Technische Sessie:
Aan de slag met de Tygron AI Suite

Frank Baars - Senior Software Developer bij Tygron

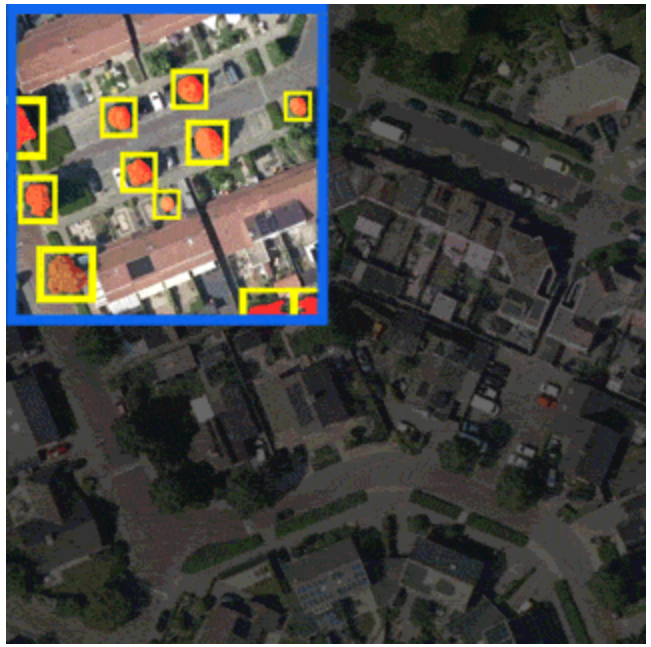
TYGRON PLATFORM
Accelerated Simulations for Engineers

Technische Sessie

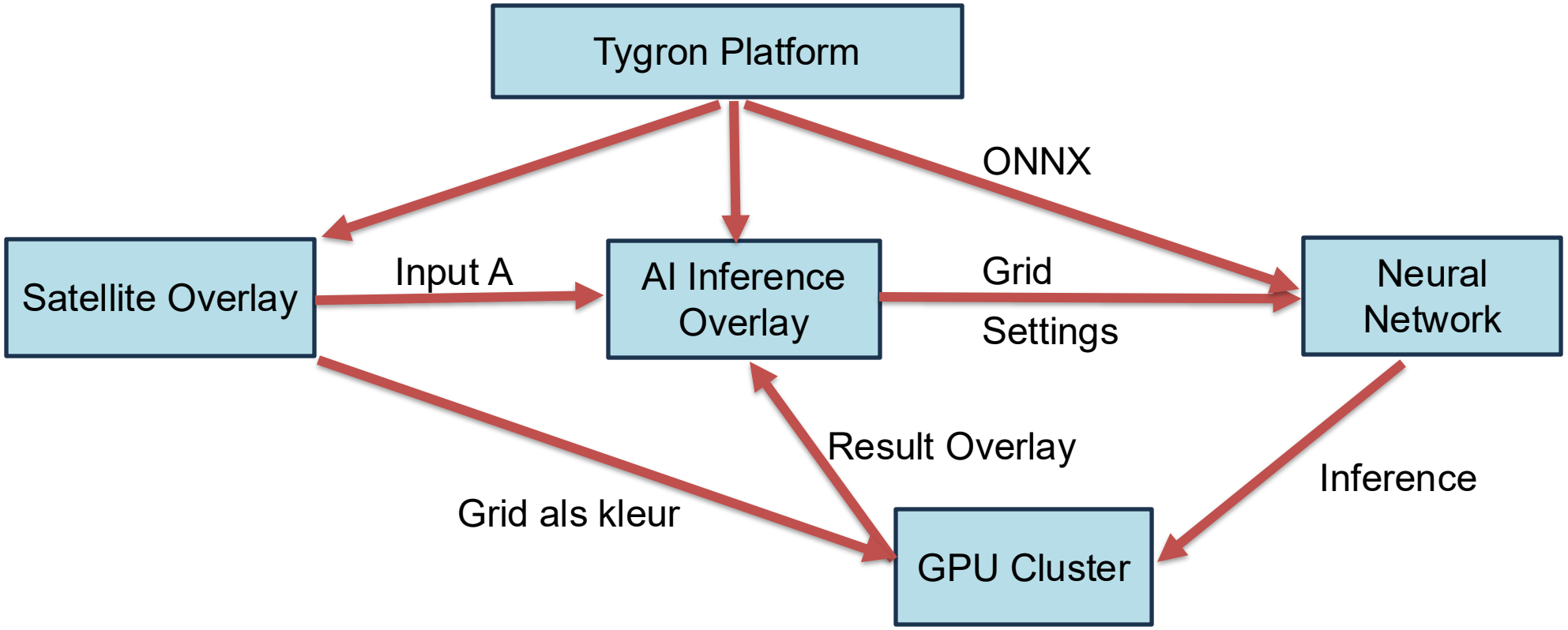
Onderwerp:
- Tygron AI Suite

Doel:
- Exporteer Train en Test Data.
- Doorlopen stappen voor Trainen AI Model.

