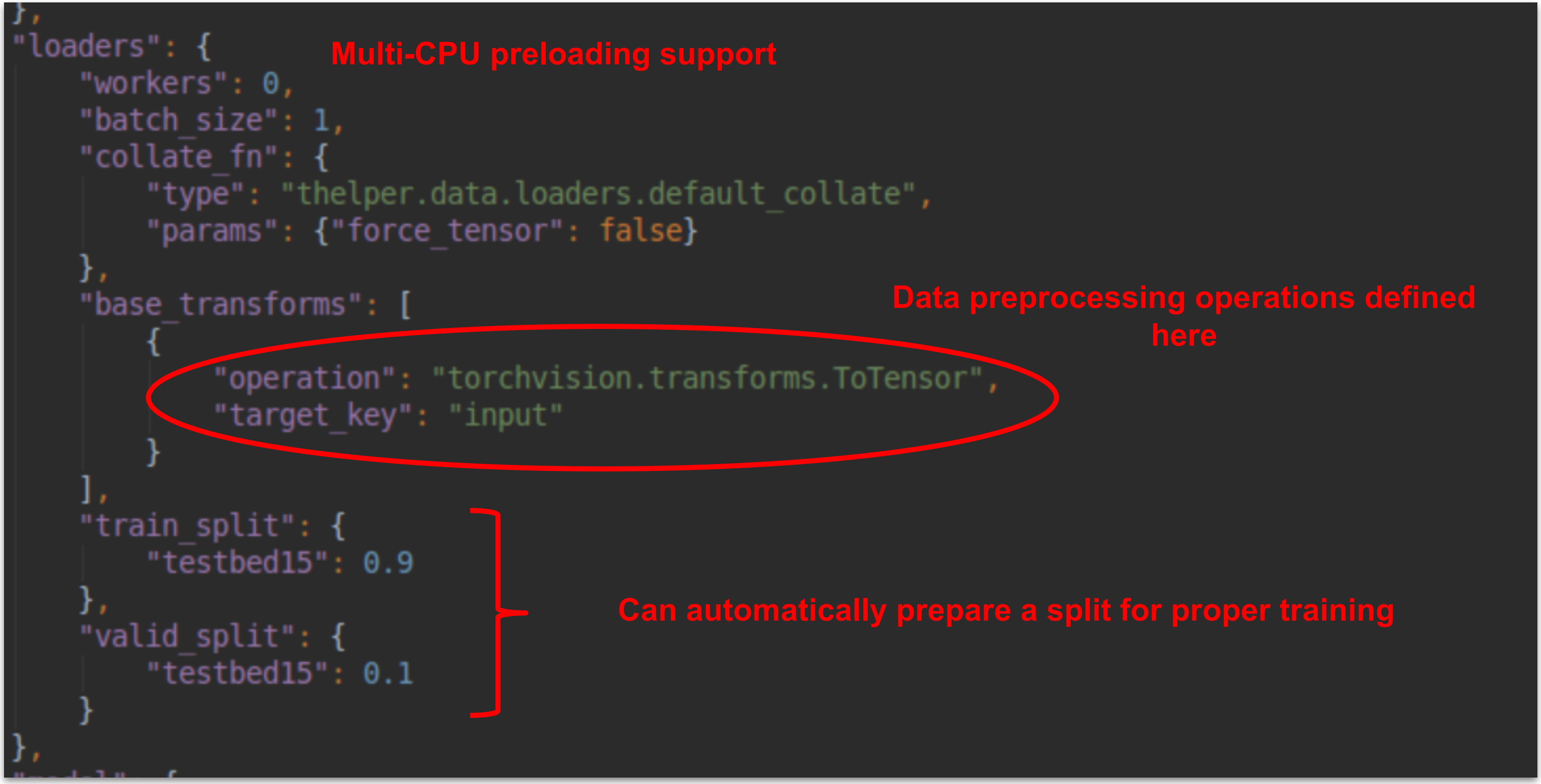
Task-specific metadata for specialized data parser

