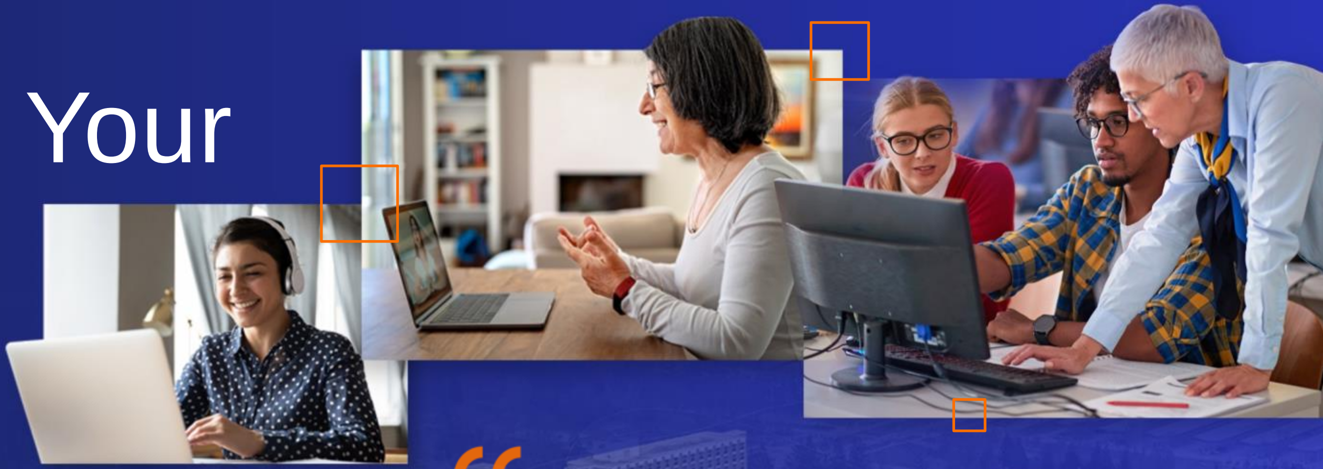
# Demystifying Deep Learning Analysis for Your Students

## A Higher Education Webinar



> The willingness to experiment with innovative approaches to GIS has yielded benefits for numerous field-based teaching and research activities across the university.
>
> *– Peter Knoop | University of Michigan*

# Webinar Housekeeping

- Microphones will be muted
- Please enter your questions in the chat box
- Q&A session at the conclusion of the webinar
- Webinar & slides will be available
- Survey following the webinar

# Poll time…

Are you teaching Imagery & Remote Sensing?

# Presenters

Vinay Viswambharan
Principal Product Manager
ArcGIS Imagery Product team

Sandeep Kumar
Senior Product Engineer
Data Science

Canserina Kurnia
Senior Solution Engineer
Education team

# Agenda

- Deep Learning overview, applications and an end-to-end example
- Deep learning workflow in ArcGIS
  - Option 1:  Use pre-trained deep learning models
  - Option 2: Train your own deep learning models
- Deep learning concepts in details
- Few best practices and resources

# Poll time…

What is your level of comfort with Deep Learning?

# Why are you here?

## The need for automation has risen

More sensors

Large volumes of imagery and raw data

Velocity of data

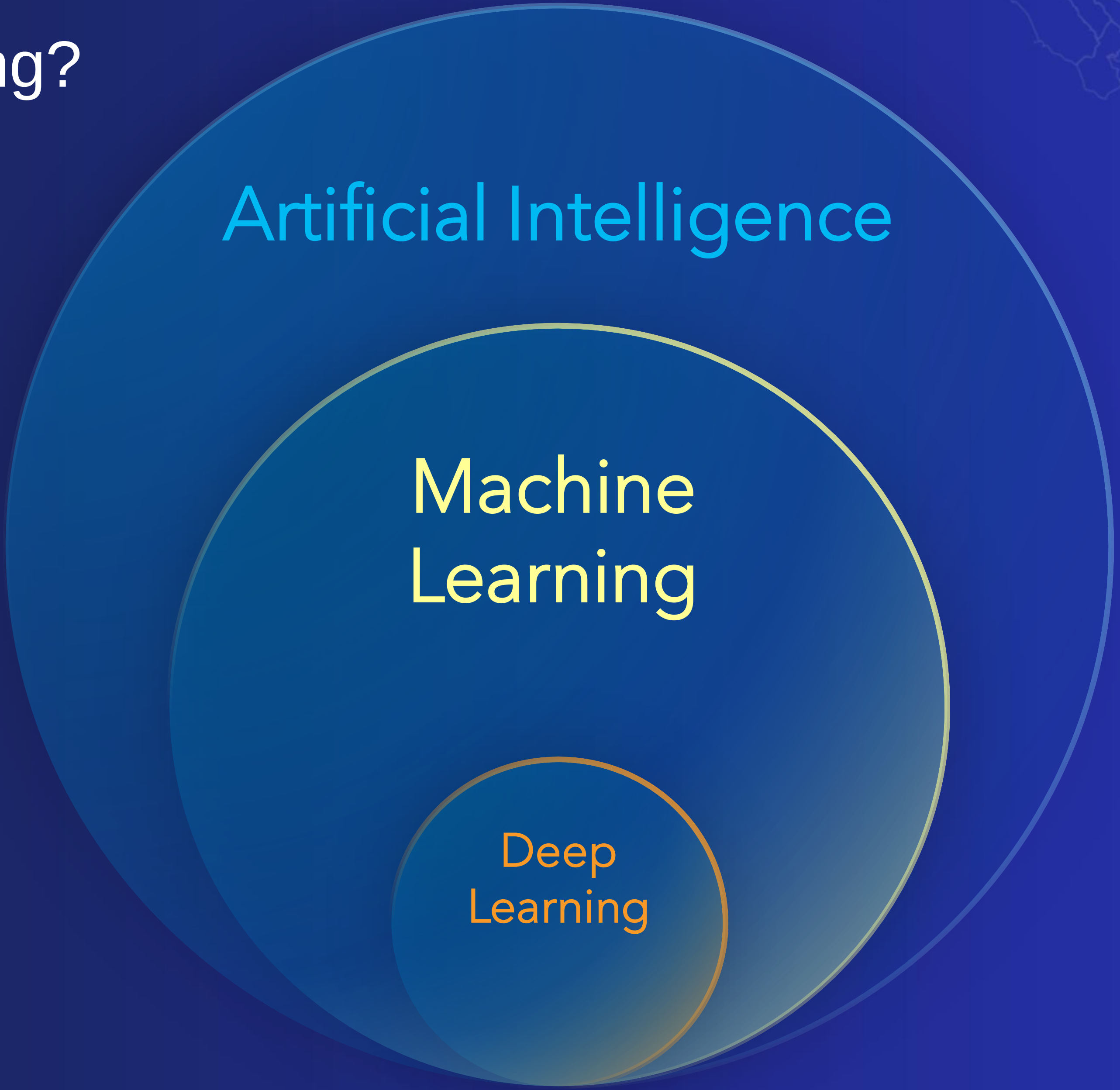
Automation

Accuracy

### Defacto Solution

- Artificial Intelligence, Machine learning, Deep Learning

# What is Deep Learning?



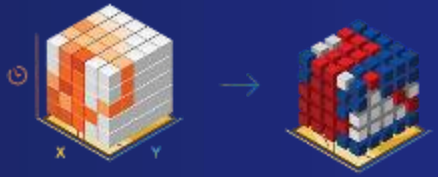
# Machine Learning in ArcGIS

## Classification

- Pixel & Object Based
- Image Segmentation
- Maximum Likelihood
- Random Trees
- Support Vector Machine

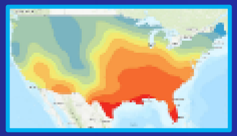
## Clustering

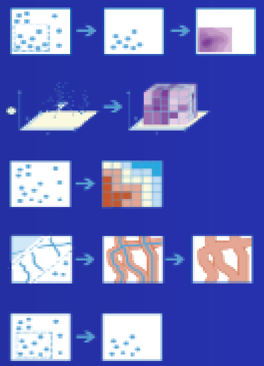
- Spatially Constrained Multivariate Clustering
- Multivariate Clustering
- Density-based Clustering
- Hot Spot Analysis
- Cluster and Outlier Analysis
- Space Time Pattern Mining



