# Geospatial data processing for image automatic analysis

FOSS4G2019 - Bucharest

*Raphaël Delhome*

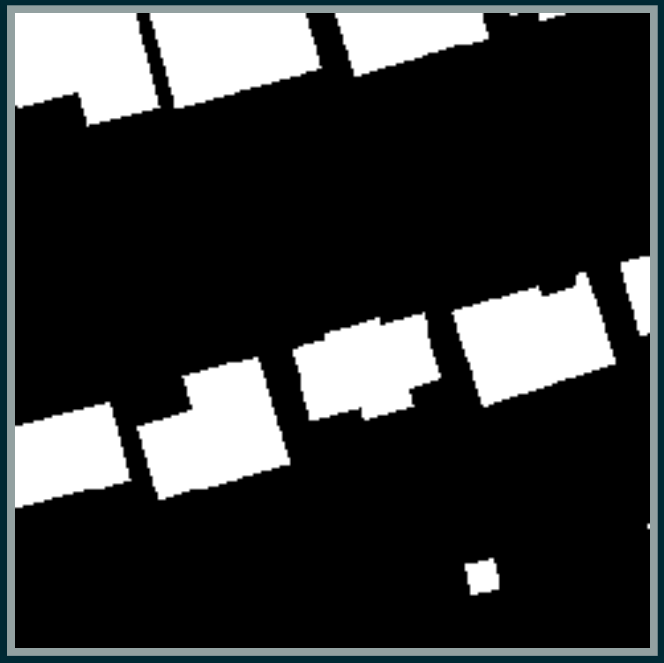
# Introduction

# Oslandia and I


- Data/R&D engineer at Oslandia for 2 years
- Working on geospatial data solutions
- Design data pipelines and AI-related algorithms

# Context

- Artificial Intelligence at Oslandia
- Aerial image democratization
- A historic use case: building footprint detection



# Deep learning and geospatial data

# Image analysis use cases at Oslandia

Tech stack: Linux, Python (Keras, Pillow, ...)

*Semantic segmentation*

- Street-scene images
- Aerial images
- OpenStreetMap data parsing

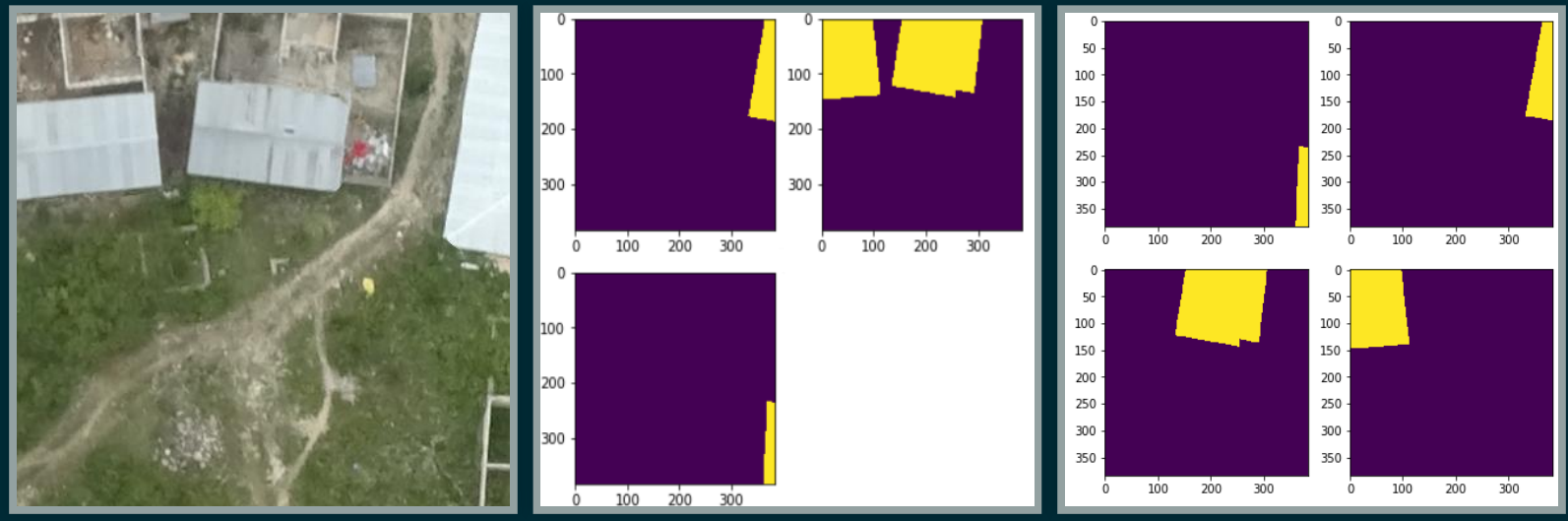
(github.com/Oslandia/deeposlandia)