OGC®

CRIM

Model training - configure data loader

Setup data loaders



```
},
"loaders": {
    "workers": 0,
    "batch_size": 1,
    "collate_fn": {
        "type": "thelper.data.loaders.default_collate",
        "params": {"force_tensor": false}
    },
    "base_transforms": [
        {
            "operation": "torchvision.transforms.ToTensor",
            "target_key": "input"
        }
    ],
    "train_split": {
        "testbed15": 0.9
    },
    "valid_split": {
        "testbed15": 0.1
    }
},
"model": {
```

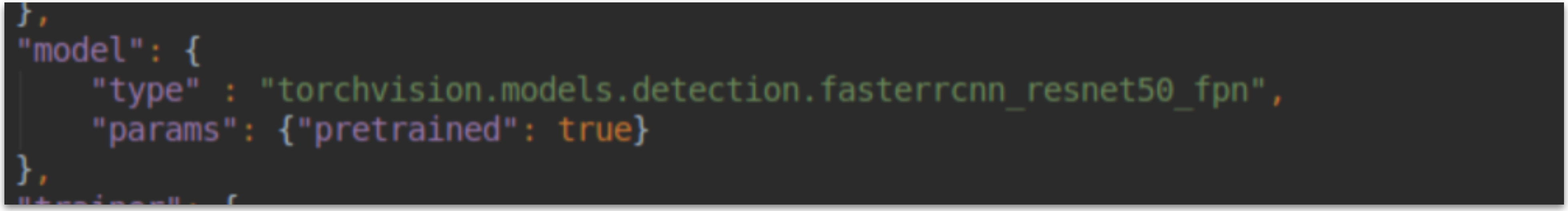
Multi-CPU preloading support

Data preprocessing operations defined here

Can automatically prepare a split for proper training

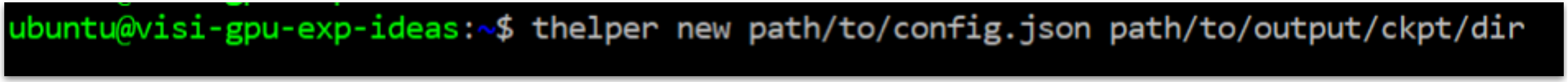
Model training - configure model

Setup model



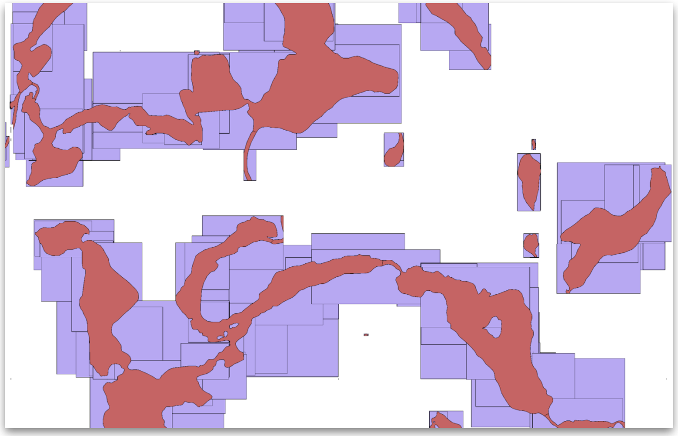
```
},
"model": {
    "type" : "torchvision.models.detection.fasterrcnn_resnet50_fpn",
    "params": {"pretrained": true}
},
"trainer": {
```