## Prediction

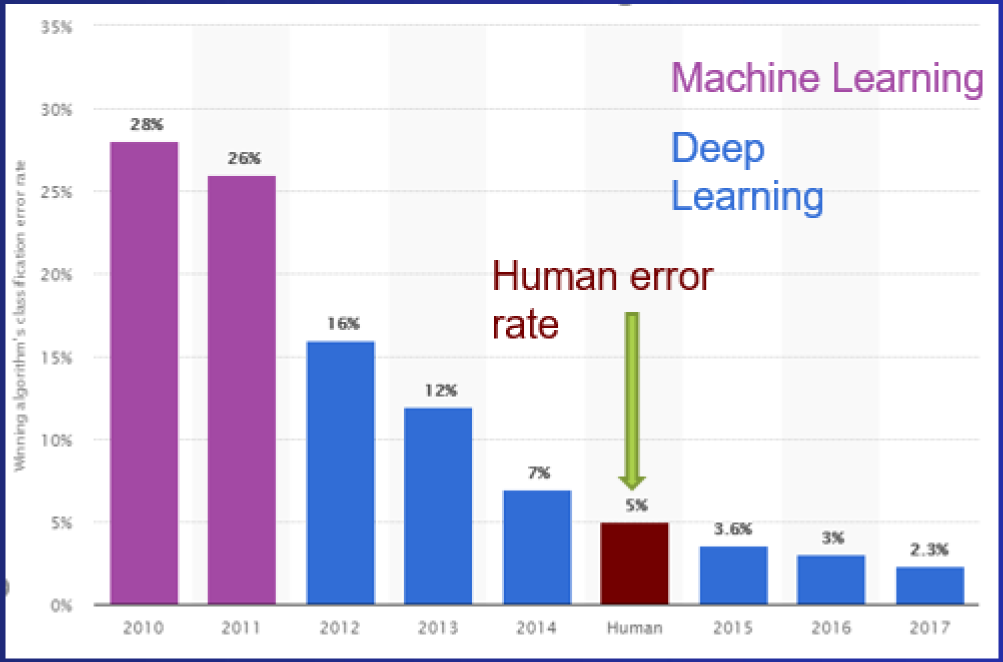
- Empirical Bayesian Kriging
- Areal Interpolation
- EBK Regression Prediction
- Ordinary Least Squares Regression and Exploratory Regression
- Geographically Weighted Regression





# How has Deep Learning Evolved

**Computer vision is now almost as good, if not better, than human vision**



ImageNet Visual Recognition Challenge error rate

# Deep Learning Applications and Models in ArcGIS

- 30 different models for various geospatial workflows
- Models for edge detection, change detection, road extraction and image translation
- Models for non spatial data
  - models for time-series data
  - models for natural language processing
- Allows integration with popular ML libs – scikit-learn

Object Classification Assign a label to a given image / geographical feature

Object Detection Find objects and their locations (bounding boxes)

Pixel Classification Assign a label to each pixel

Instance Segmentation Find objects and their precise locations (polygonal features)

Road Extraction Extract Roads from Aerial or Satellite Imagery

Image Translation Translate images from one domain to another

Edge Detection Find pixels belonging to edges

Change Detection Identify areas of persistent change

Models:
- ChangeDetector (based on STANet)

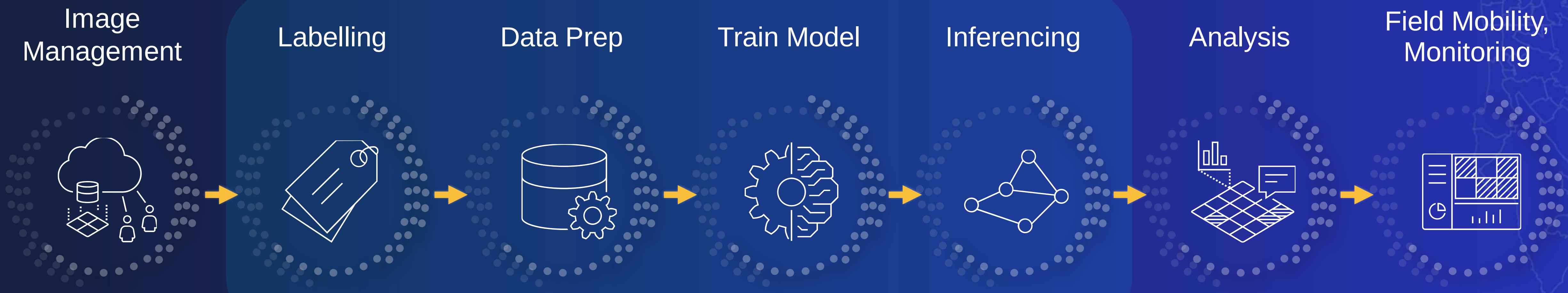
Applications:
- Identify new buildings
- Identify new construction

# Deep learning Applications

- Tour of DL applications
- Damage Classification Scenario

# Deep Learning Workflow in ArcGIS

End-to-end from raw imagery to structured information products

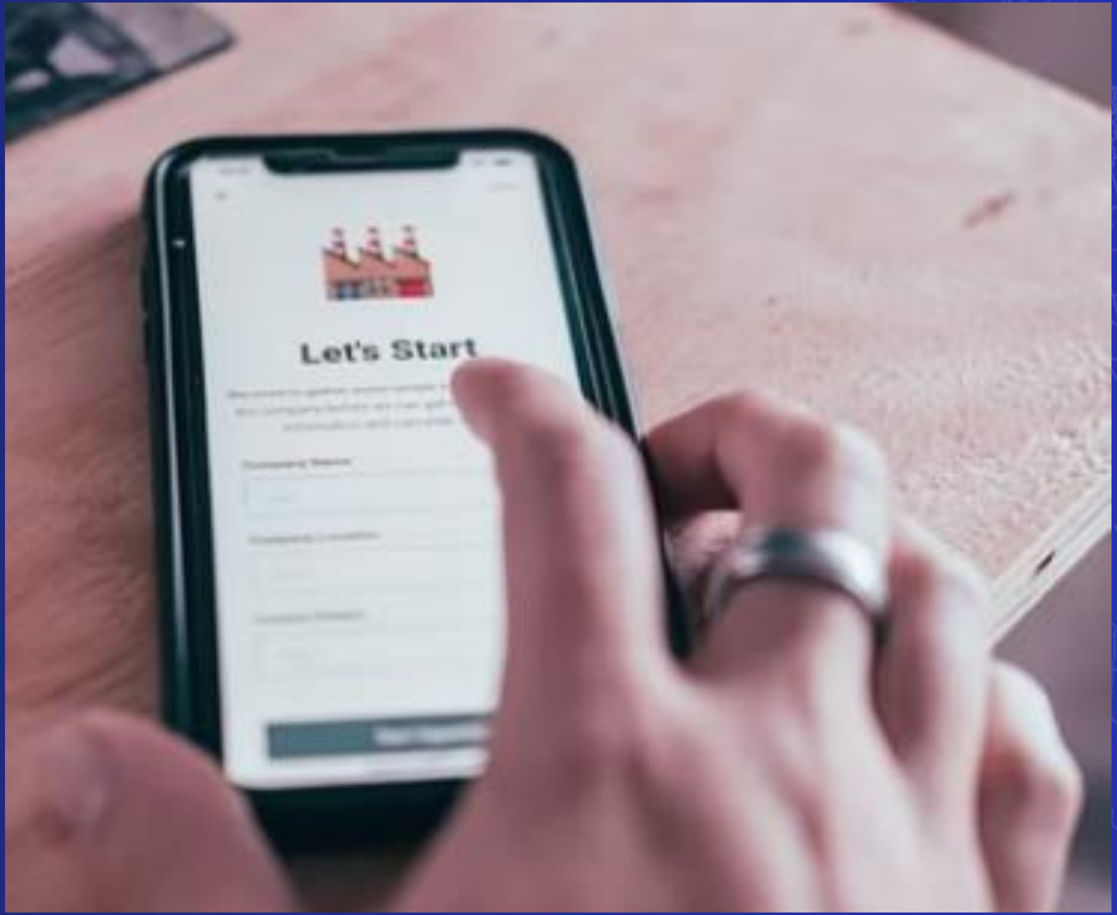


# Poll time…

What is the primary data type you would like to run Deep Learning on?

# Getting Started

- **Option 1: Use pre-trained models**
- **Option 2: Train your own model**



# Deep Learning in ArcGIS

## Option1: Pre-trained models

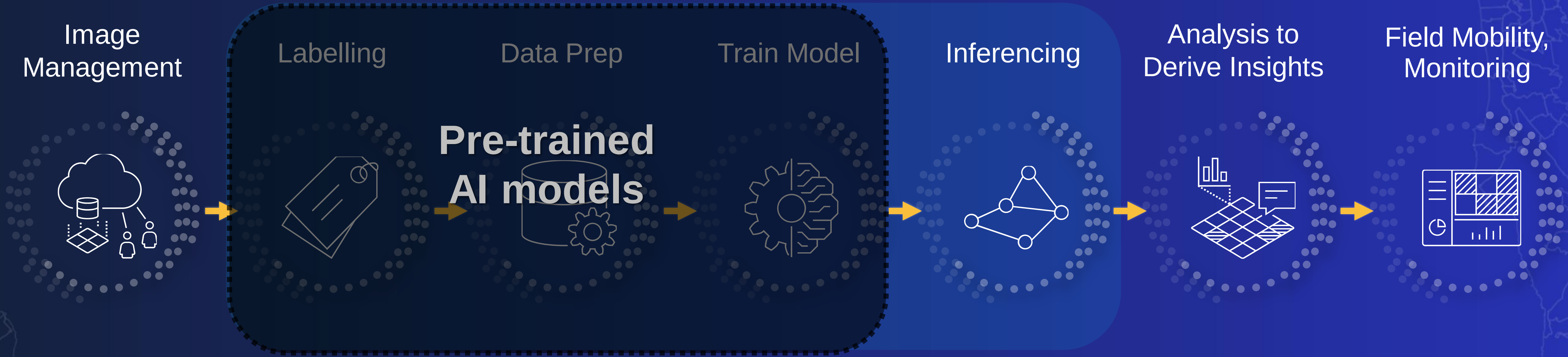
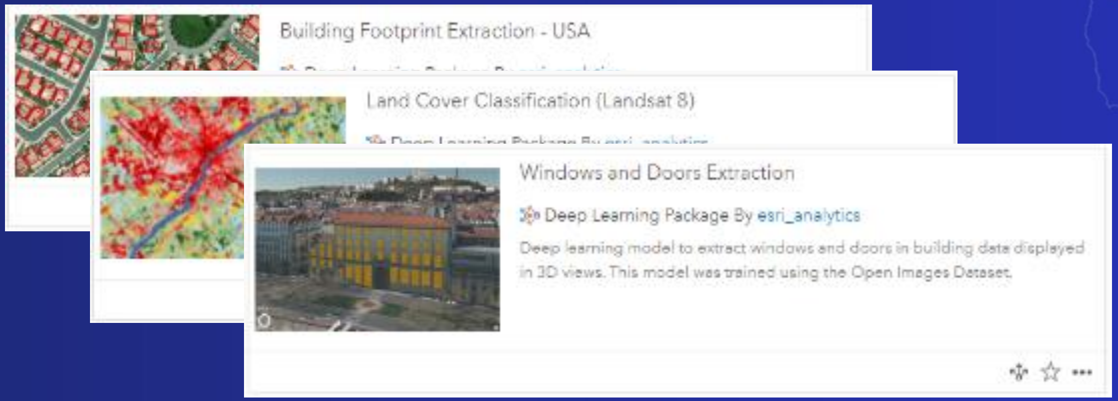


