



# Deep Learning for Tree Counting and Building Extraction

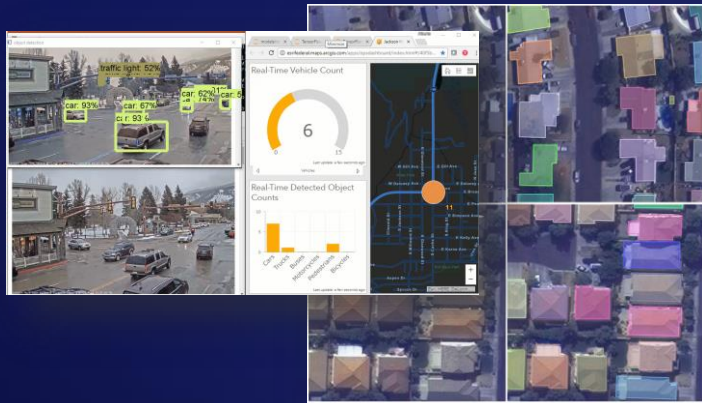
**Yoga Abdilah**

June 11<sup>th</sup>, 2020



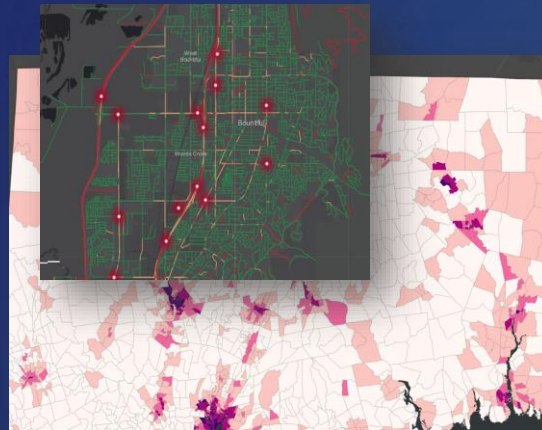
# GeoAI Pattern

## Object Detection



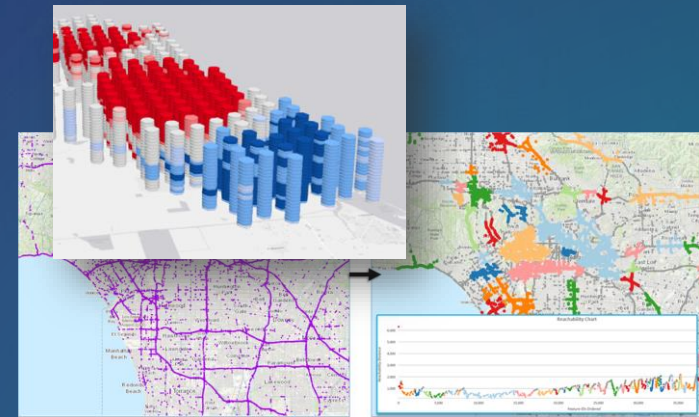
*Buildings, Road Segments,  
Swimming Pools, Blight,  
Graffiti, Overgrowth, Road  
Signs, Vehicles from CCTVs,  
and more*

## Prediction



*Water Pipe Breaks, Diseases,  
Crimes, Crashes, Incidents,  
Fires, Congestion, 911 Calls,*

## Pattern Detection



*Top Risky Segments, Emerging  
Hotspots of 911 Calls, Disease  
Clusters, and more*

# Examples for Imagery AI Workflows

*Object Detection, Instance Segmentation, Land Cover, Change Detection..*

Tree Counting



Building Footprints



Land Cover



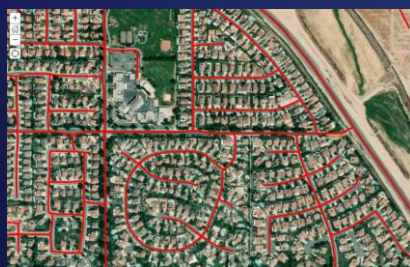
Pipeline Encroachment



Swimming Pools



Roads



Oil Pads



Road Cracks



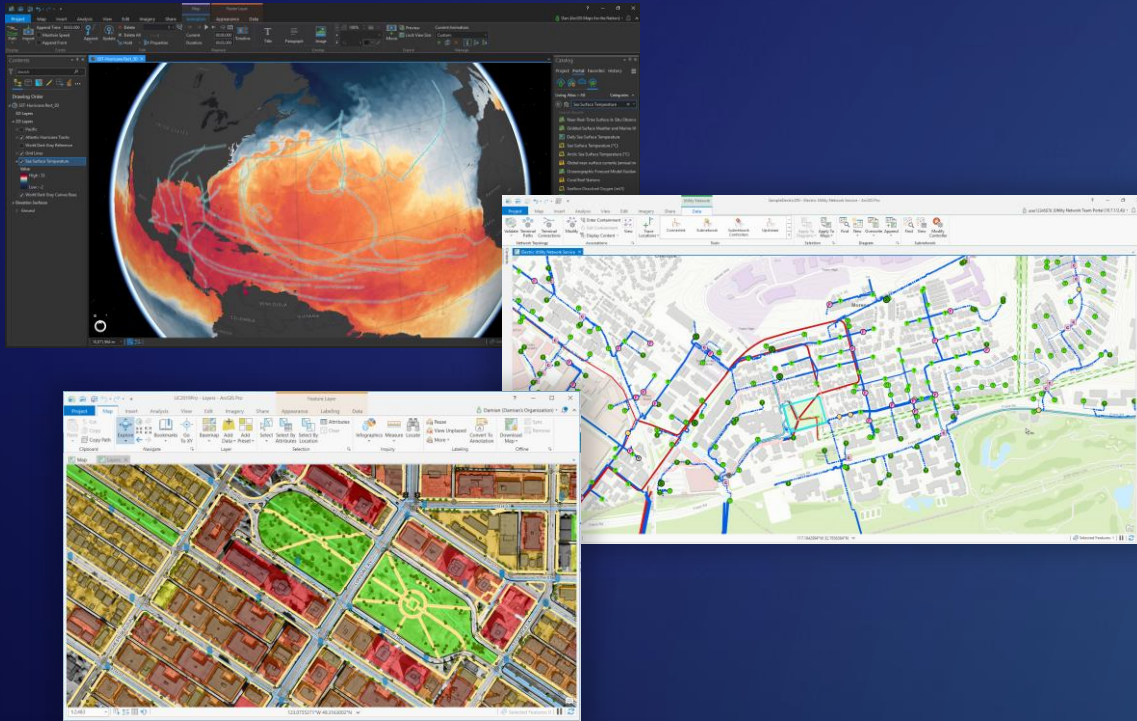
Cars



Damaged Structures



# ArcGIS Pro Professional GIS



Advanced Mapping, Visualization,  
Editing and Analysis

## Improvements

- Utility Management
- Faster Startup
- Editing Tools
- Catalog Preview
- Definition Expressions
- Contingent Attributes