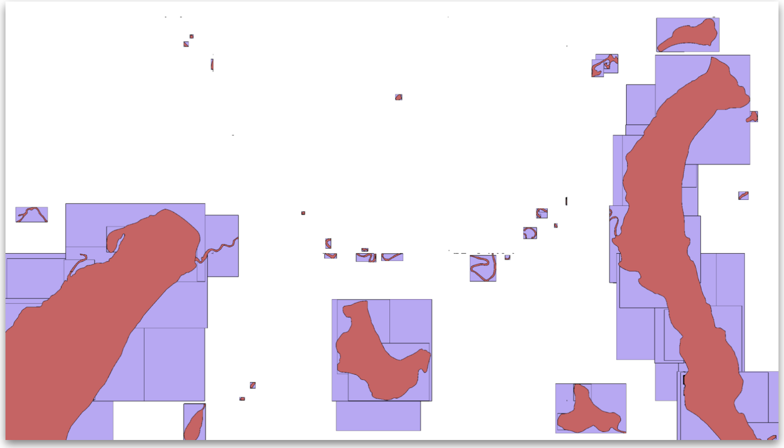
Train!

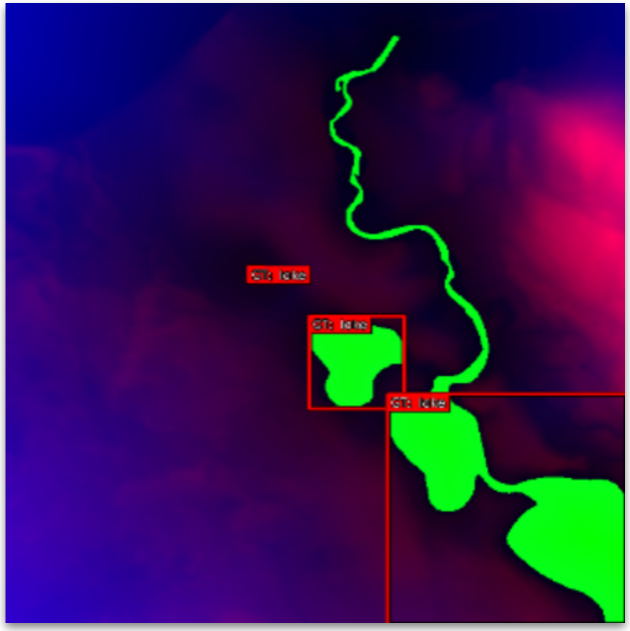


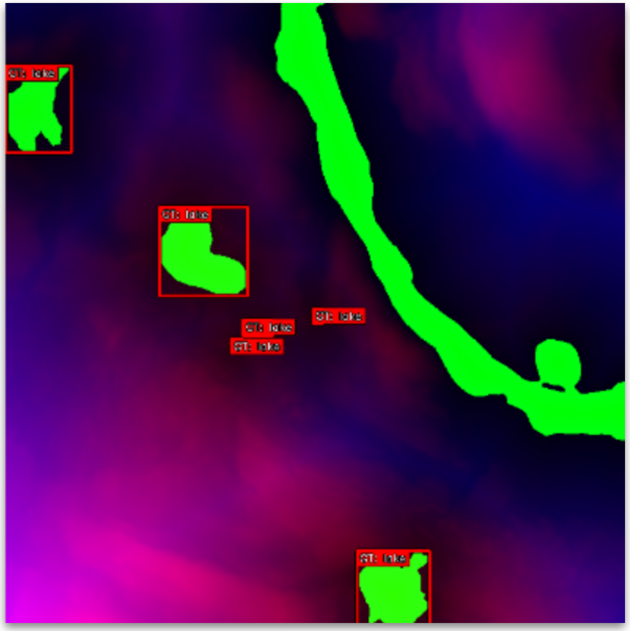
```
ubuntu@visi-gpu-exp-ideas:~$ thelper new path/to/config.json path/to/output/ckpt/dir
```