Image Management → Labelling → Data Prep → Train Model → Inferencing → Analysis to Derive Insights → Field Mobility, Monitoring

**Pre-trained AI models**

- Eliminates:
  - Imagery requirements for model training
  - Labelling requirements
  - Training AI models
  - Massive compute requirements



Building Footprint Extraction - USA

Land Cover Classification (Landsat 8)

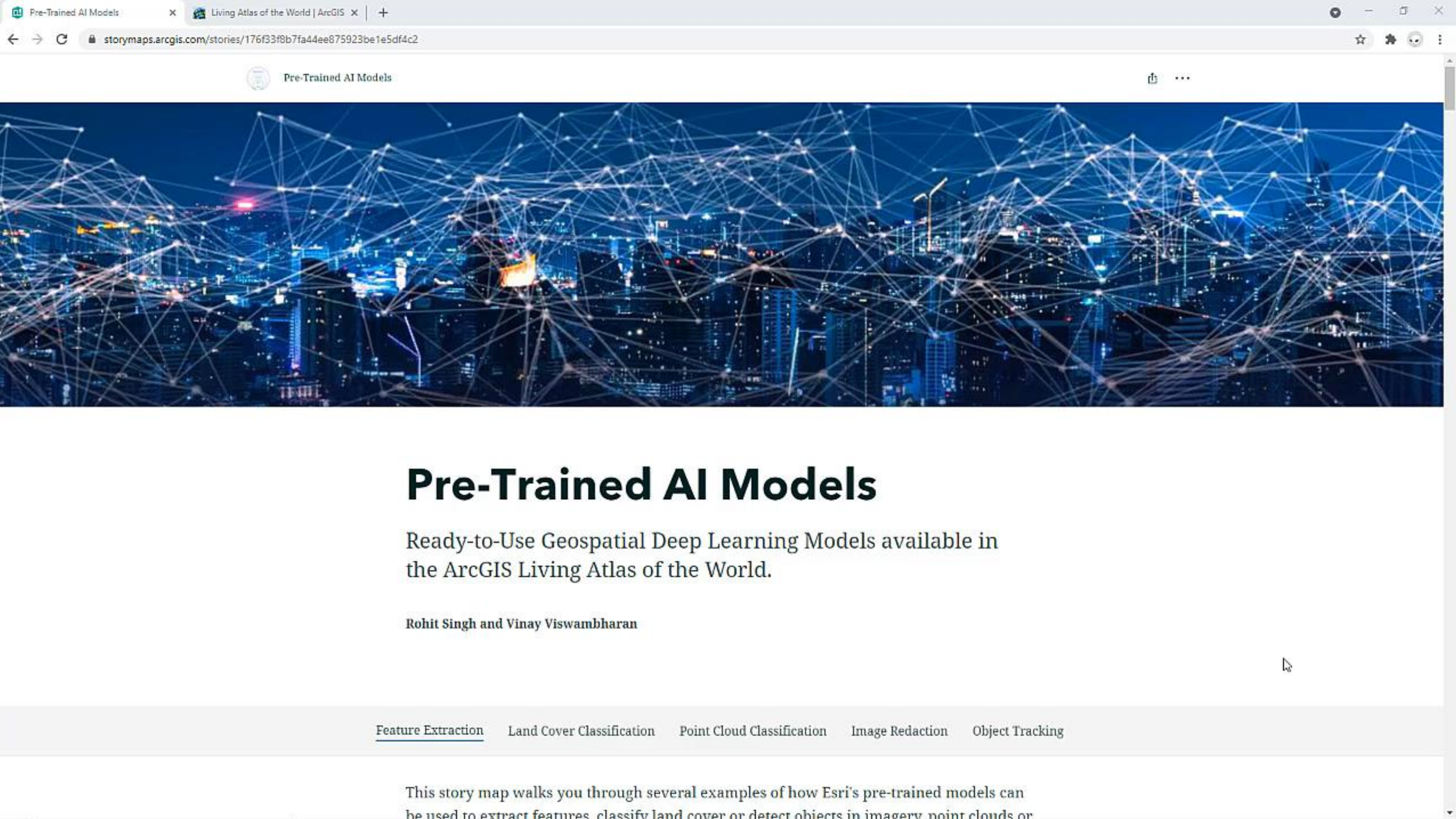
Windows and Doors Extraction

Deep Learning Package By esri_analytics

Deep learning model to extract windows and doors in building data displayed in 3D views. This model was trained using the Open Images Dataset.

# Pre-trained models – A tour

- Living Atlas - Review models
- Developing a GIS with pre-trained models


Pre-Trained AI Models
Living Atlas of the World | ArcGIS
storymaps.arcgis.com/stories/176f33f8b7fa44ee875923be1e5df4c2
Pre-Trained AI Models

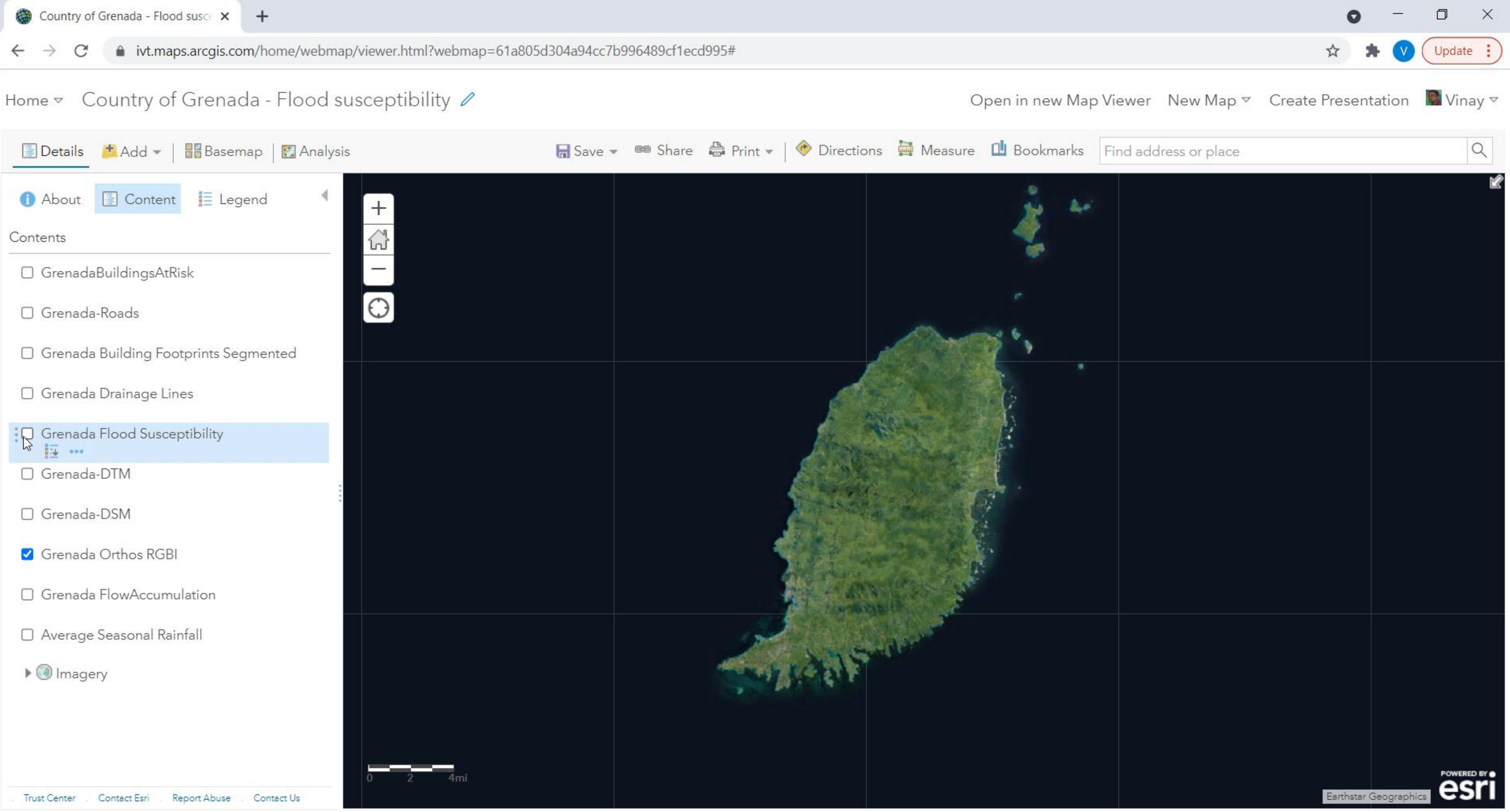
# Pre-Trained AI Models

Ready-to-Use Geospatial Deep Learning Models available in the ArcGIS Living Atlas of the World.