Hoe:
- Exporteer AI Train Data.
- Richt Python AI Train omgeving in.
- Configureer AI Model in Jupyter Notebook.
- Train AI Model met PyTorch.
- Exporteer als ONNX en importeer in project.



Tygron Platform



Stap 1: Ingetekende AI Dataset

- Project met ingetekende **Areas**

  Demo Training Data

- **Open dit project nu!**

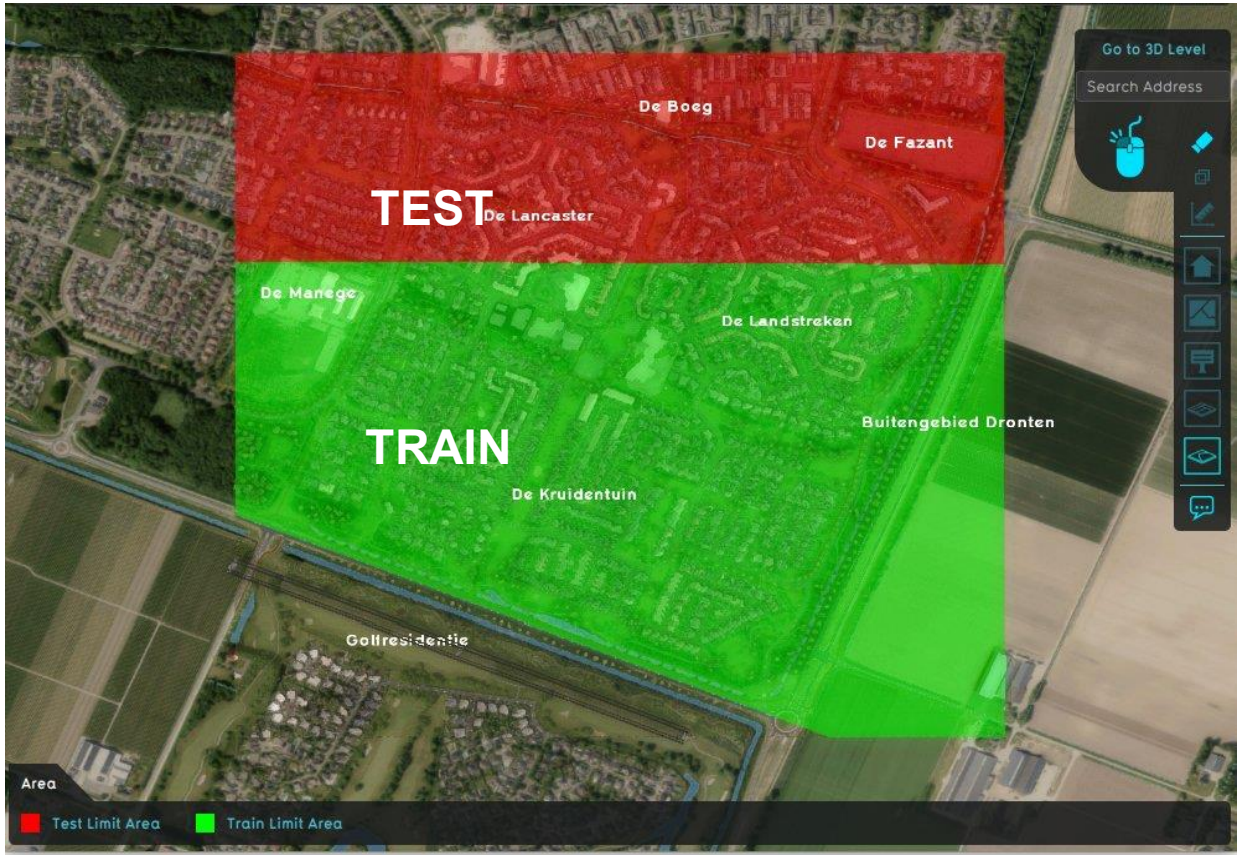
- Ingetekend bovenop **Satellite Overlay**

- Ingetekend met QGIS:

  https://previewsupport.tygron.com/wiki/How_to_create_AI_train_data_with_QGIS

Stap 2: Opdeling AI Dataset

- Opdeling dataset in 2 losstaande areas: **TRAIN** en **TEST**
- TRAIN set groter dan TEST set