## New

- Reports
- Dimensions
- Spell Check
- Publish to Server
- Deep Learning
- Parcel Management

## Coming

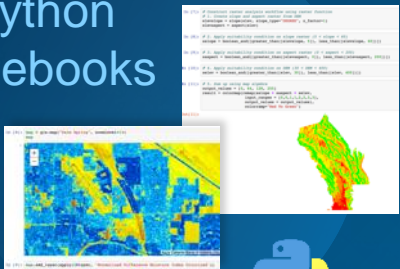
- Offset Printing
- More Parallel Processing
- Voxel Support



# Integrating Open Science, AI and Machine Learning

## Revolutionizing Spatial Analysis and Data Science

Python Notebooks



Analytic Services

Python API

Integration

AI & Machine Learning

Open Science Tools

Big Data GeoAnalytics

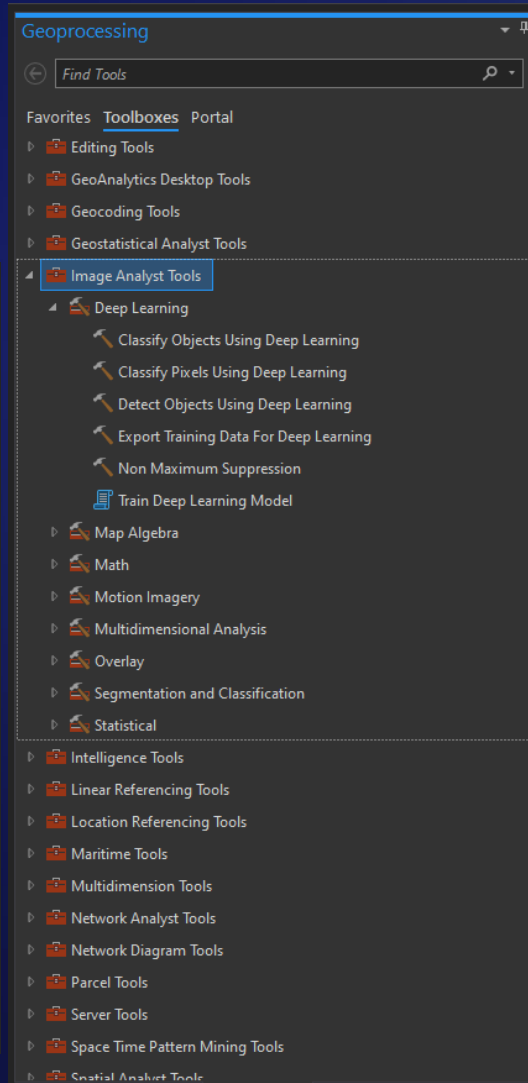
Spatial Analysis & Geoprocessing

Geospatial Infrastructure

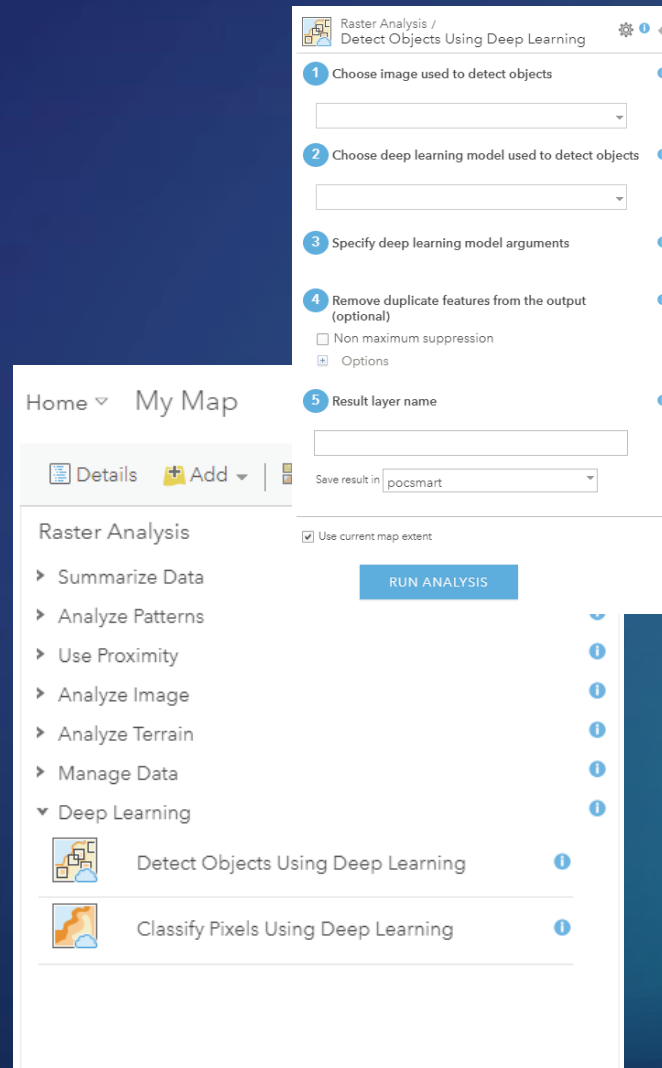


# GeoAI User Experience

## Pro



## Web



## Python API



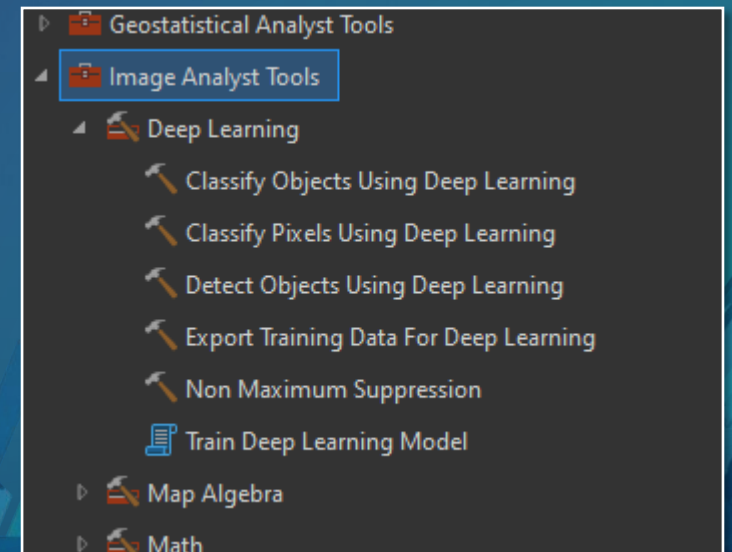
# Tree Counting

With Deep Learning in ArcGIS Pro

# Tree Counting

*with Deep Learning in ArcGIS Pro*

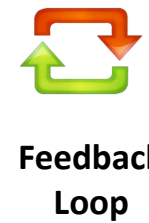
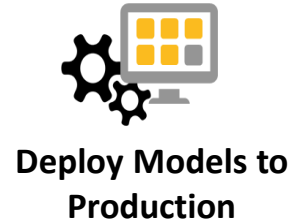
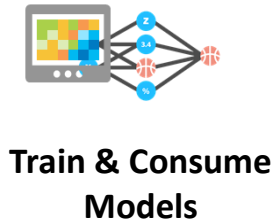
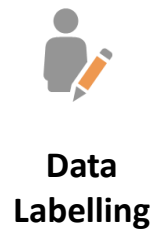
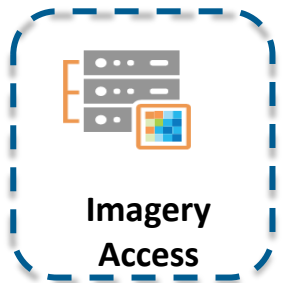
- Only available in **ArcGIS Pro**.
- Require **Image Analyst** extension.
- Object detection workflow first introduced in ArcGIS Pro 2.3
- Begin with installing deep learning frameworks for ArcGIS.





# End to End Tree Counting Workflow

- Identify how we access imagery data.
  - Raster file?
  - Mosaic Dataset?
  - Mosaic Dataset in a shared directory/datacenter?



# End to End Tree Counting Workflow

- Identify image format, spatial reference, number of bands.
- Specify processing extent.

Raster Information	
Columns	17761
Rows	25006
Number of Bands	3
Cell Size X	0.0859005781778122

Spatial Reference	
Projected Coordinate System	WGS 1984 UTM Zone 47N
Projection	Transverse Mercator
WKID	32647
Authority	EPSG
Linear Unit	Meters (1.0)
False Easting	500000.0
False Northing	0.0
Central Meridian	99.0
Scale Factor	0.9996
Latitude Of Origin	0.0



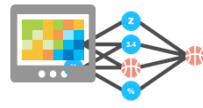
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