**Rohit Singh and Vinay Viswambharan**

Feature Extraction
Land Cover Classification
Point Cloud Classification
Image Redaction
Object Tracking

This story map walks you through several examples of how Esri's pre-trained models can be used to extract features, classify land cover or detect objects in imagery, point clouds or

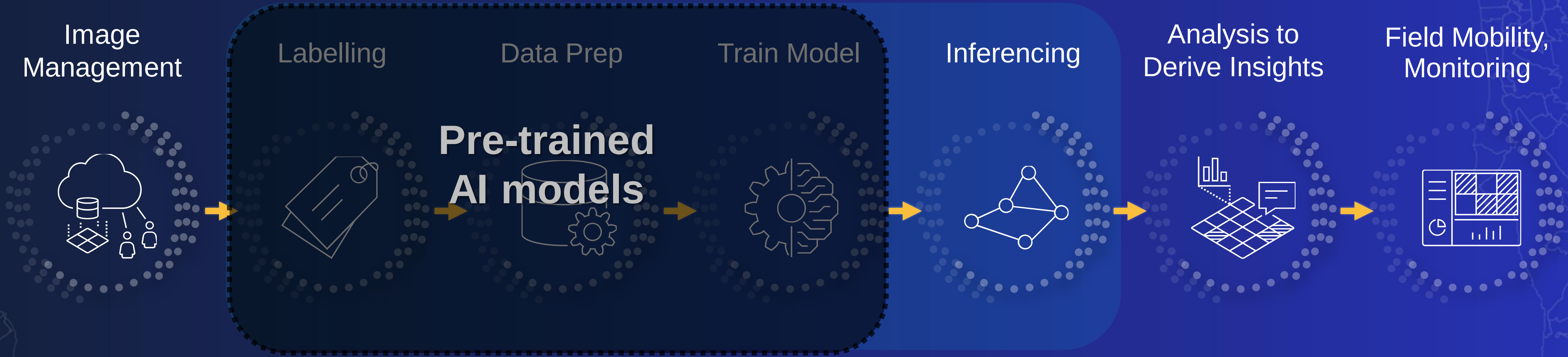


Contents

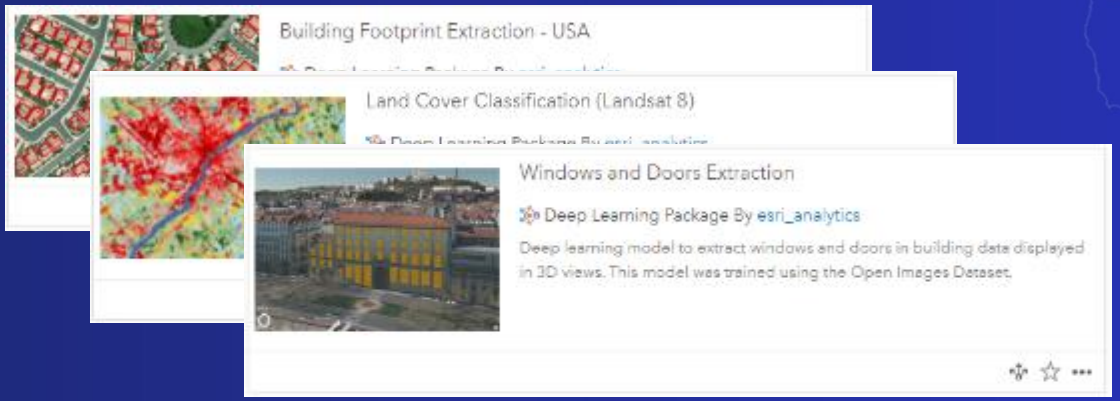
- ☐ GrenadaBuildingsAtRisk
- ☐ Grenada-Roads
- ☐ Grenada Building Footprints Segmented
- ☐ Grenada Drainage Lines
- ☐ Grenada Flood Susceptibility
- ☐ Grenada-DTM
- ☐ Grenada-DSM
- ☑ Grenada Orthos RGBI
- ☐ Grenada FlowAccumulation
- ☐ Average Seasonal Rainfall
- Imagery

# Deep Learning in ArcGIS

## Option1: Pre-trained models



- Eliminates:
  - Imagery requirements for model training
  - Labelling requirements
  - Training AI models
  - Massive compute requirements



Building Footprint Extraction - USA

Land Cover Classification (Landsat 8)

Deep Learning Package By esri_analytics

Windows and Doors Extraction

Deep Learning Package By esri_analytics

Deep learning model to extract windows and doors in building data displayed in 3D views. This model was trained using the Open Images Dataset.

# Deep Learning in ArcGIS

## Option 2: Train your own models

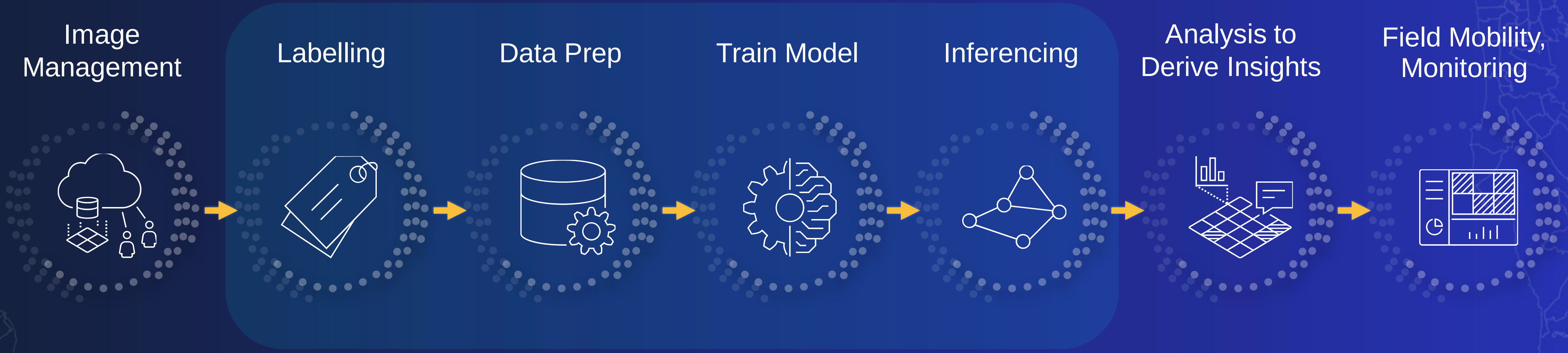
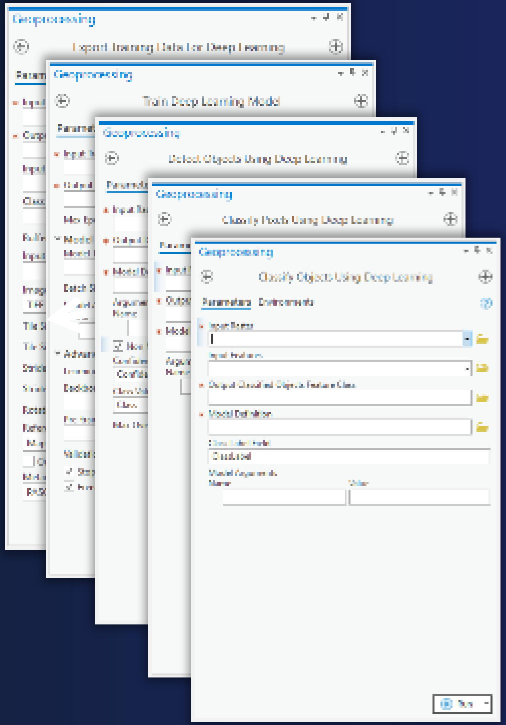


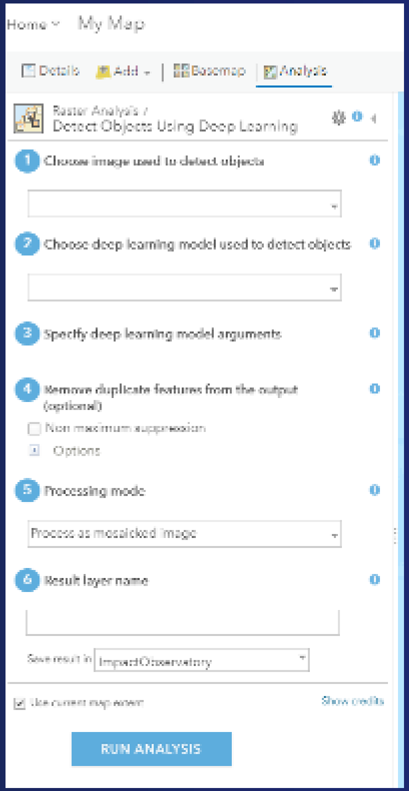
Image Management → Labelling → Data Prep → Train Model → Inferencing → Analysis to Derive Insights → Field Mobility, Monitoring

- End-to-end workflow
- Models designed for
  - Specific geographies
  - Resolutions
  - Imagery properties
  - Specific asset types