Stap 3: Exporteer AI Training Dataset

Exporteer Train Dataset:

- Current Situation > Areas > Export Geo Data
- Filter: **FOLIAGE**
- Format: **AI Training Data PNG**
- Intersecting Areas: **TRAIN_LIMIT_AREA**
- Export Option > Overlay: **Original Satellite**
- Export Files: AI MAP > Data > Train > Project Naam

Stap 4: Exporteer AI Test Dataset

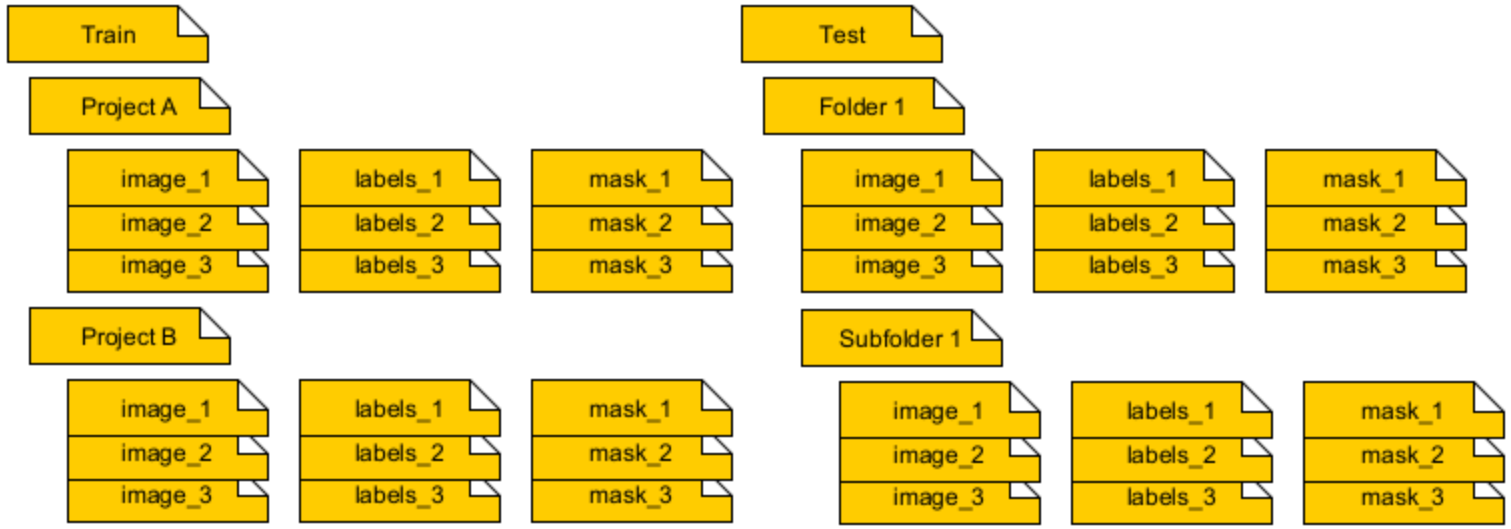
Exporteer **Test** Dataset:

- Current Situation > Areas > Export Geo Data
- Filter: FOLIAGE
- Format: AI Training Data PNG
- Intersecting Areas: **TEST_LIMIT_AREA**
- Export Option > Overlay: Original Satellite
- Export Files: AI MAP > Data > **Test** > Project Naam