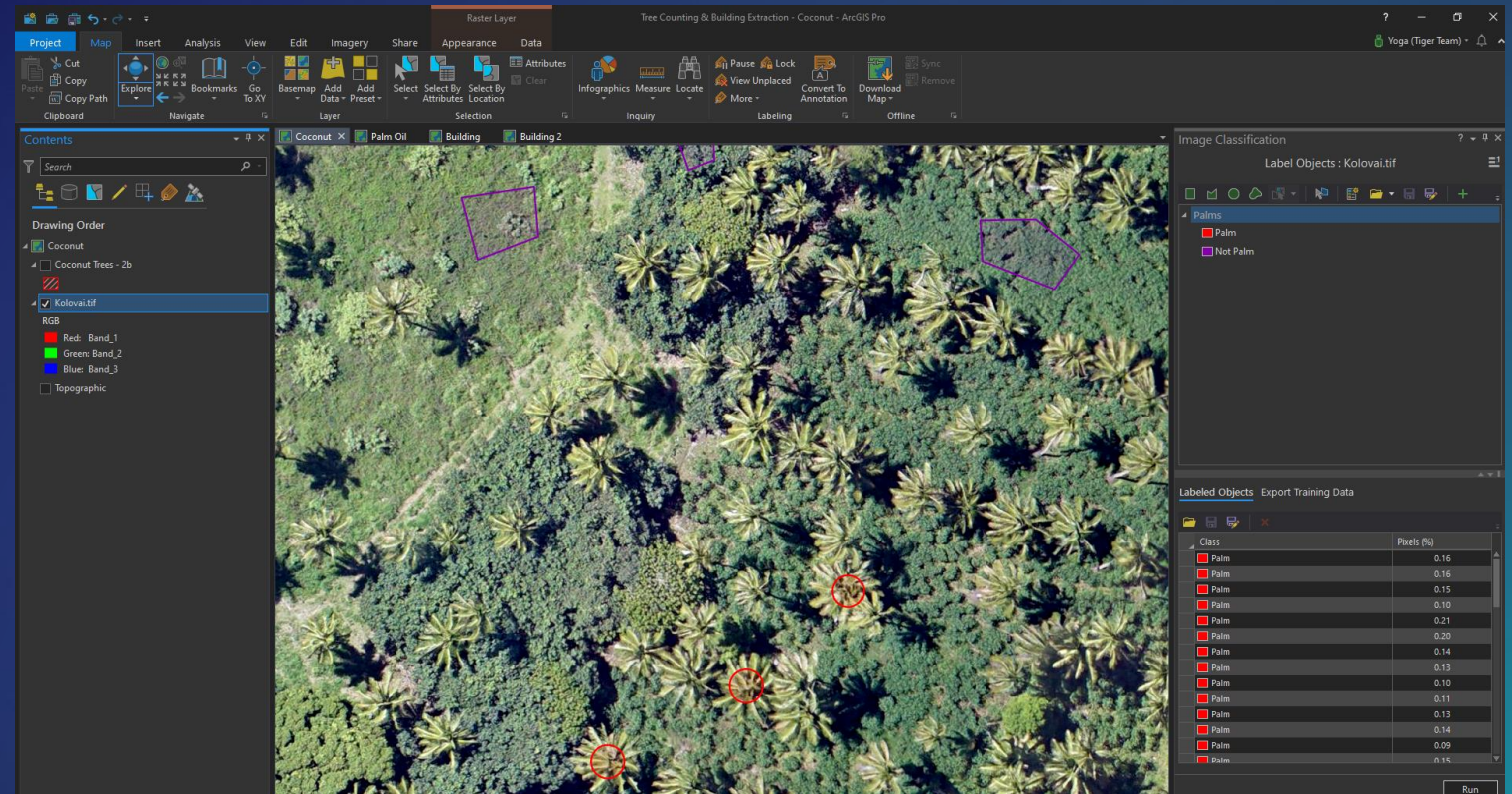
Feedback  
Loop



Take  
Action

# End to End Tree Counting Workflow

- Create training samples.
- Number of samples might vary
  - Method used
  - Image size



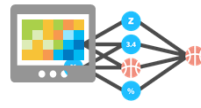
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



Feedback  
Loop



Take  
Action

# End to End Tree Counting Workflow

- Start by Exporting data