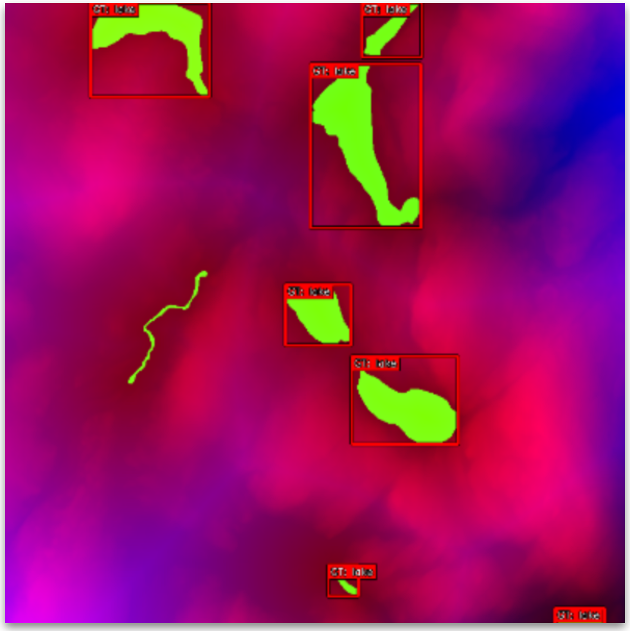
Results for trained model











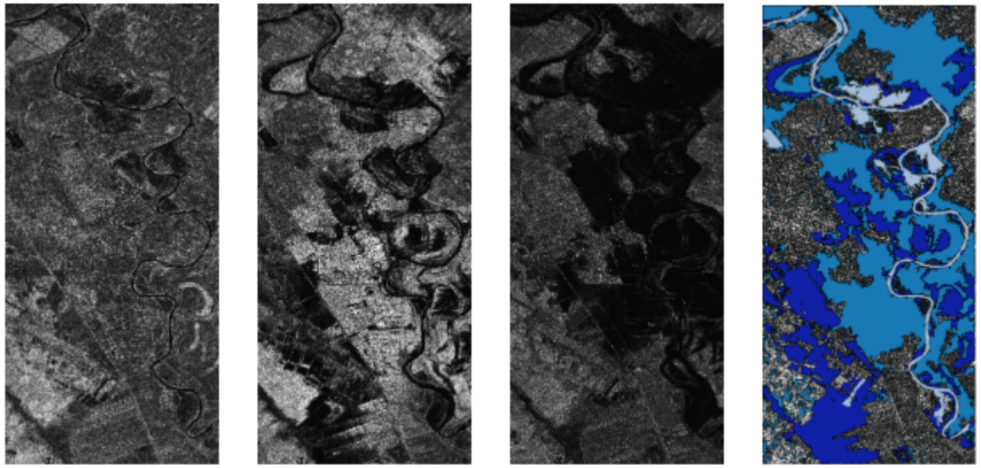
Conclusion

- Lake detection results from trained model are encouraging. Further performance increase to be expected from:
  - Better dataset sampling and splitting
  - Different base model architectures
- Interactive model training (a.k.a experimenting) through WPS is challenging:
  - Partially trained model acceptable by user at any time
  - Real-time logs better used in Tensorboard
- Applicable to climate projections
- Deployment of ML apps is functional