Stap 5: Inspecteer Datasets

Inspecteer Datasets

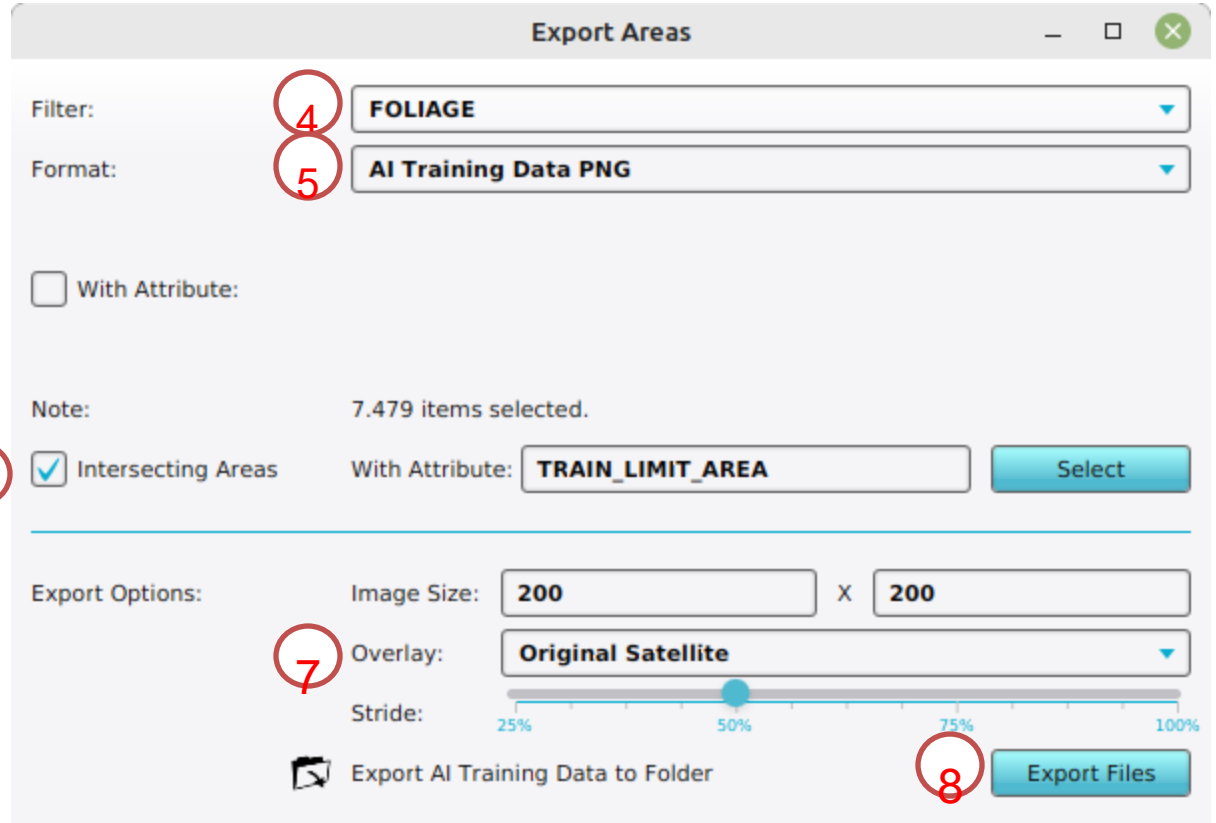
- Terecht gekomen in losse mappen?
- Meerdere sub-folders toegestaan.



Handout

1. Open Project: Demo Training Data
2. Selecteer Current Situation
3. Hover over Areas en selecteer Export Geo Data
4. Selecteer Filter: Foliage
5. Selecteer Format: AI Training Data PNG
6. Selecteer Intersecting Areas With Attribute: **TRAIN_LIMIT_AREA**
7. Export Option Overlay: Original Satellite
8. Click op Export Files en sla op onder pad: ai/data/**train**

9. Herhaal voor test data:
    1. Intersecting Areas : **TEST_LIMIT_AREA**
    2. Test data pad: ai/data/**test**



Model Trainen: Tygron AI Suite

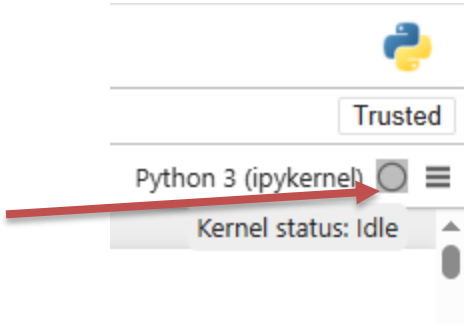
- Tygron AI Suite op Github https://github.com/Tygron/tygron-ai-suite
- Download zip of installeer git for windows: https://gitforwindows.org
- CMD (Command line tool windows):
  cd ai
  git clone https://github.com/Tygron/tygron-ai-suite.git

Model Trainen: Python Environment

- Download Conda-forge's Miniforge: https://conda-forge.org/download/
  Python Environment Manager
- Open Miniforge Terminal
- Change directory naar map tygron ai suite:
  cd ai/tygron_ai_suite
- Create Environment, download en installeer python packages:
  conda env create -n tygronai -f tygronai.yml
- Activeer tygronai enviroment:
  conda activate tygronai
- Open Jupyter notebook:
  jupyter notebook