  - GP Toolbox: **Export Training Data for Deep Learning**



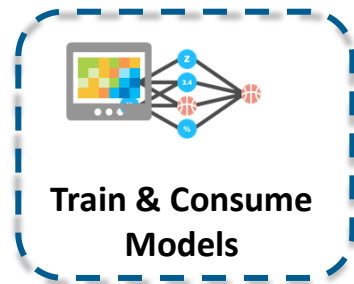
Imagery  
Access



Imagery  
Prep



Data  
Labelling



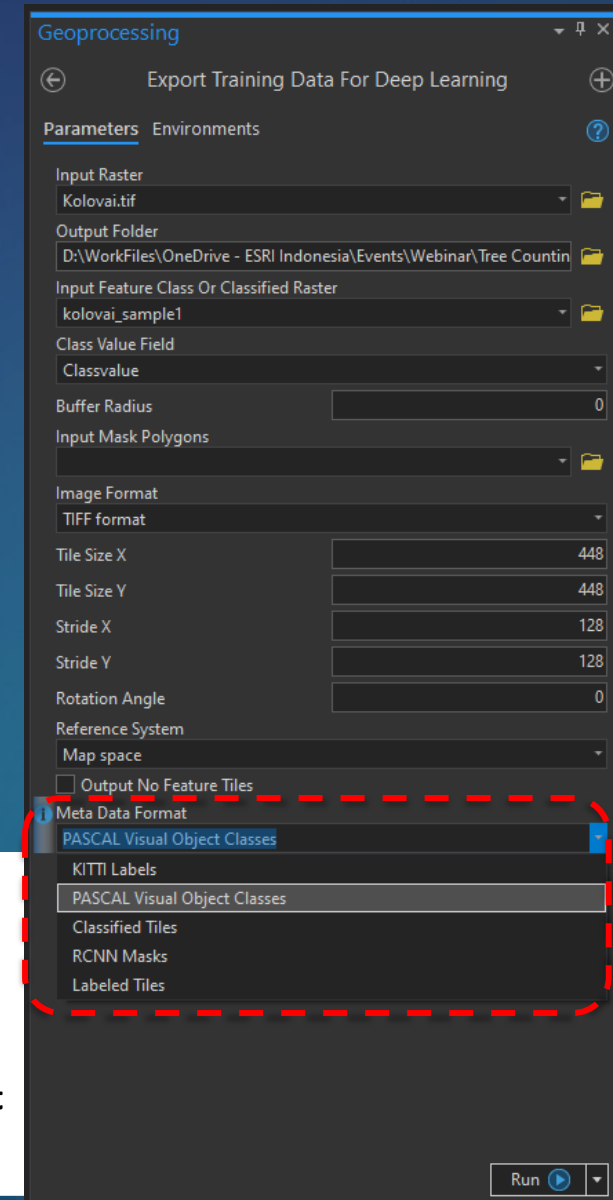
Train & Consume  
Models



Deploy Models to  
Production

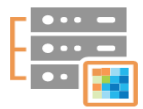
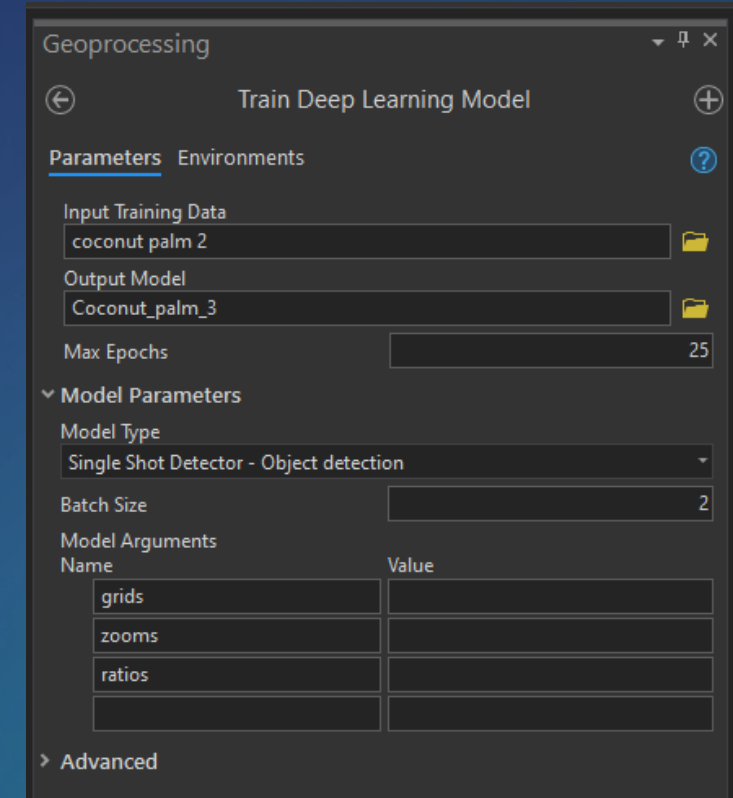