*Instance segmentation*

- Aerial images

# Deep learning methods

- Image (RGB, 8/16-bandwidth) instead of 3D geometries
- Semantic segmentation (object classes)
- Instance segmentation (individual objects)



# Tanzania dataset

- Released during last FOSS4G conference (see Challenge), buildings in Tanzania
- Building instance detection and status discrimination (completed, unfinished, foundation)
- 13 images (from 17k x 42k to 51k x 51k pixels)

# AerialImage (INRIA)

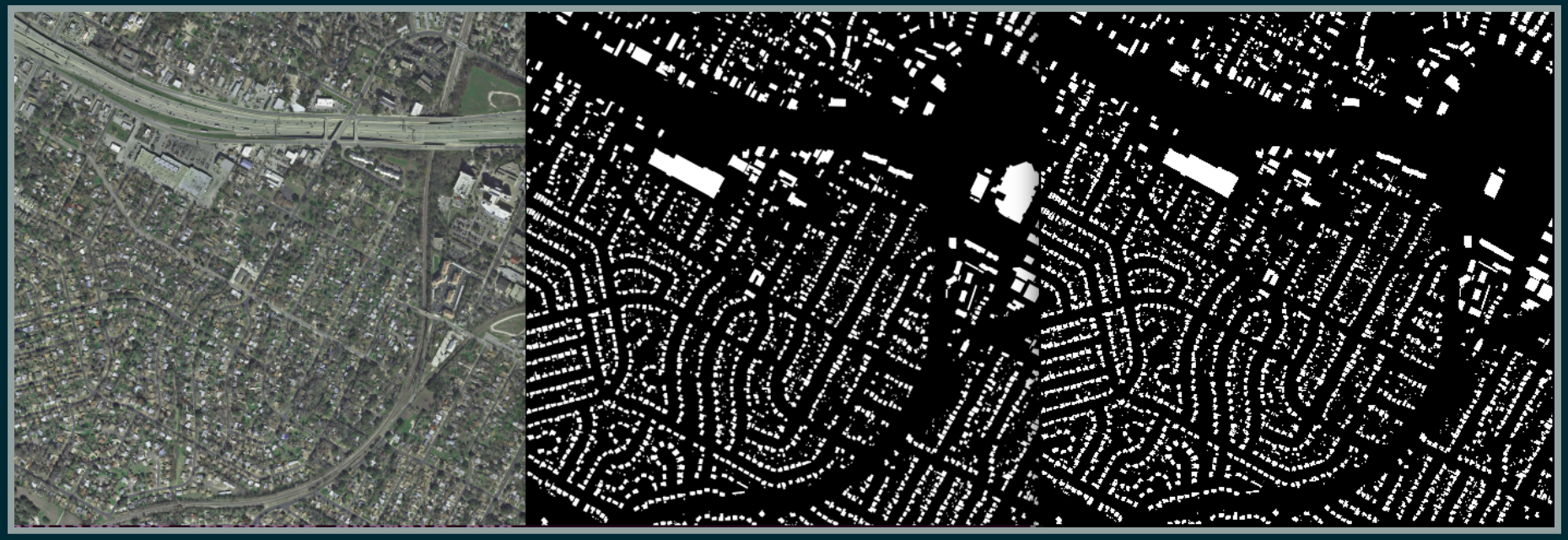
- Georeferenced *.tiff* images (5000 * 5000 pixels)
- 360 images (10 cities of 36 tiles each)
- 50% training, 50% testing