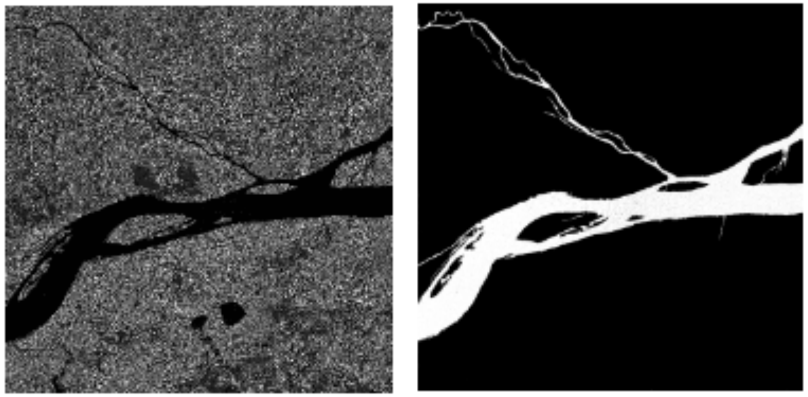
EXTRA SLIDES

Geospatial Machine Learning at CRIM

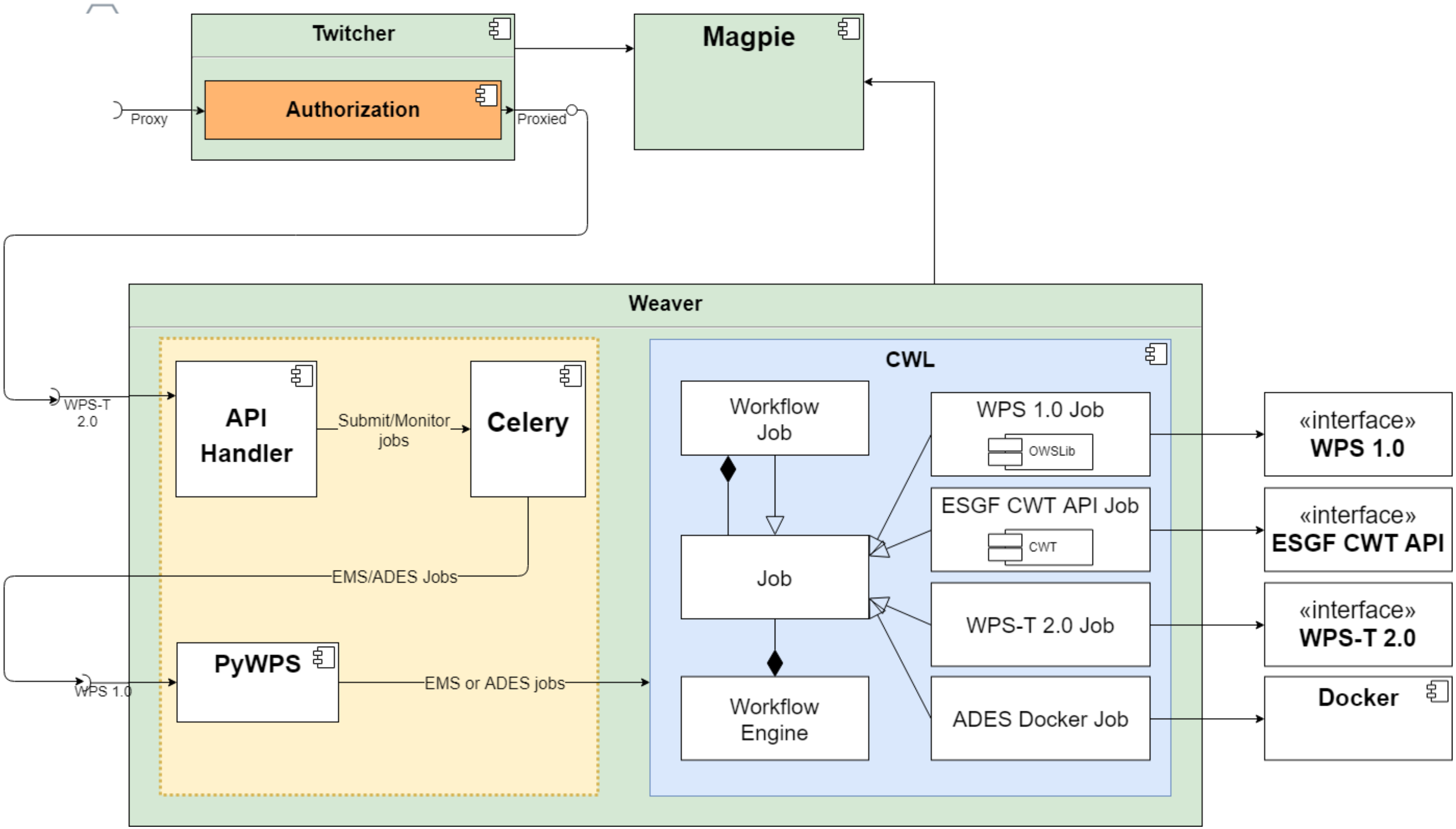
- Past projects in water detection (shallow learning approaches):
  - Active contour applied to SAR and Multispectral (MS) imagery
  - Update and change detection of waterbodies in CanVec (RNCan):
    - http://cangeo.crim.ca
  - River detection based on the Max-Tree filtering technique
  - Multimodal flood mapping
- Ongoing Projects (Deep Learning oriented):
  - GeoImageNet project (Land cover mapping based on VHR images)
  - MUSE Project (Land cover mapping using deep learning techniques)
  - DACCS-EO Data Cube
  - Geo-Deep-Learning framework (collaboration with NRCan CCMEO)