# Accessing Deep Learning Capabilities in ArcGIS

## Route I: GUI Tools


Geoprocessing
Export Training Data For Deep Learning
Geoprocessing
Train Deep Learning Model
Geoprocessing
Detect Objects Using Deep Learning
Geoprocessing
Classify Pixels Using Deep Learning
Geoprocessing
Classify Objects Using Deep Learning
Parameters
Environments
Run



Raster Analysis /
Detect Objects Using Deep Learning

1 Choose image used to detect objects

2 Choose deep learning model used to detect objects

3 Specify deep learning model arguments

4 Remove duplicate features from the output (optional)

□ Non maximum suppression

Options

5 Processing mode

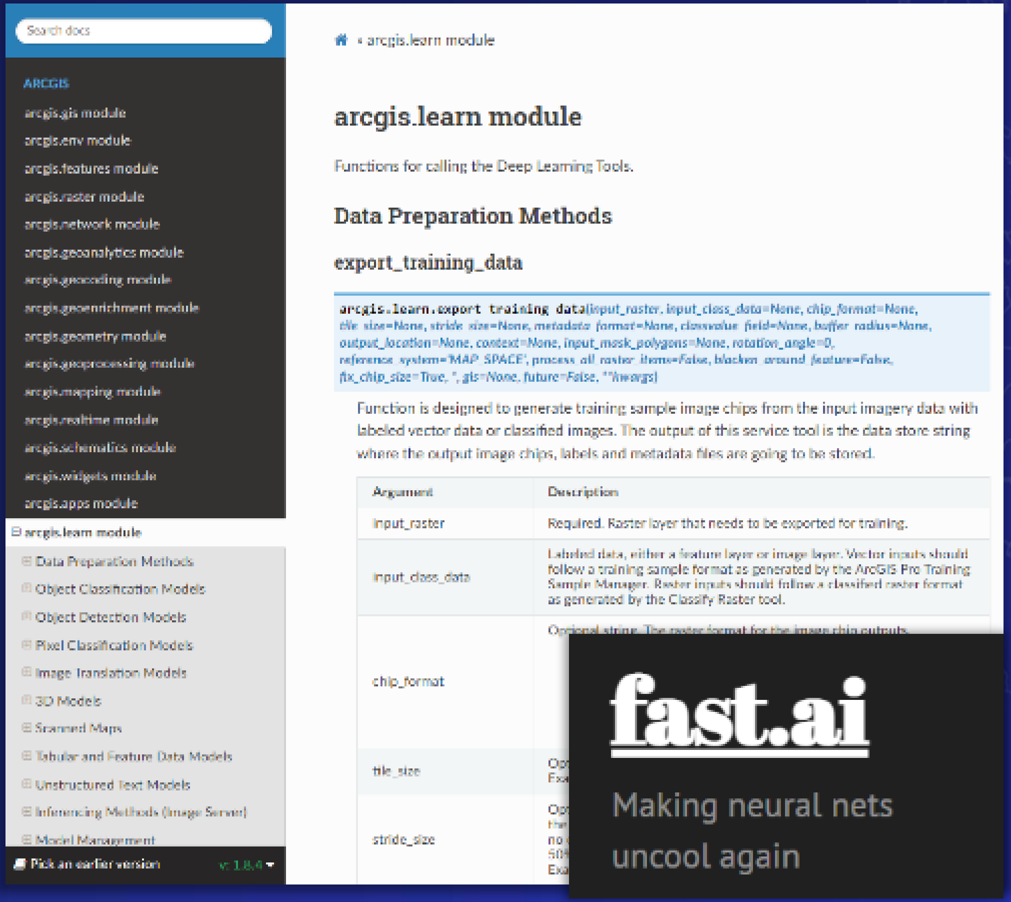
Process as mosaicked image

6 Result layer name

Save result in ImpactObservatory

☑ Use current map extent

## Route II: Python API



### arcgis.learn module

Functions for calling the Deep Learning Tools.

### Data Preparation Methods

#### export_training_data

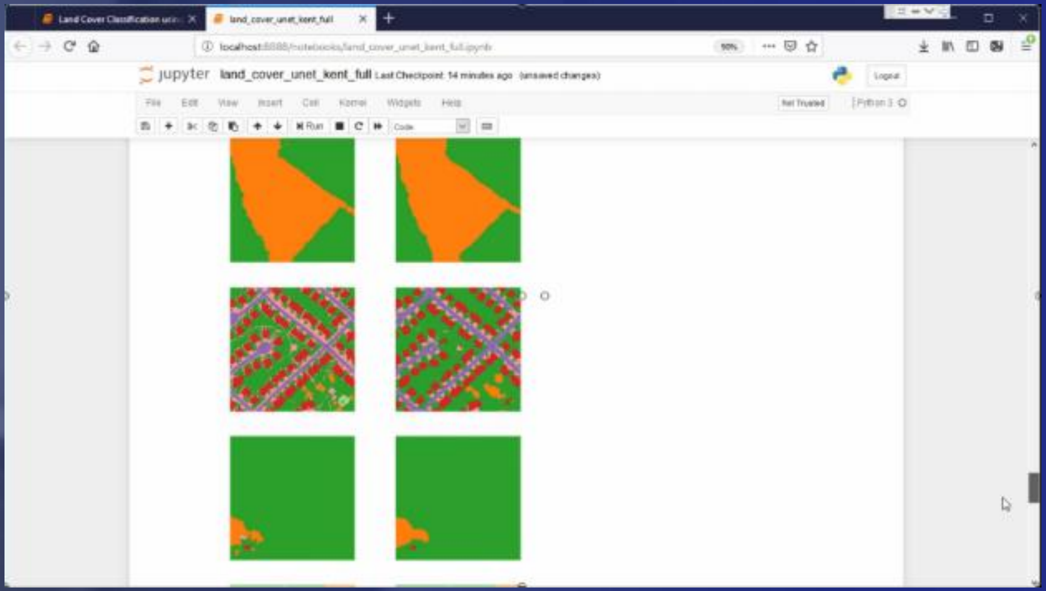
```
arcgis.learn.export_training_data(input_raster, input_class_data=None, chip_format=None,
tile_size=None, stride_size=None, metadata_format=None, classvalue_field=None, buffer_radius=None,
output_location=None, context=None, input_mask_polygons=None, rotation_angle=0,
reference_system='MAP_SPACE', process_all_raster_items=False, blacken_around_feature=False,
fix_chip_size=True, *, gis=None, future=False, **kwargs)
```

Function is designed to generate training sample image chips from the input imagery data with labeled vector data or classified images. The output of this service tool is the data store string where the output image chips, labels and metadata files are going to be stored.

| Argument | Description |
|---|---|
| input_raster | Required. Raster layer that needs to be exported for training. |
| input_class_data | Labeled data, either a feature layer or image layer. Vector inputs should follow a training sample format as generated by the ArcGIS Pro Training Sample Manager. Raster inputs should follow a classified raster format as generated by the Classify Raster tool. |
| chip_format | Optional string. The raster format for the image chip outputs. |
| tile_size | Op<br>Ex |
| stride_size | Op<br>The<br>no<br>50<br>Exa |