# Link with OSM data

- Rebuild labelled images starting from OSM database
- OSM data as Ground-truth *OR* additional input data
- Process:
  - Extract coordinates (GDAL)
  - Query OSM data (Overpass)
  - Store the data (osm2pgsql)
  - Generate raster tile (Mapnik)

(github.com/Oslandia/osm-deep-labels)

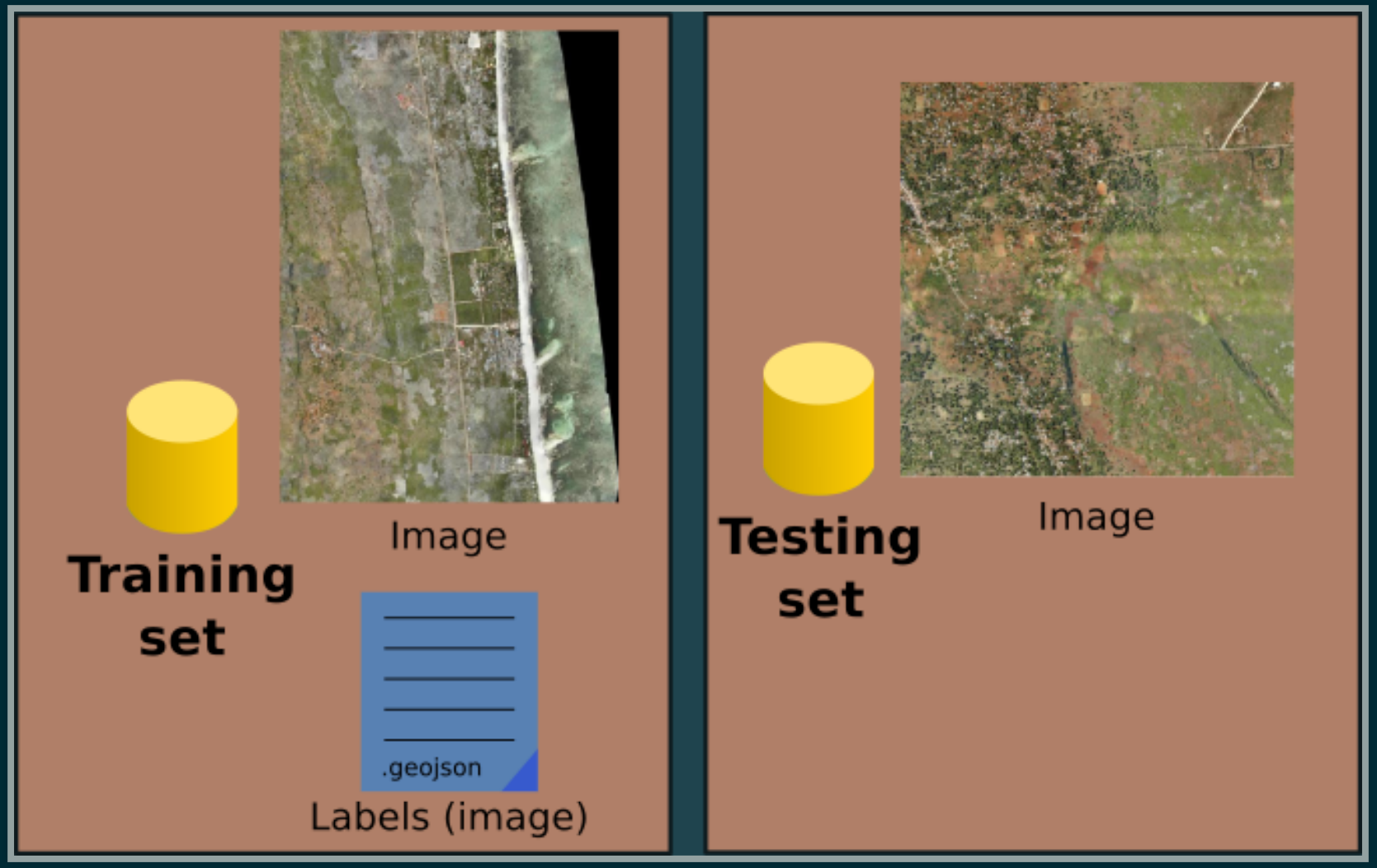