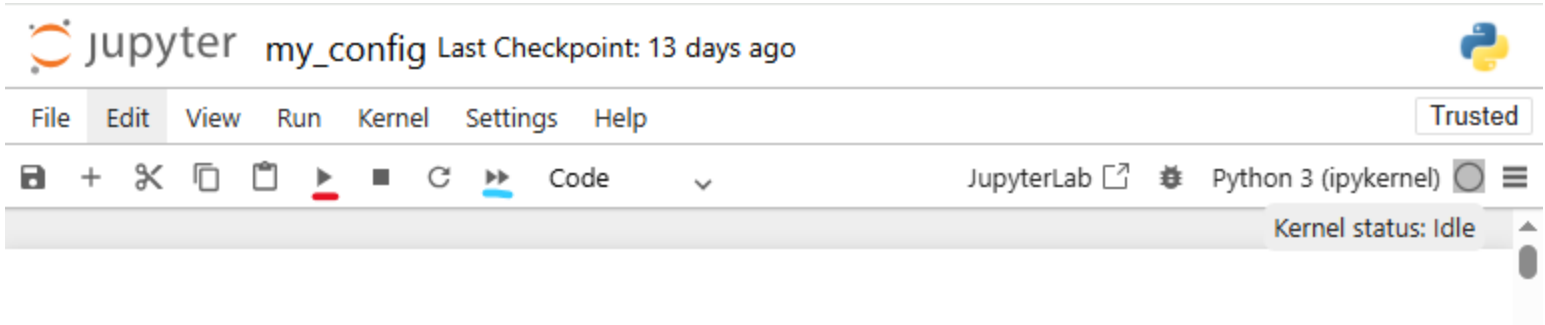
Model Trainen: Jupyter Notebook

- Dupliceer example_config.ipynb > my_config.ipynb
- Doe aanpassingen aan:
  - Configuratie
  - Train Dataset en Test Dataset locaties
  - Doorloop stappen: Stuk voor stuk of alles in 1 keer
  - Valideer
    device: cuda
    kernel blijft draaien



Model Trainen: Controls en Kernel

- Kernel voert het Notebook uit
- Open example_config.ipynb
- Rood: voer 1 stap uit
- Blauw: reset en voer alle stappen uit



Model Trainen: Libraries

- Inference training library door Tygron gemaakt
- Bevat verzameling methodes en variabelen
- Roept onderliggend PyTorch aan; PyTorch Model, PyTorch Training etc.

Imports

First we will import all functions that we will use in this notebook.

[1]:
```
from inference_training import Configuration, ImageDataset
from inference_training import initCudaEnvironment, createTransforms
from inference_training import drawImageAndFeatureMasks
from inference_training import exportOnnxModel, writeONNXMeta, loadONNX
from inference_training import trainModel, saveModel, loadModel
from inference_training import createModelInstance, testInference
from inference_training import logger
```

Model Trainen: Hardware en Configuratie

- Hardware en Model configuratie

Training environment

With this function we will initialize the cude environment, if present on our workstation.

In case multiple gpu's (cuda devices) are present on your workstations, you can adjust the

[2]:
```
initCudaEnvironment(numCudaDevices=1,
                    visibleCudaDevices="0",
                    clearCudaDeviceCount=False)
```

Configuration

Here we initialize a configuration instance, used by the datasets and the creation and train

[3]:
```
# train on the GPU or on the CPU, if a GPU is not available
config = Configuration()
print("Using Device: " + str(config.device))
```

```
Using Device: cuda
```

Model Trainen: Configuratie Parameters

- Dataset locatie
- Model naam, input grootte
- Autolimit: Automatische aanpassing invalide labels (classificatie nummer)

```
import os

trainDirectory = "../data/foliage/train/"
testDirectory = "../data/foliage/test/"

assert os.path.isdir(trainDirectory), f'Train directory does not exists! {trainDirectory}';
assert os.path.isdir(testDirectory), f'Test directory does not exists! {testDirectory}';

config.setDatasetPaths(trainPath=trainDirectory, testPath=testDirectory)
config.setFilePrefix("")
config.setModelName("my_model")
config.setInputSizes(inputWidth=200, inputHeight=200)
config.setInputCellSize(cellSizeM=0.25, minCellSizeM=0.25, maxCellSizeM=0.25)
config.setAutoLimitLabel(True)

print("Model name: " + str(config.getModelName()))
print("Version: " + str(config.version))
```

```
Model name: my_model
Version: 20251021
```

Model Trainen: Training Configuratie

- Hoeveelheid klassen (achtergrond met waarde 0 is altijd extra klasse)
- Aantal input kanalen van beeld (Rood Groen Blauw)
- Overlap en hoeveel features maximaal per plaatje
- Aantal train & test rondes (epochs)
  Balans tussen genoeg trainen en overtraind zijn!

```
numberOfClasses = 1

config.setModelInfo(channels=3, #
                    numClasses=numberOfClasses+1,  # ( +1 background)
                    bboxOverlap=True, #
                    bboxPerImage=250, #
                    reuseModel=False) #
config.setEpochs(10)
```