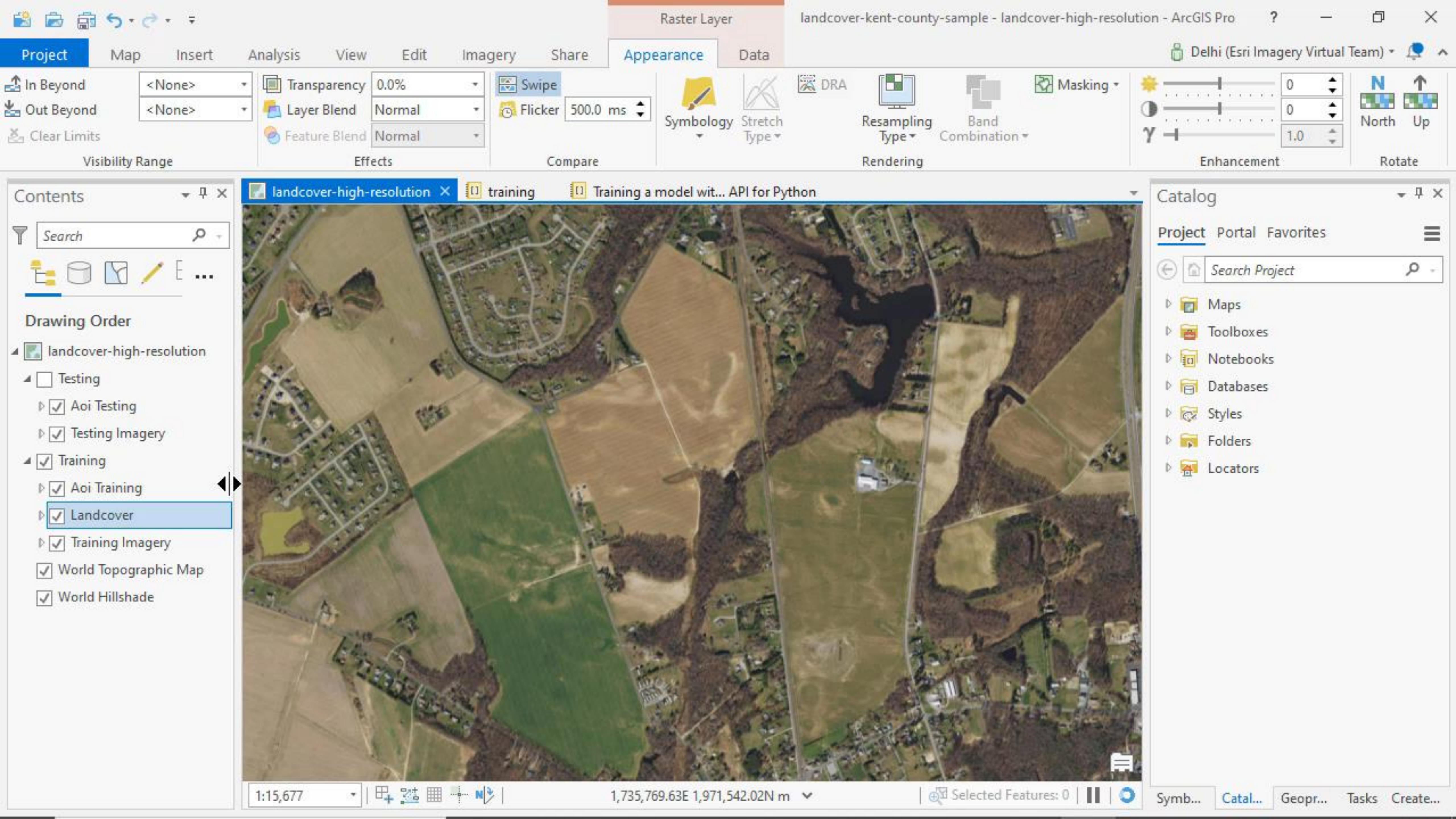
fast.ai

Making neural nets uncool again


Land Cover Classification usin
land_cover_unet_kent_full
localhost:8888/notebooks/land_cover_unet_kent_full.ipynb
jupyter
land_cover_unet_kent_full Last Checkpoint: 14 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

# End-to-end Deep learning in ArcGIS

## - Land Cover Classification



landcover-kent-county-sample - landcover-high-resolution - ArcGIS Pro

Delhi (Esri Imagery Virtual Team)

In Beyond <None>

Out Beyond <None>

Transparency 0.0%

Layer Blend Normal

Feature Blend Normal

Flicker 500.0 ms

Enhancement: 0, 0, 1.0

Drawing Order

- ☑ landcover-high-resolution
  - □ Testing
    - ☑ Aoi Testing
    - ☑ Testing Imagery
  - ☑ Training
    - ☑ Aoi Training
    - ☑ Landcover
    - ☑ Training Imagery
  - ☑ World Topographic Map
  - ☑ World Hillshade

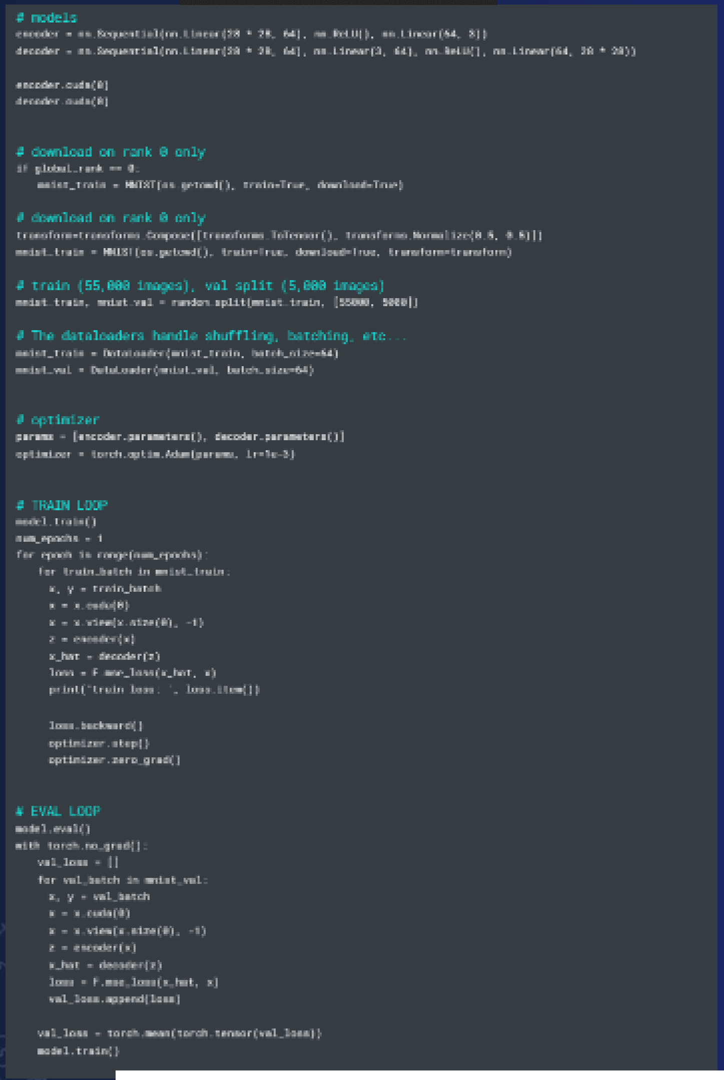
1:15,677

1,735,769.63E 1,971,542.02N m

Selected Features: 0

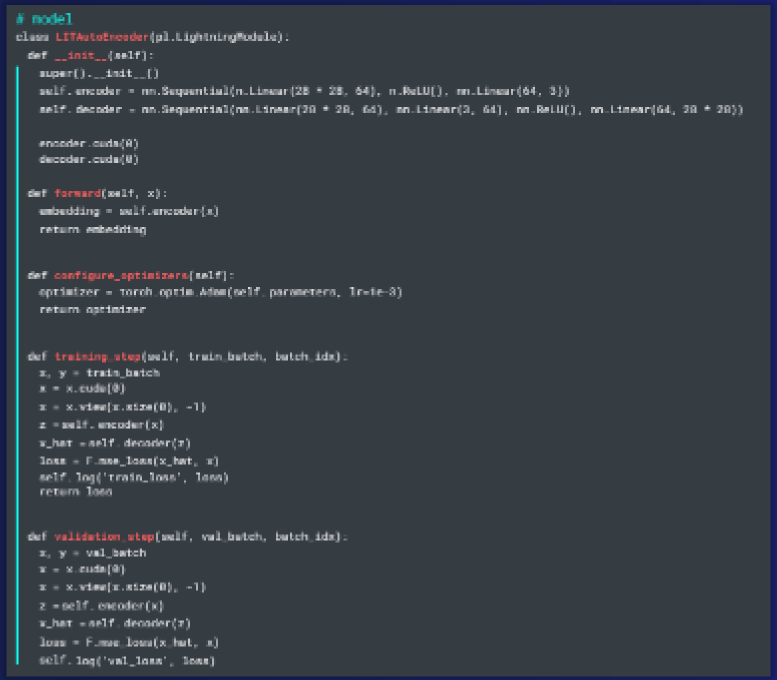
# Whats special about the ArcGIS API for Python

PyTorch



```
# models
encoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.ReLU(), nn.Linear(64, 3))
decoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.Linear(3, 64), nn.ReLU(), nn.Linear(64, 28 * 28))

encoder.cuda(0)
decoder.cuda(0)


# download on rank 0 only
if global_rank == 0:
    mnist_train = MNIST(os.getcwd(), train=True, download=True)

# download on rank 0 only
transform=transforms.Compose([transforms.ToTensor(), transforms.Normalize(0.5, 0.5)])
mnist_train = MNIST(os.getcwd(), train=True, download=True, transform=transform)

# train (55,000 images), val split (5,000 images)
mnist_train, mnist_val = random_split(mnist_train, [55000, 5000])

# The dataloaders handle shuffling, batching, etc...
mnist_train = DataLoader(mnist_train, batch_size=64)
mnist_val = DataLoader(mnist_val, batch_size=64)


# optimizer
params = [encoder.parameters(), decoder.parameters()]
optimizer = torch.optim.Adam(params, lr=1e-3)


# TRAIN LOOP
model.train()
num_epochs = 1
for epoch in range(num_epochs):
    for train_batch in mnist_train:
      x, y = train_batch
      x = x.cuda(0)
      x = x.view(x.size(0), -1)
      z = encoder(x)
      x_hat = decoder(z)
      loss = F.mse_loss(x_hat, x)
      print('train loss: ', loss.item())

      loss.backward()
      optimizer.step()
      optimizer.zero_grad()


# EVAL LOOP
model.eval()
with torch.no_grad():
    val_loss = []
    for val_batch in mnist_val:
      x, y = val_batch
      x = x.cuda(0)
      x = x.view(x.size(0), -1)
      z = encoder(x)
      x_hat = decoder(z)
      loss = F.mse_loss(x_hat, x)
      val_loss.append(loss)

    val_loss = torch.mean(torch.tensor(val_loss))
    model.train()
```