# Link with OSM data



- Left : raw image
- Center : ground-truth label
- Right : OSM raster

# Steps of a deep learning project

# Data parsing



# Data preprocessing

Transform raw images into exploitable tiles:

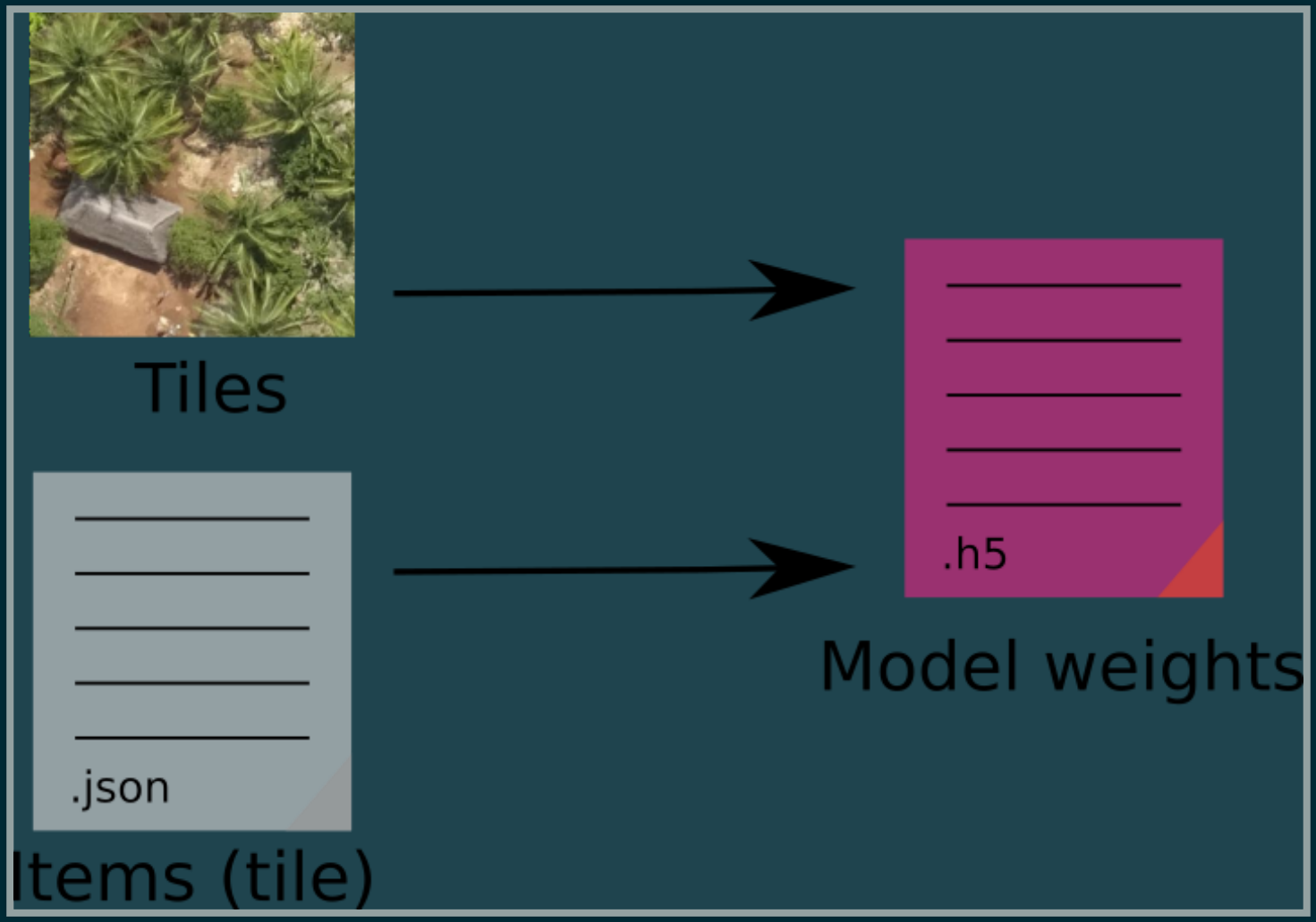
- GDAL

```
gdal_translate -srcwin <min-x> <min-y> <tile-width> <tile-h
  <input-path> <output-path>
```