(i) Pre-disaster image (site 3) (j) Post-disaster image t_1(site3) (k) Post-disaster image t_2(site3) (l) Obtained result



Execution Management System - Weaver implementation



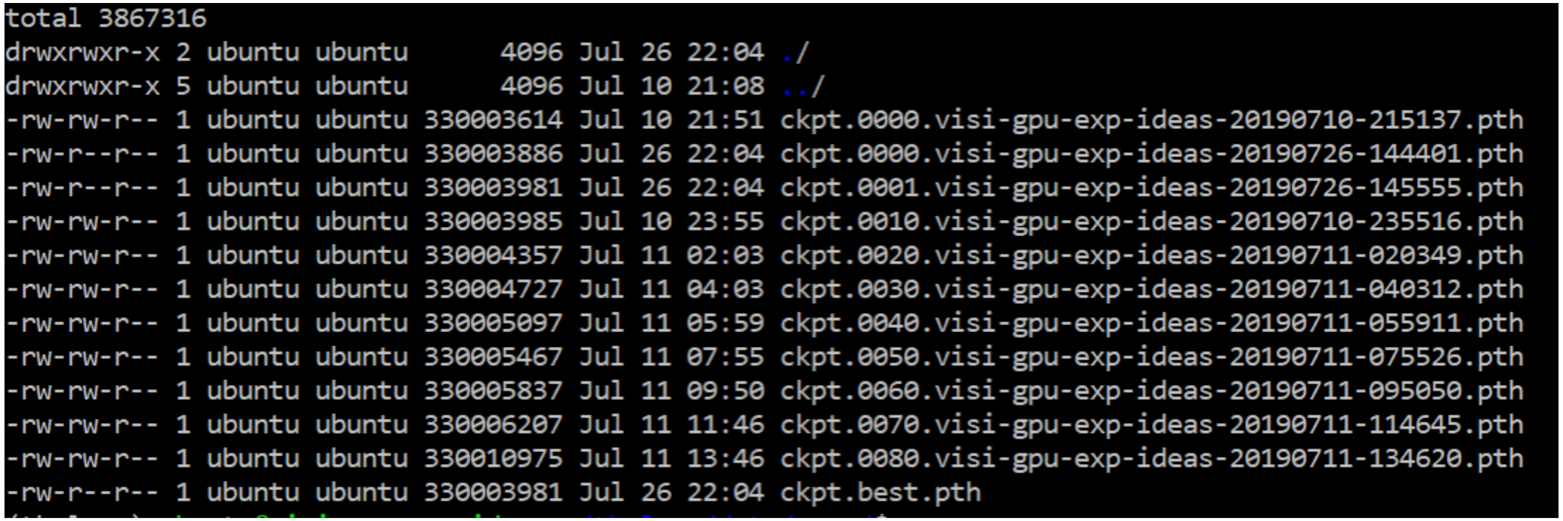
ML app package deployment

- **Deployment**
  [POST https://ogc-ems.crim.ca/weaver/processes]
  - CWL package pointing to Docker image with **thelper**
    toy-example-application.cwl
  - WPS REST application deployed with the CWL package
    toy-example-deploy.json

- **Execution**
  [POST https://ogc-ems.crim.ca/weaver/processes/toy-example/jobs]
  - The **job** pulls the Docker image defined in CWL from the registry and run it with converted WPS → CWL inputs (ie: *model* & *config*)
  - Once the appropriate output GeoJSON file is generated, the process execution will serve it as a job result.