Run Inference at  
SCALE



# End to End Tree Counting Workflow

- Next: Train a deep learning model!
  - Using ArcGIS Pro (since 2.5)



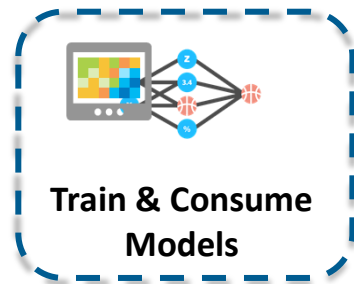
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



Feedback  
Loop



Take  
Action

# End to End Tree Counting Workflow

- Next: Train a deep learning model!
  - Using ArcGIS Pro (**since 2.5**)
    - Single Shot Detector
    - U-Net
    - PSPNET
    - RetinaNet
    - MaskRCNN

## Model Type (Optional)

Specifies the model type to use for training the deep learning model.

- Single Shot Detector - Object detection—The Single Shot Detector (SSD) approach will be used to train the model. SSD is used for object detection.
- U-Net - Pixel classification—The U-Net approach will be used to train the model. U-Net is used for pixel classification.
- Feature classifier - Object classification—The Feature Classifier approach will be used to train the model. This is used for object or image classification.
- Pyramid Scene Parsing Network - Pixel classification—The Pyramid Scene Parsing Network (PSPNET) approach will be used to train the model. PSPNET is used for pixel classification.
- RetinaNet - Object detection—The RetinaNet approach will be used to train the model. RetinaNet is a model type used for object detection.
- MaskRCNN - Object detection—The MaskRCNN approach will be used to train the model. MaskRCNN is used for object detection. It is used for instance segmentation, which is precise delineation of objects in an image. This model type can be used to detect building footprints. It uses the MaskRCNN meta format for training data as input.



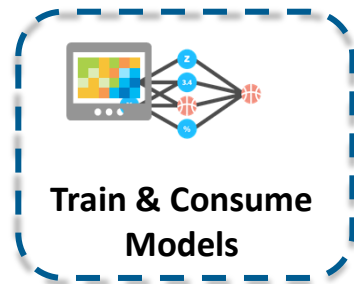
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



Feedback  
Loop



Take  
Action

# Tree Counting

## *Image Classification vs Object Detection*

Image Classification



Predict **object** in an image



Object Detection



Predict **object** in an image  
+  
**Location** of the image (bounding boxes)



# Tree Counting

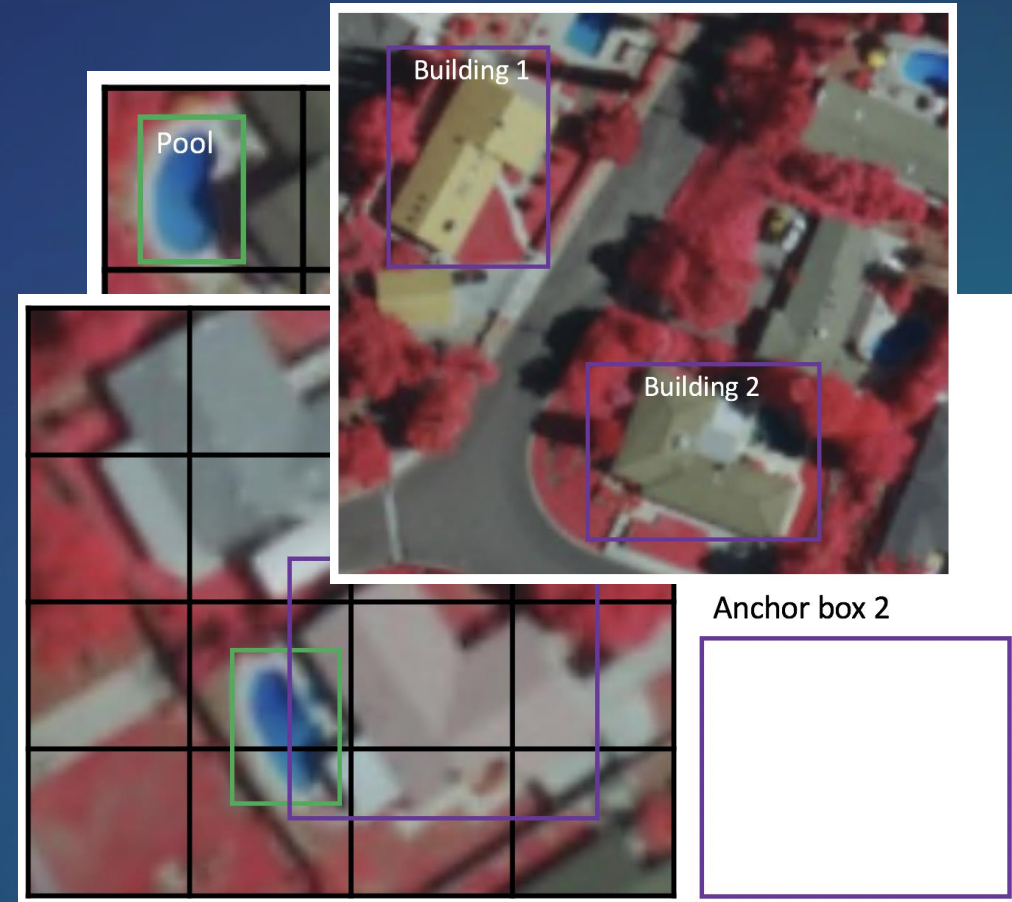
Method – Single Shot Detector (SSD)

- SSD divides the image using a grid
- Each grid cell be responsible for detecting objects
  - **Probability** that there is an object?
  - **Height** of the bounding box?
  - **Width** of the bounding box?
  - **Horizontal** coordinate of the **center point** of the bounding box?
  - **Vertical** coordinate of the **center point** of the bounding box?