- numpy

```
tile_data = image_data[
  x:(x + tile_size), y:(y + tile_size)
]
```

# Model training

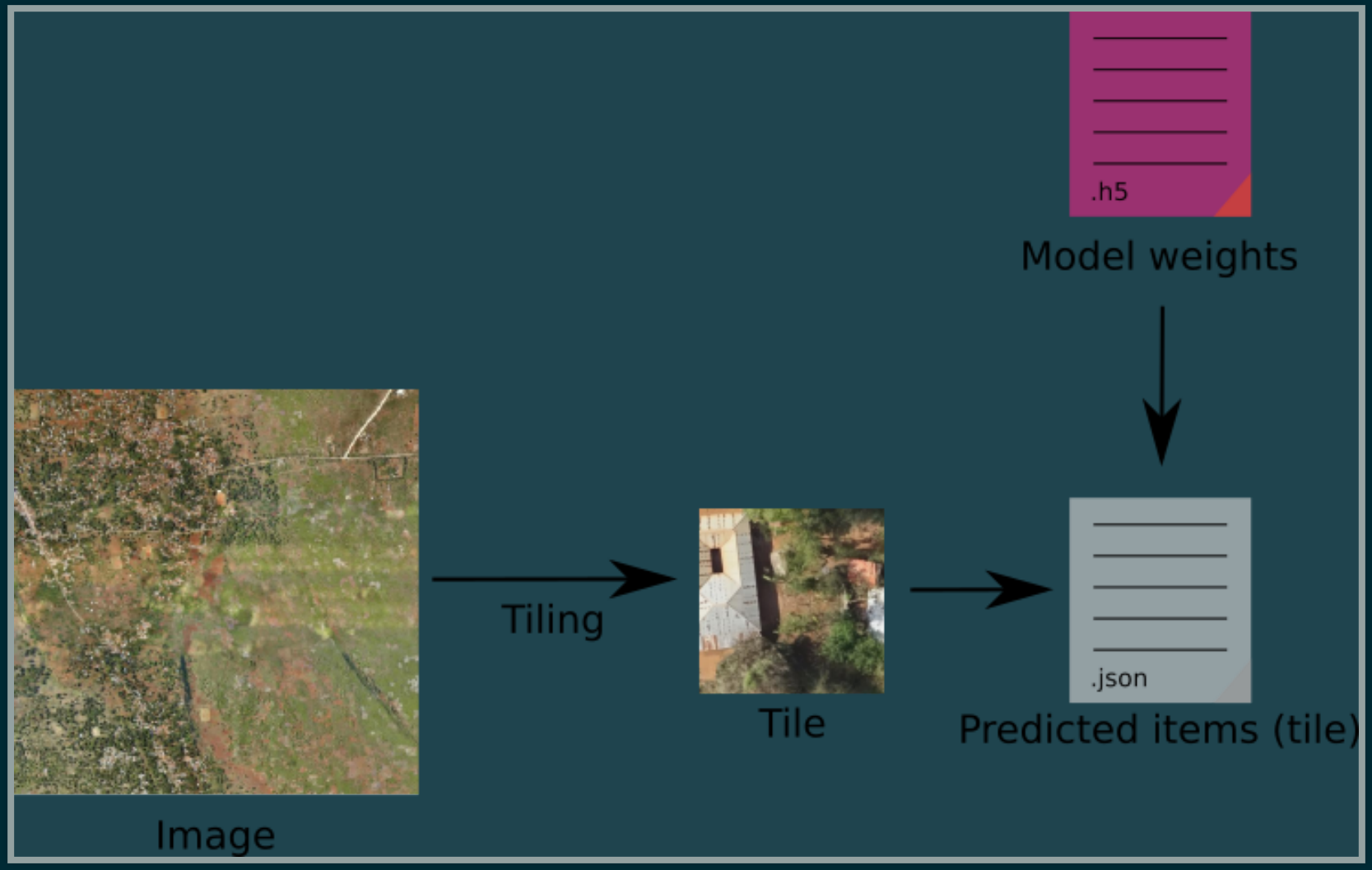


# Model training

Deep learning main challenge:

- Hyperparameter settings: number of training epochs? Learning rate?
- Hardware criticity: 1 GTX 1070Ti GPU

# Model inference



# Model inference

- Generate tiles on test images (*cf* preprocessing)
- Use `keras` API:

```
net = SemanticSegmentationNetwork()  # Create the architecture
model = Model(net.X, net.Y)  # Instanciate the NN model
model.load_weights(<weights_path>)  # Load trained weights
y_raw_pred = model.predict(<images>)  # Predict class scores
predicted_labels = np.argmax(y_raw_preds, axis=3)  # Get pixel cl
```

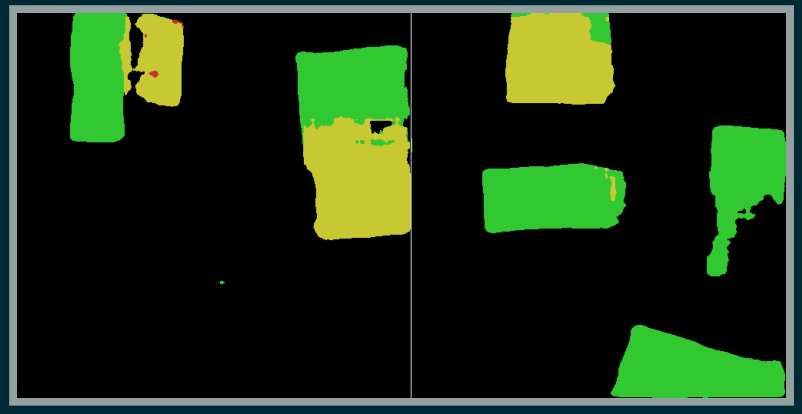
*Output: tables of class IDs for each input image*