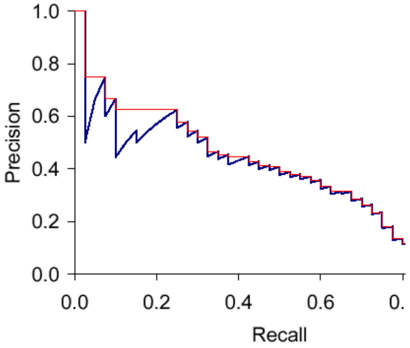
Other training outputs and metadata

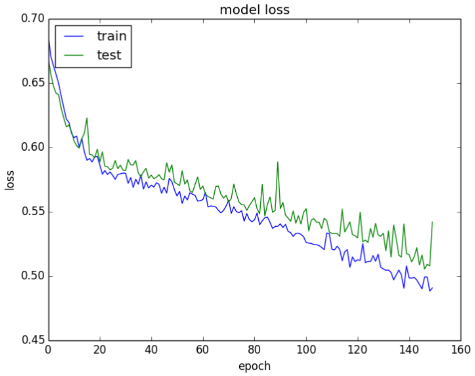
- **Notable training outputs:**
  - **Checkpoints (to continue training or generate predictions)**



```
total 3867316
drwxrwxr-x 2 ubuntu ubuntu      4096 Jul 26 22:04 ./
drwxrwxr-x 5 ubuntu ubuntu      4096 Jul 10 21:08 ../
-rw-rw-r-- 1 ubuntu ubuntu 330003614 Jul 10 21:51 ckpt.0000.visi-gpu-exp-ideas-20190710-215137.pth
-rw-r--r-- 1 ubuntu ubuntu 330003886 Jul 26 22:04 ckpt.0000.visi-gpu-exp-ideas-20190726-144401.pth
-rw-r--r-- 1 ubuntu ubuntu 330003981 Jul 26 22:04 ckpt.0001.visi-gpu-exp-ideas-20190726-145555.pth
-rw-rw-r-- 1 ubuntu ubuntu 330003985 Jul 10 23:55 ckpt.0010.visi-gpu-exp-ideas-20190710-235516.pth
-rw-rw-r-- 1 ubuntu ubuntu 330004357 Jul 11 02:03 ckpt.0020.visi-gpu-exp-ideas-20190711-020349.pth
-rw-rw-r-- 1 ubuntu ubuntu 330004727 Jul 11 04:03 ckpt.0030.visi-gpu-exp-ideas-20190711-040312.pth
-rw-rw-r-- 1 ubuntu ubuntu 330005097 Jul 11 05:59 ckpt.0040.visi-gpu-exp-ideas-20190711-055911.pth
-rw-rw-r-- 1 ubuntu ubuntu 330005467 Jul 11 07:55 ckpt.0050.visi-gpu-exp-ideas-20190711-075526.pth
-rw-rw-r-- 1 ubuntu ubuntu 330005837 Jul 11 09:50 ckpt.0060.visi-gpu-exp-ideas-20190711-095050.pth
-rw-rw-r-- 1 ubuntu ubuntu 330006207 Jul 11 11:46 ckpt.0070.visi-gpu-exp-ideas-20190711-114645.pth
-rw-rw-r-- 1 ubuntu ubuntu 330010975 Jul 11 13:46 ckpt.0080.visi-gpu-exp-ideas-20190711-134620.pth
-rw-r--r-- 1 ubuntu ubuntu 330003981 Jul 26 22:04 ckpt.best.pth
```

  - **Evaluation results (based on the preconfigured metrics)**


Precision
Recall


model loss
train
test
loss
epoch