Model Trainen: Onnx Export Information

- Beschrijving voor neural network
- Legenda voor Inference Overlay
- Threshold attributen voor Inference Overlay
- Type input: A of B, RGB, Genormaliseerd?

```
description = "Model description"
config.setOnnxInfo(producer="Tygron", description=description)

#Legend entries used by the label result type of an Inference Overlay in the Tygron Platform
config.clearLegendEntries()
config.addLegendEntry("Background", 0, "#00000000")
config.addLegendEntry("Foliage", 1, "#00ffbf")

missingLegendEntries = numberOfClasses + 1 - len(config.getLegendEntries())
if missingLegendEntries > 0:
    logger.warning(str(missingLegendEntries) + " Legend Entries are not yet defined! Add more legend e

# Inference Overlay model parameters used when applying this ONNX model.
config.setOnnxMetaData(scoreThreshold=0.2,
                       maskThreshold=0.3,
                       strideFraction=0.5)

#Name of the input tensor and the batch amount set when exporting this model to an ONNX file.
config.setTensorInfo(tensorName='input_A:RGB_normalized', batchAmount=1)
```

Model Trainen: Datasets

- Datasets aanmaken en valideren
- Aanvullende transformaties? Zelf toe te voegen:
  - Beelden afsnijden, schalen, spiegelen
  - Saturatie, scherpte, resolutie
- https://docs.pytorch.org/vision/stable/auto_examples/index.html

```
trainingDataset = ImageDataset(config, True, createTransforms(True))
testDataset = ImageDataset(config, False, createTransforms(False))

logger.info("Train Image count: "+str(trainingDataset.__len__()))
logger.info("Test Image count: "+str(testDataset.__len__()))

trainingDataset.validate()
testDataset.validate()

logger.info("Pytorch model name " + config.getPytorchModelFileName())
logger.info("Onnx file name " + config.getOnnxFileName())
```

Model Trainen: Input valideren

- Voorbeeld bekijken
  - Features aanwezig? Juiste achtergrond?
  - Probeer eens ander nummer, en test Dataset

```
imageNumber = 5
print(trainingDataset.getLabels(imageNumber))
drawImageAndFeatureMasks(config, trainingDataset, imageNumber)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Model Trainen: Trainingsrondes

- Model trainen en achteraf opslaan
- Trainingsduur afhankelijk van:
- Hardware
- Hoeveelheid train en test data
- Aantal rondes (epochs)

```
model = trainModel(config, trainingDataset, testDataset)
saveModel(config, model, path=config.getPytorchModelFileName())
```

```
C:\Users\Tygron\OneDrive\Documenten\ai\tygron-ai-suite\engine.py:30: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=scaler is not None):
Epoch: [0]  [  0/795]  eta: 0:26:29  lr: 0.000011  loss: 1.2128 (1.2128)  loss_classifier: 1.2122 (1.2122)  loss_box_reg: 0.0000 (0.0000)  loss_mask: 0.0000 (0.0000)  loss_objectness: 0.0006 (0.0006)  loss_rpn_box_reg: 0.0000 (0.0000)  time: 2.0000  data: 0.0313  max mem: 1654
Epoch: [0]  [ 10/795]  eta: 0:19:46  lr: 0.000074  loss: 4.0940 (4.2191)  loss_classifier: 0.9974 (1.0062)  loss_box_reg: 0.4796 (0.4518)  loss_mask: 1.2142 (1.1928)  loss_objectness: 1.2713 (1.4461)  loss_rpn_box_reg: 0.1293 (0.1223)  time: 1.5115  data: 0.0184  max mem: 2080
Epoch: [0]  [ 20/795]  eta: 0:19:31  lr: 0.000137  loss: 3.1494 (3.4633)  loss_classifier: 0.7216 (0.7935)  loss_box_reg: 0.4796 (0.5283)  loss_mask: 0.8395 (0.9945)  loss_objectness: 0.9251 (1.0139)  loss_rpn_box_reg: 0.1293 (0.1331)  time: 1.4874  data: 0.0213  max mem: 2152
Epoch: [0]  [ 30/795]  eta: 0:19:02  lr: 0.000200  loss: 2.0815 (2.9369)  loss_classifier: 0.5129 (0.6594)  loss_box_reg: 0.4706 (0.5101)  loss_mask: 0.5922 (0.8267)  loss_objectness: 0.3452 (0.8021)  loss_rpn_box_reg: 0.1038 (0.1386)  time: 1.4835  data: 0.0230  max mem: 2152
Epoch: [0]  [ 40/795]  eta: 0:18:43  lr: 0.000263  loss: 1.9439 (2.6309)  loss_classifier: 0.4020 (0.5933)  loss_box_reg: 0.5062 (0.5122)  loss_mask: 0.4621 (0.7368)  loss_objectness: 0.2311 (0.6563)  loss_rpn_box_reg: 0.1286 (0.1324)  time: 1.4634  data: 0.0220  max mem: 2152
Epoch: [0]  [ 50/795]  eta: 0:18:31  lr: 0.000326  loss: 1.7010 (2.4371)  loss_classifier: 0.4083 (0.5493)  loss_box_reg: 0.6101 (0.5275)  loss_mask: 0.4061 (0.6662)  loss_objectness: 0.1914 (0.5635)  loss_rpn_box_reg: 0.1355 (0.1306)  time: 1.4884  data: 0.0234  max mem: 2153
```

Model Trainen: Eindoverzicht

- Eind overzicht na 10 rondes trainen:

```
Test:  [815/816]  eta: 0:00:00  model_time: 0.3437 (0.3655)  evaluator_time: 0.0312 (0.0325)
Test: Total time: 0:05:34 (0.4098 s / it)
Averaged stats: model_time: 0.3437 (0.3655)  evaluator_time: 0.0312 (0.0325)
Accumulating evaluation results...
DONE (t=0.29s).
Accumulating evaluation results...
DONE (t=0.30s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.250
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.507
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.219
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.197
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.491
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.074
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.035
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.227
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.322
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.274
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.564
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.122
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.248
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.496
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.227
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.195
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.481
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.012
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.035
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.223
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.318
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.276
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.530
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.011
```

Model Trainen: Valideer Model

- Model evaluatie > optimalisatie
- Valideer getraind model:
  - Herkent het iets
  - Hoe goed?
- Let wel:
  - Dit is op gebruikte data
- Daadwerkelijke evaluatie moet op:
  - Losstaande dataset
  - Nieuw project in Tygron Platform

Model Trainen: Exporteren naar ONNX

- Getraind model is PyTorch Model.
- Exporteren naar Open Neural Network Exchange formaat (ONNX)

Export to an ONNX file

In this part we will export the onnx model, set the metadata (for use in the Tygron Platform)

Using the configuration instance, we can simply export the trained model to an ONNX file using this function:

```
exportOnnxModel(config, model)
```

```
C:\Users\Tygron\miniforge3\envs\tygronai\Lib\site-packages\torch\nn\functional.py:4624: UserWarning: To
to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than
  * torch.tensor(scale_factors[i], dtype=torch.float32)
C:\Users\Tygron\miniforge3\envs\tygronai\Lib\site-packages\torchvision\ops\boxes.py:166: UserWarning: To
d to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather tha
```

Model Trainen: ONNX Metadata

- Metadata kan worden weggeschreven na export
- Opgehaald uit configuratie
- Achteraf ook aan te passen:
  - Legenda
  - Beschrijving
  - Producer

```
writeONNXMeta(config)
```

Validating the ONNX file

In this last step, we will try to load the exported ONNX file and see if the meta data is present.