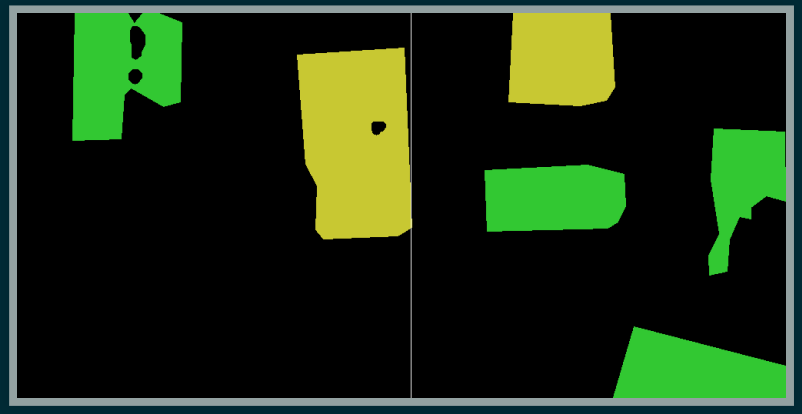
# Postprocessing

- Post-process detection output and produce georeferenced objects
  - Detect polygon contours within boolean masks: OpenCV
  - Transform pixels into geographical coordinates
  - Build polygons with geojson and shapely

```
geom = geojson.Polygon(<list-of-points>)
polygon = shapely.geometry.shape(geom)
```

# Result visualization





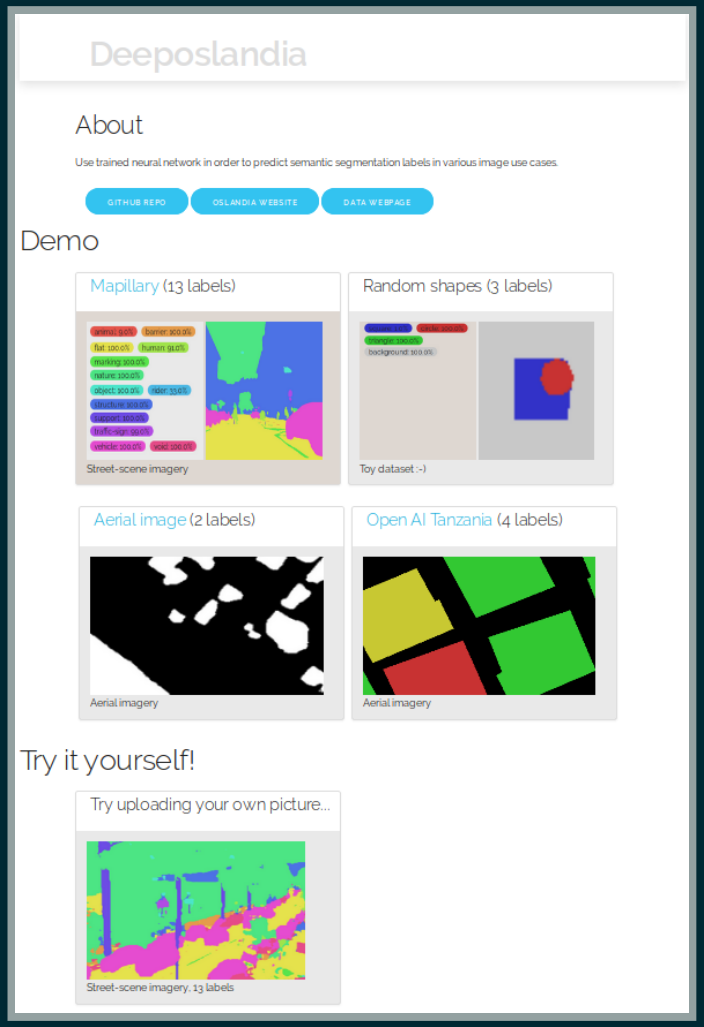
- Predict labels on a bigger image
- Transform them as geometries

# Conclusion and perspectives

# Conclusion and perspectives

- Geospatial data pipeline Proof of Concept: `preprocessing` -> `training` -> `inference` -> `postprocessing`
  github.com/Oslandia/tanzania-challenge
- Modest results for now... Need for spending time on:
  - data preprocessing
  - model training!
- Towards a QGIS plugin!

# Bonus track: web app demo



# Thank you for your attention!

Find out more:

- https://oslandia.com/en/blog/
- github.com/Oslandia/deeposlandia
- github.com/Oslandia/osm-deep-labels
- github.com/Oslandia/tanzania-challenge
- http://data.oslandia.io