Anchor box

Zoom level

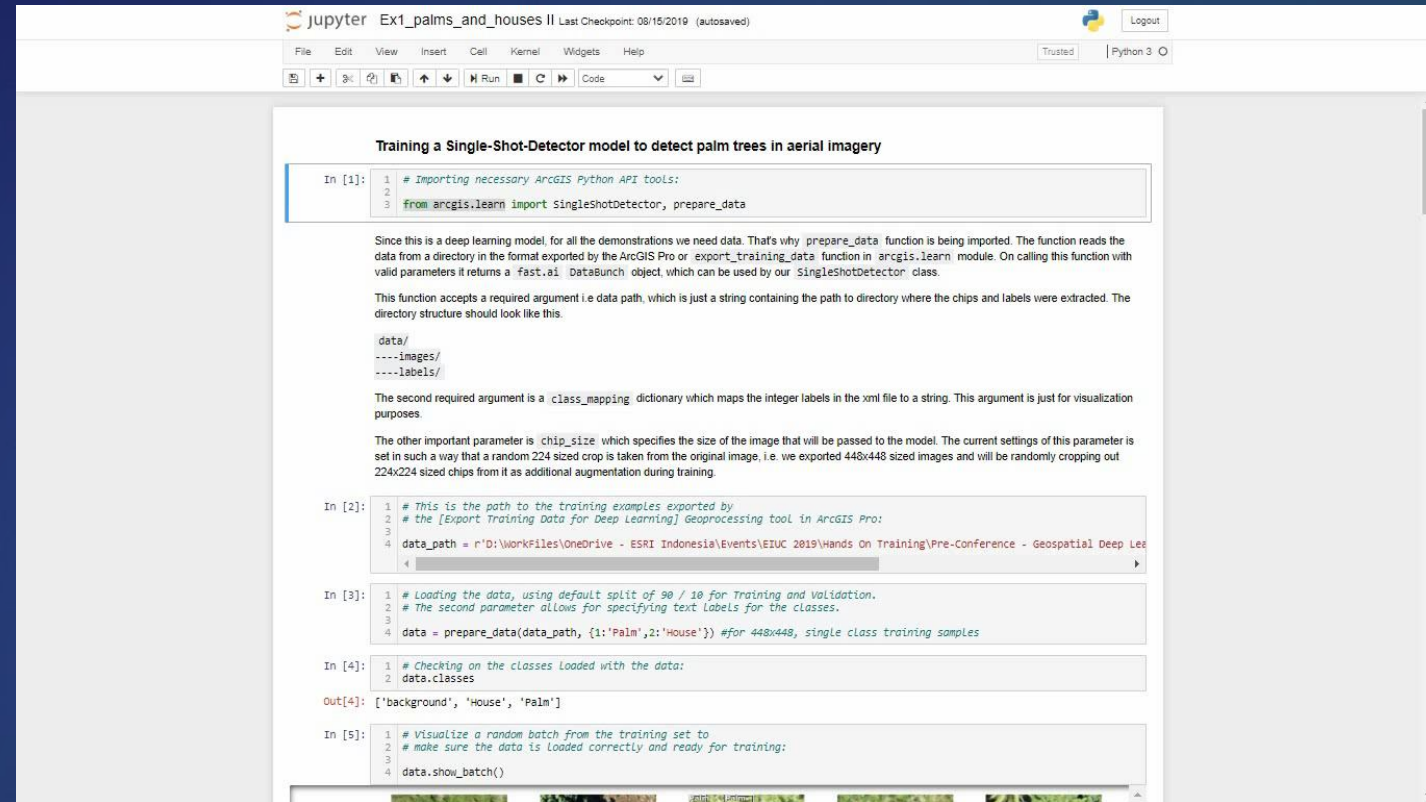
Aspect ratio





# End to End Tree Counting Workflow

- Next: Train a deep learning model!
  - Using ArcGIS Pro (since 2.5)
  - Using **arcgis.learn** (ArcGIS API for Python)
  - Using **open libraries**



```
In [1]: 1 # Importing necessary ArcGIS Python API tools:
        2
        3 from arcgis.learn import SingleShotDetector, prepare_data

Since this is a deep learning model, for all the demonstrations we need data. That's why prepare_data function is being imported. The function reads the data from a directory in the format exported by the ArcGIS Pro or export_training_data function in arcgis.learn module. On calling this function with valid parameters it returns a fast.ai DataBunch object, which can be used by our SingleShotDetector class.

This function accepts a required argument i.e data path, which is just a string containing the path to directory where the chips and labels were extracted. The directory structure should look like this

data/
---Images/
---Labels/

The second required argument is a class_mapping dictionary which maps the integer labels in the xml file to a string. This argument is just for visualization purposes.

The other important parameter is chip_size which specifies the size of the image that will be passed to the model. The current settings of this parameter is set in such a way that a random 224 sized crop is taken from the original image, i.e. we exported 448x448 sized images and will be randomly cropping out 224x224 sized chips from it as additional augmentation during training.

In [2]: 1 # This is the path to the training examples exported by
        2 # the [Export Training Data for Deep Learning] geoprocessing tool in ArcGIS Pro:
        3
        4 data_path = r'D:\WorkFiles\OneDrive - ESRI Indonesia\Events\EIUC 2019\Hands On Training\Pre-Conference - Geospatial Deep Lea
        5
In [3]: 1 # Loading the data, using default split of 90 / 10 for Training and Validation.
        2 # The second parameter allows for specifying text labels for the classes.
        3
        4 data = prepare_data(data_path, {'1':'Palm',2:'House'}) #for 448x448, single class training samples

In [4]: 1 # Checking on the classes loaded with the data:
        2 data.classes
Out[4]: ['background', 'House', 'Palm']

In [5]: 1 # Visualize a random batch from the training set to
        2 # make sure the data is loaded correctly and ready for training:
        3
        4 data.show_batch()
```



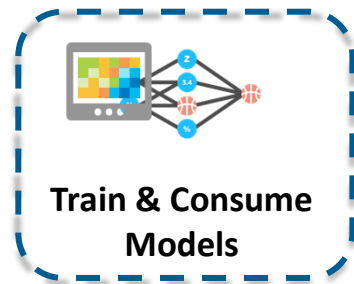
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



Feedback  
Loop

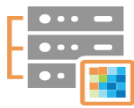


Take  
Action

# End to End Tree Counting Workflow

- Deploy Deep Learning model
  - Accessible also as end or **Deep Learning Package**