```
onnx_model = loadONNX(config)
print(f"metadata_props={onnx_model.metadata_props}")
```

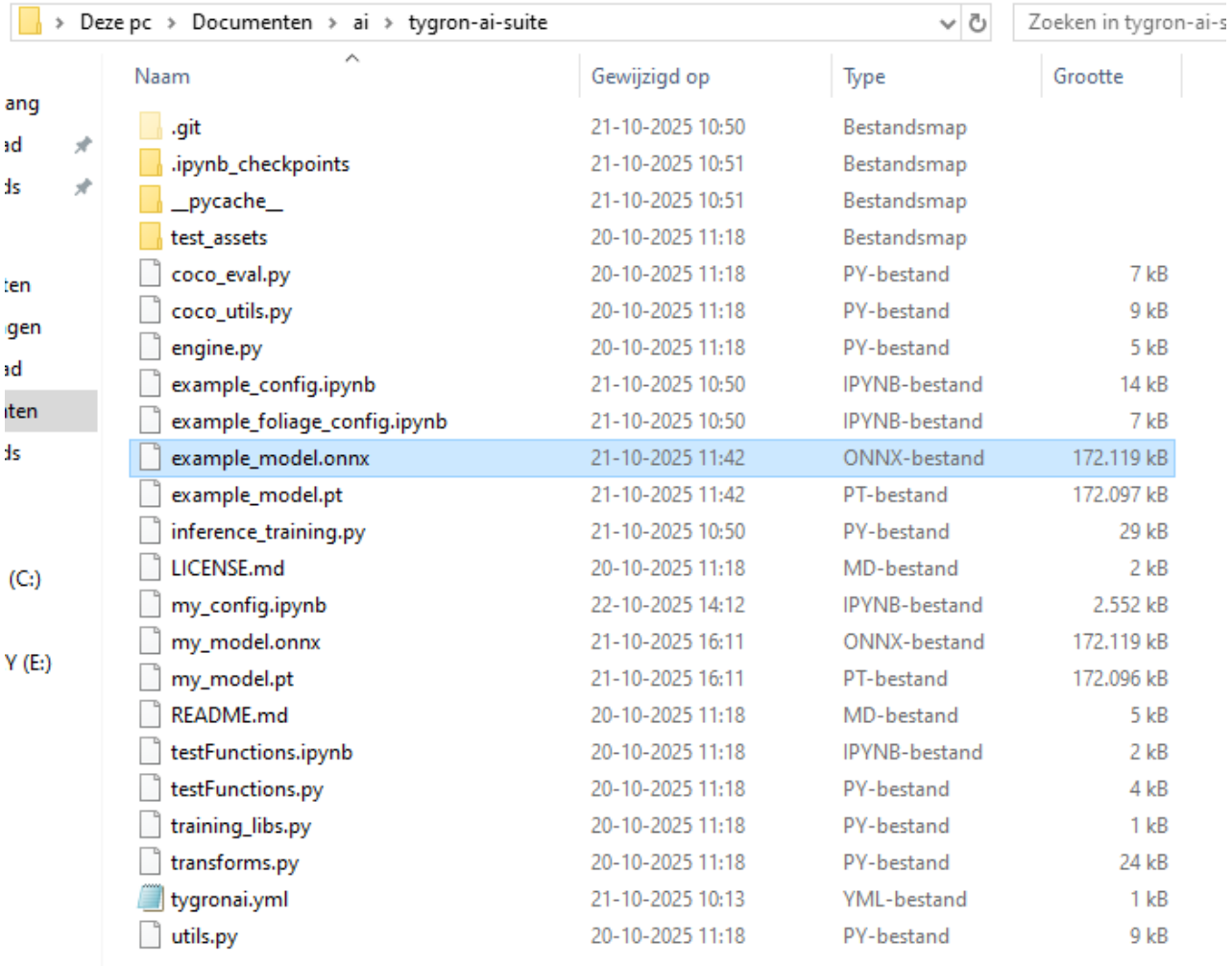
```
metadata_props=[key: "VERSION"
value: "20251021"
, key: "DESCRIPTION"
value: "Model description"
, key: "PRODUCER"
value: "Tygron"
, key: "MIN_CELL_SIZE_M"
value: "0.25"
, key: "MAX_CELL_SIZE_M"
value: "0.25"
, key: "MASK_THRESHOLD"
value: "0.3"
, key: "SCORE_THRESHOLD"
value: "0.2"
, key: "INFERENCE_MODE"
value: "1"
, key: "BOX_OVERLAP"
value: "1"
, key: "STRIDE_FRACTION"
value: "0.5"
, key: "MAX_GT_INSTANCES"
value: "250"
, key: "LEGEND_NAMES"
value: "Background, Foliage"
, key: "LEGEND_VALUES"
value: "0, 1"
, key: "LEGEND_COLORS"
value: "#00000000, #00ffbf"
]
```

Model Trainen: Bestanden Overzicht

- Bestanden overzicht
- Example_model.pt ook te laden

```
model = createModelInstance(config)
loadModel(config, model)
```

- Verder trainen dus ook mogelijk



Model Importeren

- Open een project in Tygron Platform
- Drag-and-drop het ONNX bestand
- Importeer als Inference Overlay
  - ONNX als Neural Network in Project opgeslagen
- Configureer Satellite Overlay als input voor Inference Overlay
- Pas grid-grootte aan
- Update Overlays

AI Inference Resultaat Overlays

Labels
Wat is het?

Boom?

Scores
Hoe waarschijnlijk?

Rood is waarschijnlijkst

Masks
Welke pixels horen erbij?

Boxes
Potentiële match

Howto's

- https://previewsupport.tygron.com/wiki/How_to_export_AI_Training_Data
- https://previewsupport.tygron.com/wiki/How_to_train_your_own_AI_model_for_an_Inference_Overlay
- https://previewsupport.tygron.com/wiki/How_to_import_an_ONNX_file_using_drag_and_drop
- How to select specific input data for AI Inference

Wat zou u graag willen herkennen met een model op maat?
Neem contact op met Hedi (hedi@tygron.com)

Scan mij!