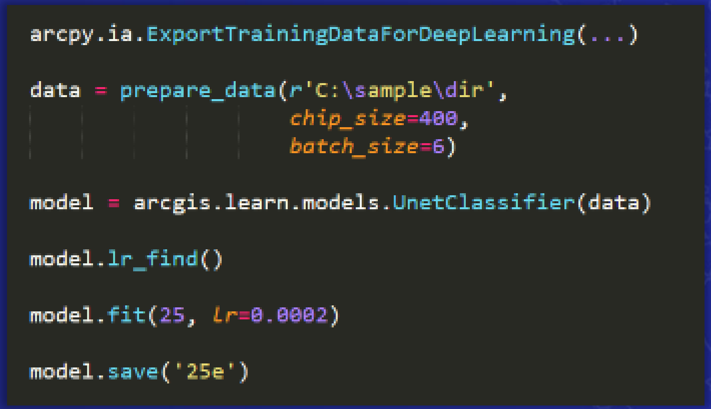
Lightning



```
# model
class LITAutoEncoder(pl.LightningModule):
  def __init__(self):
    super().__init__()
    self.encoder = nn.Sequential(n.Linear(28 * 28, 64), n.ReLU(), nn.Linear(64, 3))
    self.decoder = nn.Sequential(nn.Linear(28 * 28, 64), nn.Linear(3, 64), nn.ReLU(), nn.Linear(64, 28 * 28))

    encoder.cuda(0)
    decoder.cuda(0)

  def forward(self, x):
    embedding = self.encoder(x)
    return embedding


  def configure_optimizers(self):
    optimizer = torch.optim.Adam(self.parameters, lr=1e-3)
    return optimizer


  def training_step(self, train_batch, batch_idx):
    x, y = train_batch
    x = x.cuda(0)
    x = x.view(x.size(0), -1)
    z =self.encoder(x)
    x_hat =self.decoder(z)
    loss = F.mse_loss(x_hat, x)
    self.log('train_loss', loss)
    return loss


  def validation_step(self, val_batch, batch_idx):
    x, y = val_batch
    x = x.cuda(0)
    x = x.view(x.size(0), -1)
    z =self.encoder(x)
    x_hat =self.decoder(z)
    loss = F.mse_loss(x_hat, x)
    self.log('val_loss', loss)
```

fast.ai

arcgis.learn

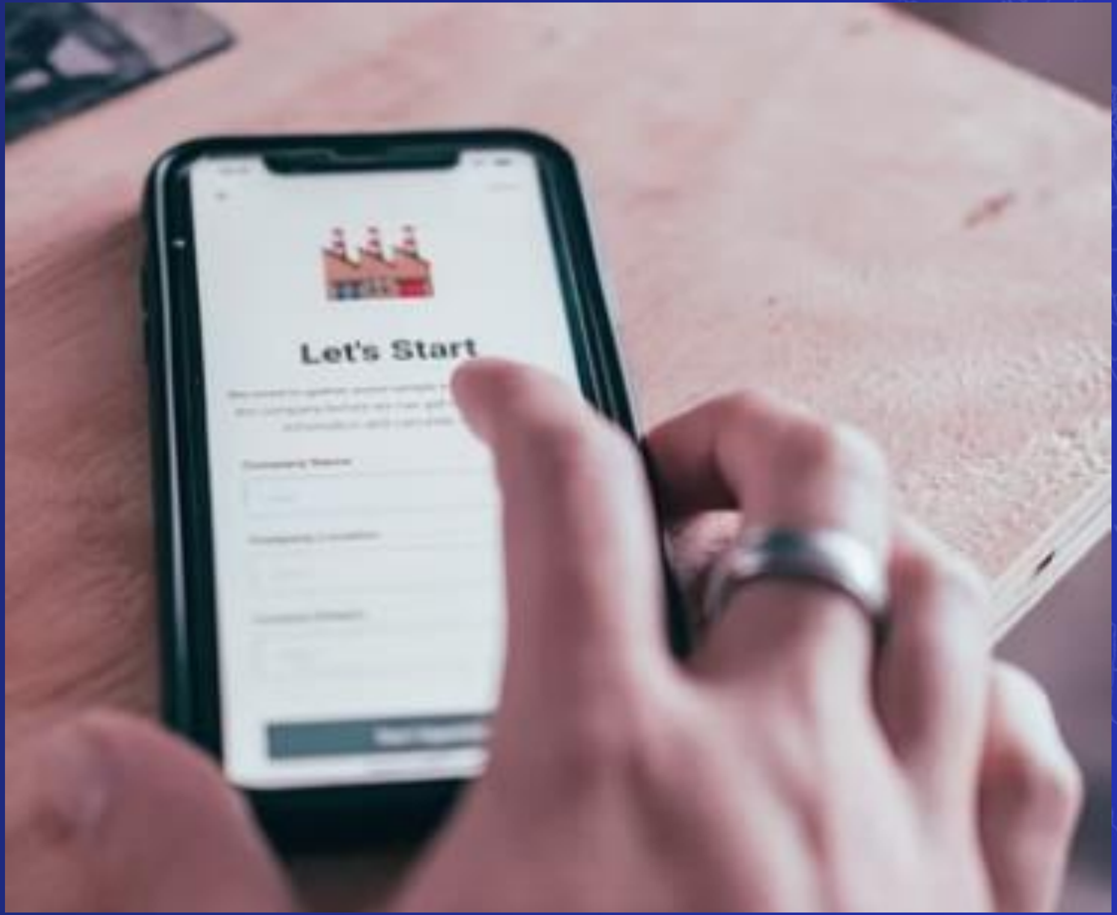


```
arcpy.ia.ExportTrainingDataForDeepLearning(...)

data = prepare_data(r'C:\sample\dir',
                    chip_size=400,
                    batch_size=6)

model = arcgis.learn.models.UnetClassifier(data)

model.lr_find()

model.fit(25, lr=0.0002)

model.save('25e')
```

Increasing Abstraction

# Getting Started

- **Option 1: Use pre-trained models**
- **Option 2: Train your own model**

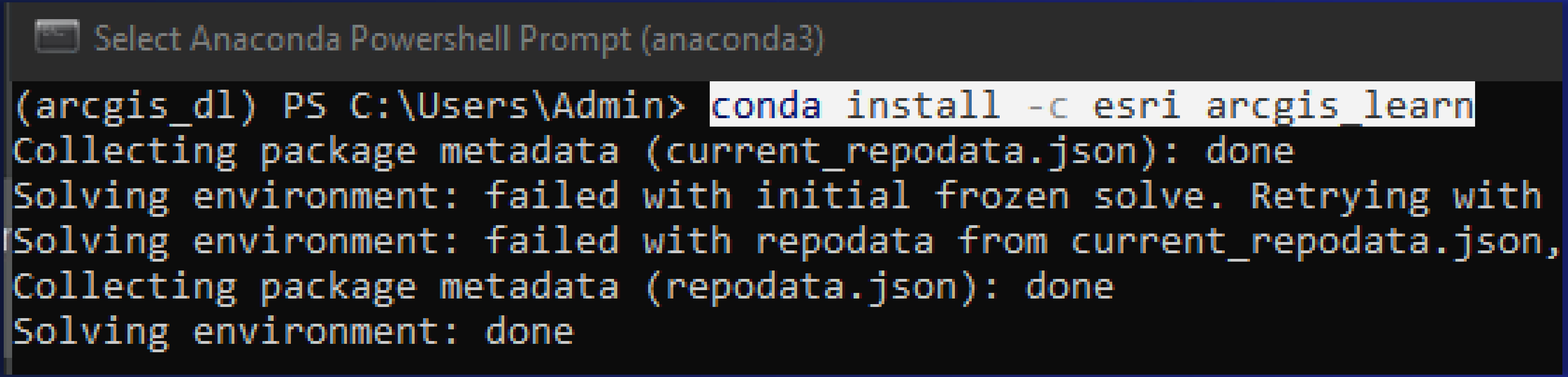


# Easy installation of deep learning libraries

- Deep Learning Libraries installer (ArcGIS Pro and Enterprise)
- arcgis_learn conda metapackage (Anaconda)
- arcgis_dl_backbones metapackage (for disconnected users)