Name	Date modified	Type	Size
Title			Modified
<input type="checkbox"/> Palm Model 2		Deep Learning Package	Jun 8, 2020
<input type="checkbox"/> Tree Census & Health Monitoring		Web Map	Jun 8, 2020
<input type="checkbox"/> Tree Census & Health Monitoring Dashboard		Dashboard	Jun 8, 2020
<input type="checkbox"/> Palm Oil Trees_VARI		Feature Layer (hosted)	Jun 8, 2020
<input type="checkbox"/> Palm Oil Trees_VARI		Service Definition	Jun 8, 2020



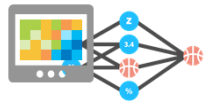
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



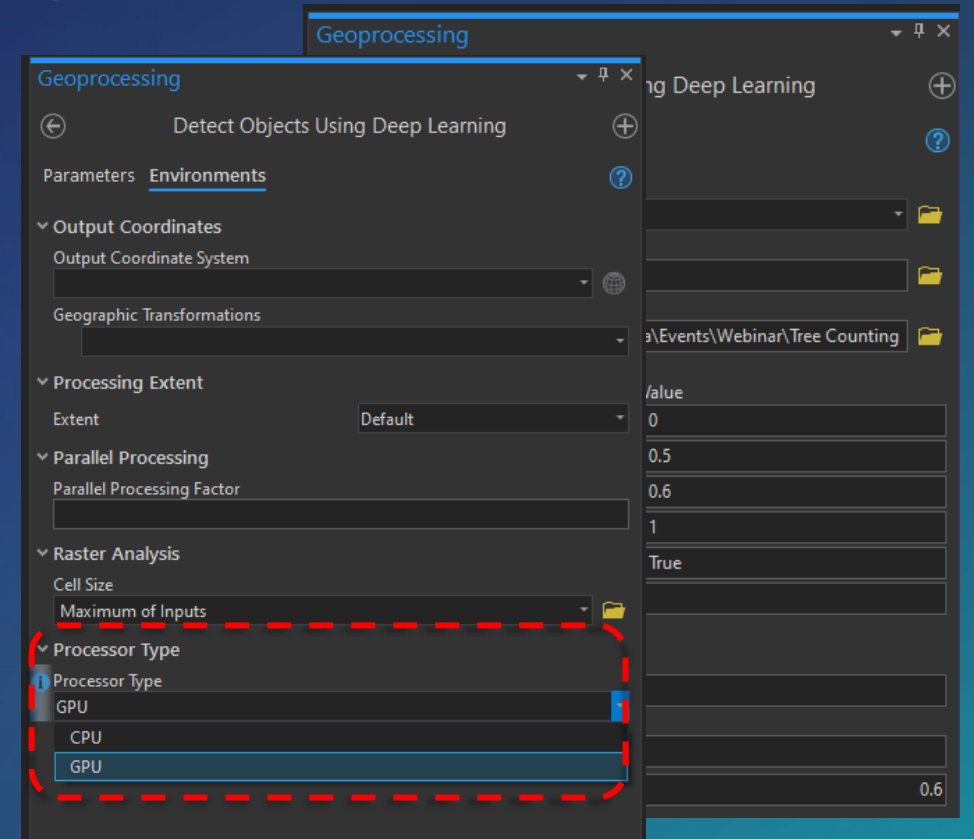
Feedback  
Loop



Take  
Action

# End to End Tree Counting Workflow

- Deploy model & start object detection
  - GP Toolbox: **Detect Objects Using Deep Learning**



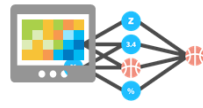
Imagery  
Access



Imagery  
Prep



Data  
Labelling



Train & Consume  
Models



Deploy Models to  
Production



Run Inference at  
SCALE



Feedback  
Loop



Take  
Action

# End to End Tree Counting Workflow

- QC
- Utilize Result:
  - **Tree Health Monitoring**



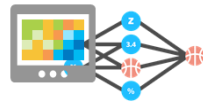
Imagery Access



Imagery Prep



Data Labelling



Train & Consume Models



Deploy Models to Production



Run Inference at SCALE



Feedback Loop



Take Action

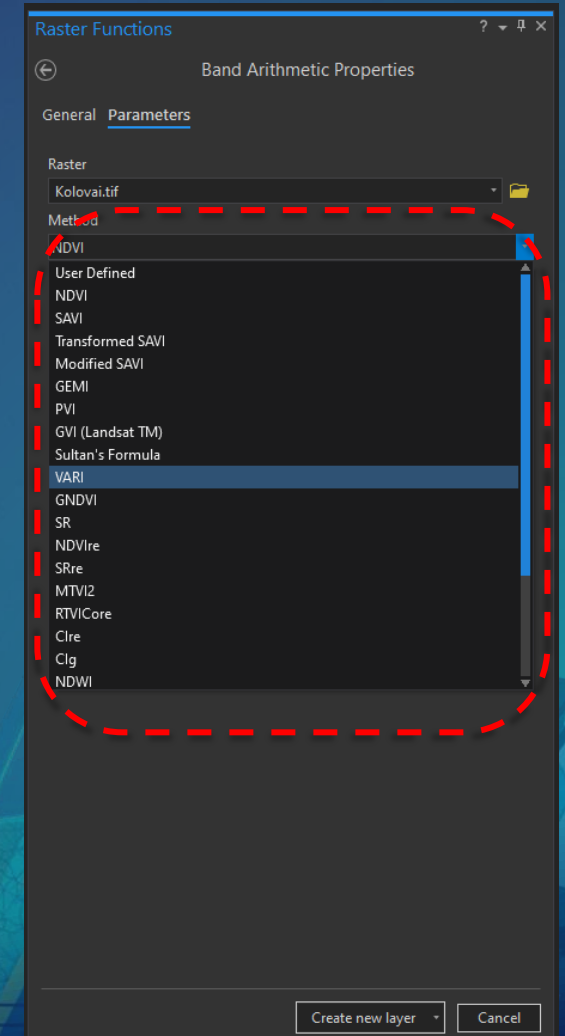
# Tree Health Monitoring

- Assessing vegetation health using **Visible Atmospherically Resistant Index (VARI)**

- Indirect measure of leaf area index (LAI) and vegetation fraction (VF)
- Using reflectance values from the visible wavelength

$$(R_g - R_r) / (R_g + R_r - R(R_g - R_b))$$

- **Preferred** to use both visible and near infrared (NIR) wavelength





**Demo**

# Tree Counting

with Deep Learning in ArcGIS Pro

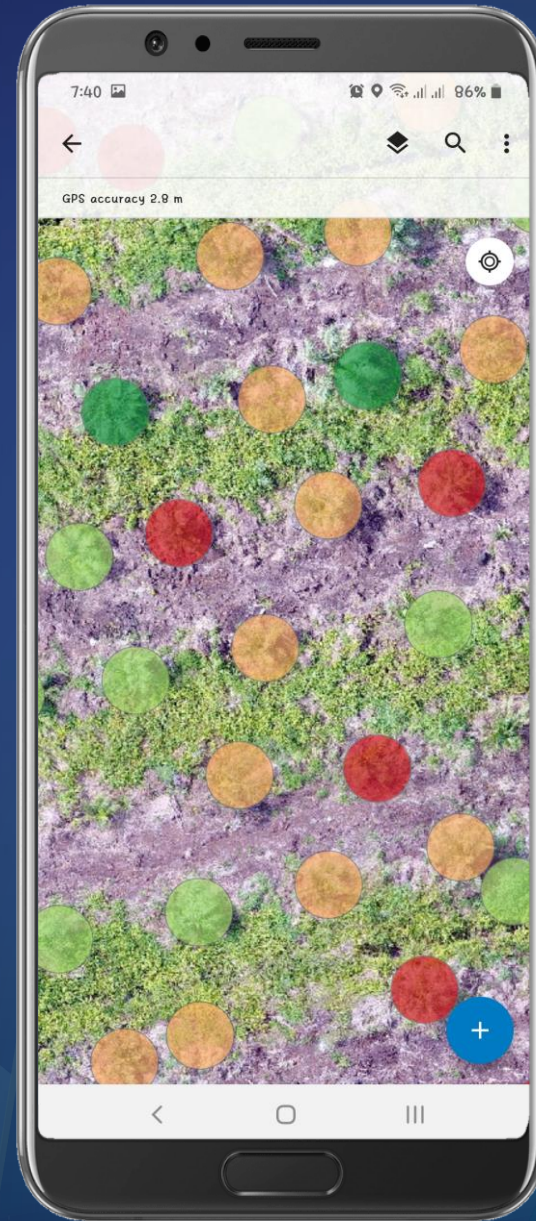
- Tree Health Monitoring
- Field Survey for Verification



# Tree Counting

*with Deep Learning in ArcGIS Pro*

- **Tree Health Monitoring**
- **Field Survey for Verification**

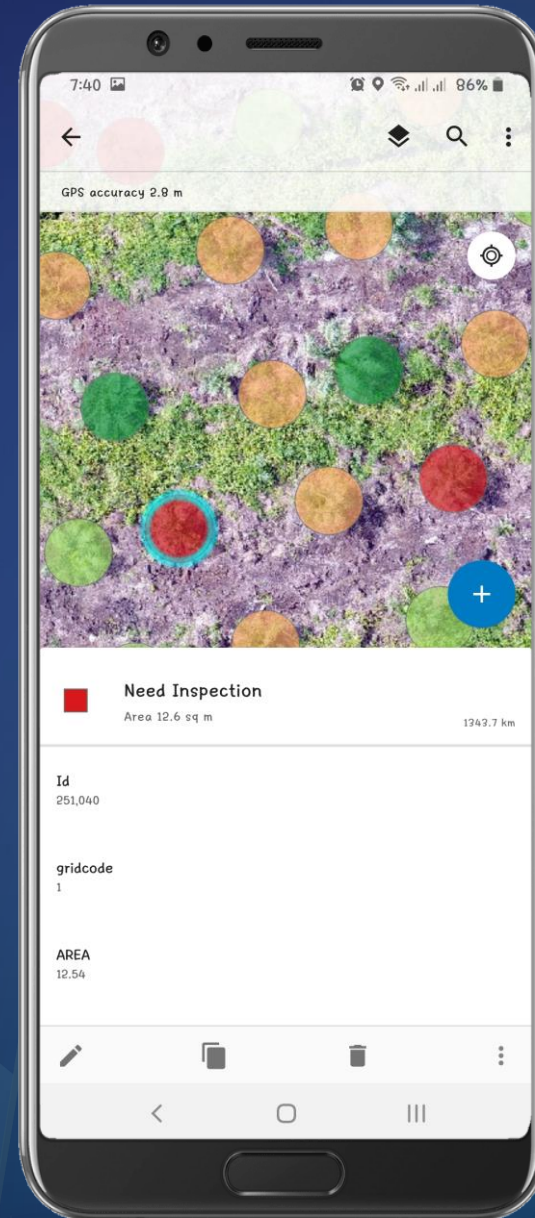




# Tree Counting

*with Deep Learning in ArcGIS Pro*

- **Tree Health Monitoring**
- **Field Survey for Verification**



# Building Extraction

With Deep Learning in ArcGIS Pro

# Building Extraction

*with Deep Learning in ArcGIS Pro*

- Only available in **ArcGIS Pro**.
- Require **Image Analyst** extension.
- Object detection workflow first introduced in ArcGIS Pro 2.3
- Begin with installing deep learning frameworks for ArcGIS.



# End to End Building Extraction Workflow



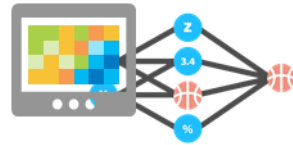
**Imagery  
Access**



**Imagery  
Prep**



**Data  
Labelling**



**Train &  
Consume  
Models**



**Deploy Models  
to Production**



**Run Inference at  
SCALE**



**Feedback  
Loop**



**Take  
Action**



**Demo**

# Conclusion

- **Deep Learning workflow helps simplify & accelerate object detection process**
- **Deep Learning is supported in ArcGIS Pro + Image Analyst Extension**
- **ArcGIS Pro support end-to-end workflow**
- **GeoAI capabilities could be integrated with other open science libraries**



**esri**

**THE  
SCIENCE  
OF  
WHERE**