Deep Learning Libraries Installers for ArcGIS

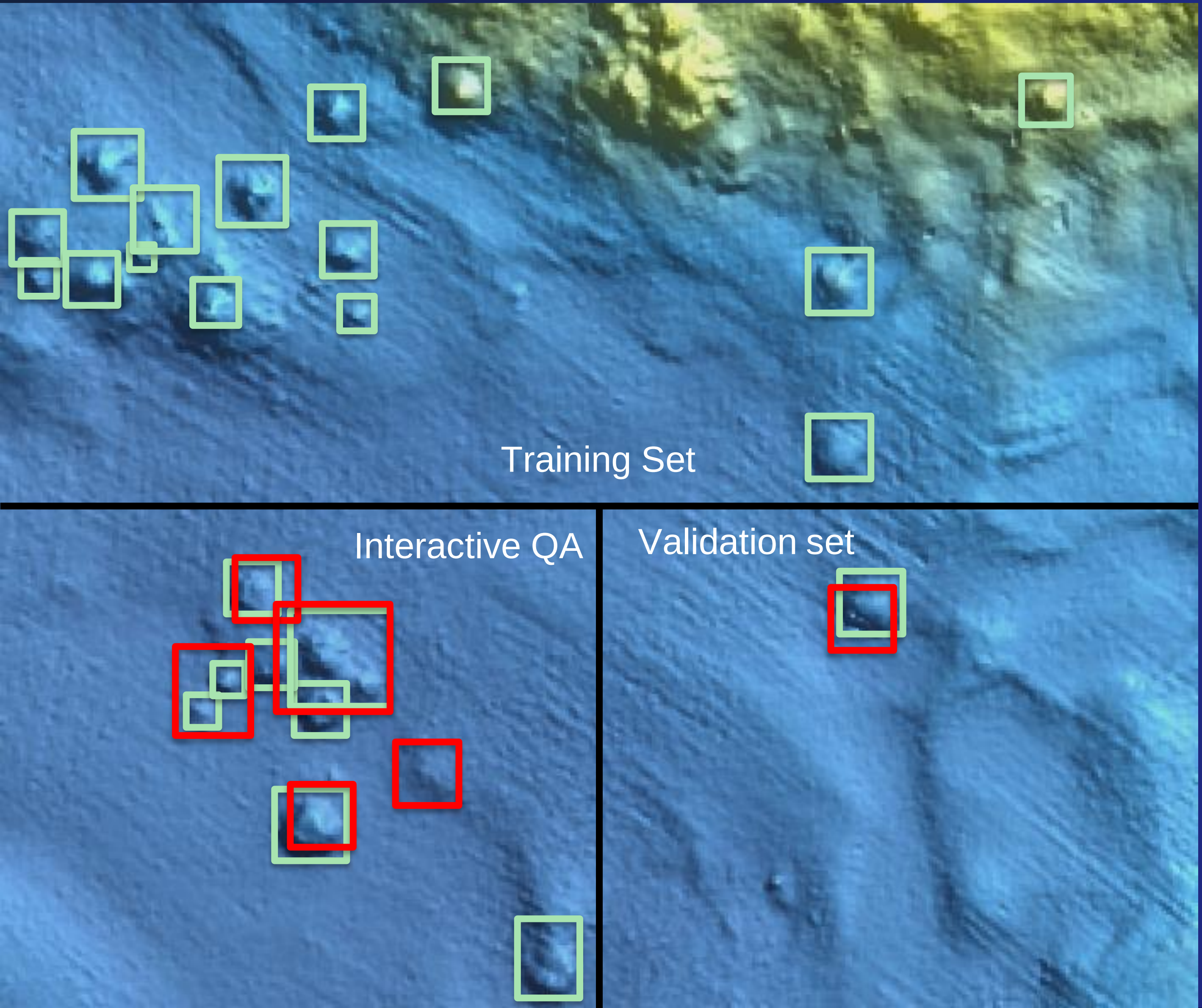
PyTorch
scikit-image
scikit learn
cuDNN
Keras
plotly
DASK
ONNX
fast.ai
TensorFlow
spaCy

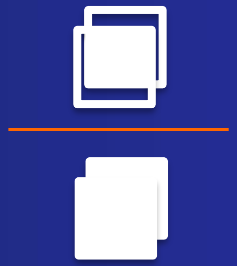


```
(arcgis_dl) PS C:\Users\Admin> conda install -c esri arcgis_learn
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with
Solving environment: failed with repodata from current_repodata.json,
Collecting package metadata (repodata.json): done
Solving environment: done
```

# Deep Learning Concepts in Detail

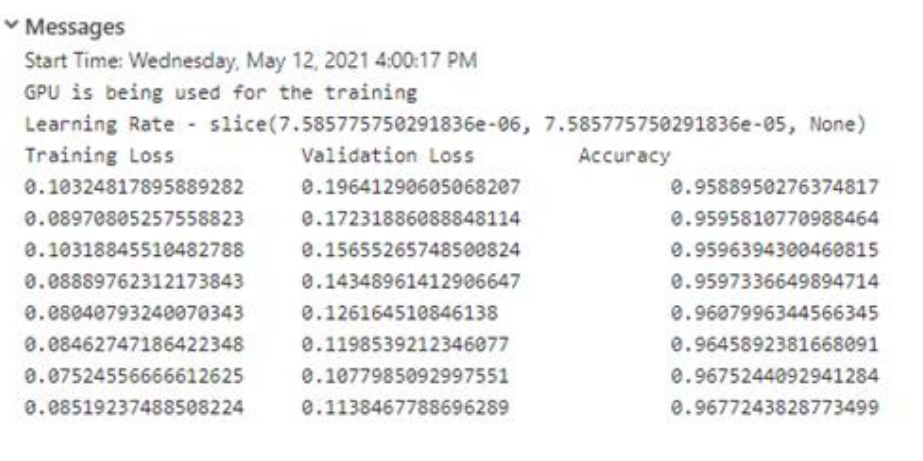
# Splitting data for Training, Validation and QA





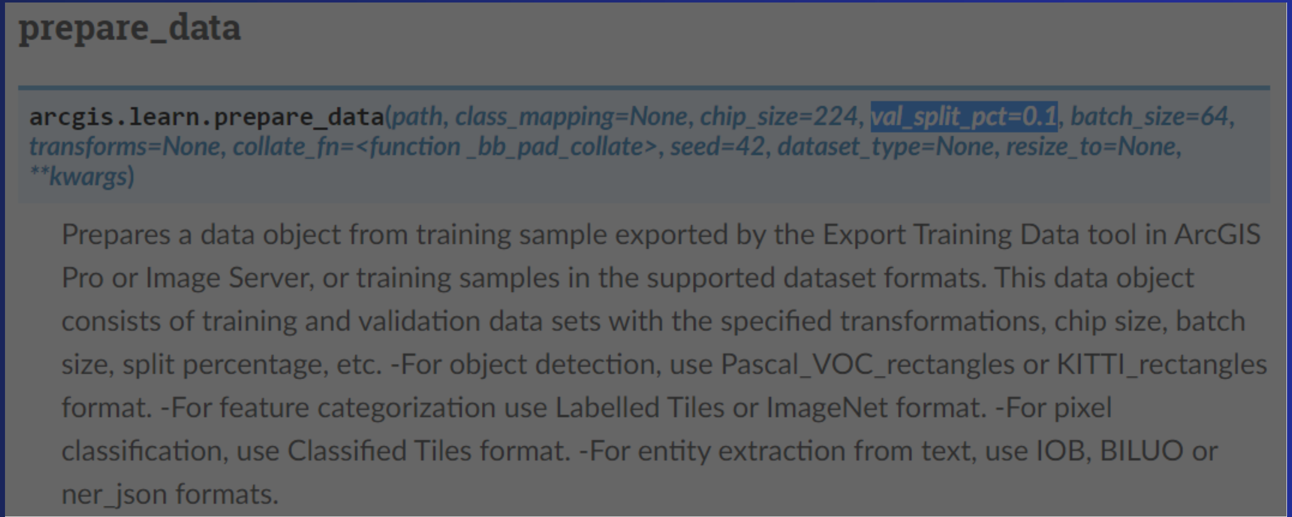
IOU (Intersect over Union)
= Error/Loss

Monitor model status in
‘Train Deep Learning Model’ tool



```
˅ Messages
  Start Time: Wednesday, May 12, 2021 4:00:17 PM
  GPU is being used for the training
  Learning Rate - slice(7.585775750291836e-06, 7.585775750291836e-05, None)
  Training Loss          Validation Loss         Accuracy
  0.10324817895889282    0.19641290605068207          0.9588950276374817
  0.08970805257558823    0.17231886088848114          0.9595810770988464
  0.10318845510482788    0.15655265748500824          0.9596394300460815
  0.08889762312173843    0.14348961412906647          0.9597336649894714
  0.08040793240070343    0.126164510846138            0.9607996344566345
  0.08462747186422348    0.1198539212346077           0.9645892381668091
  0.07524556666612625    0.1077985092997551           0.9675244092941284
  0.08519237488508224    0.1138467788696289           0.9677243828773499
```

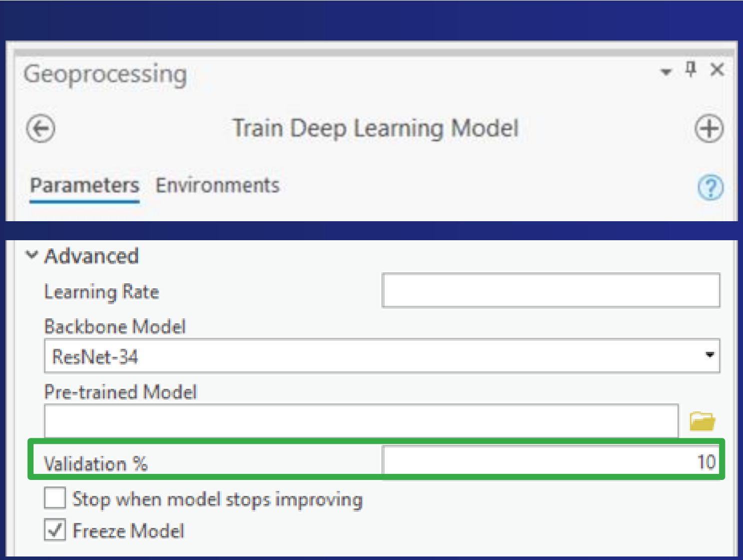
# Splitting data for Training, Validation



## prepare_data

```
arcgis.learn.prepare_data(path, class_mapping=None, chip_size=224, val_split_pct=0.1, batch_size=64, transforms=None, collate_fn=<function _bb_pad_collate>, seed=42, dataset_type=None, resize_to=None, **kwargs)
```

Prepares a data object from training sample exported by the Export Training Data tool in ArcGIS Pro or Image Server, or training samples in the supported dataset formats. This data object consists of training and validation data sets with the specified transformations, chip size, batch size, split percentage, etc. -For object detection, use Pascal_VOC_rectangles or KITTI_rectangles format. -For feature categorization use Labelled Tiles or ImageNet format. -For pixel classification, use Classified Tiles format. -For entity extraction from text, use IOB, BILUO or ner_json formats.



Advanced

Learning Rate

Backbone Model: ResNet-34

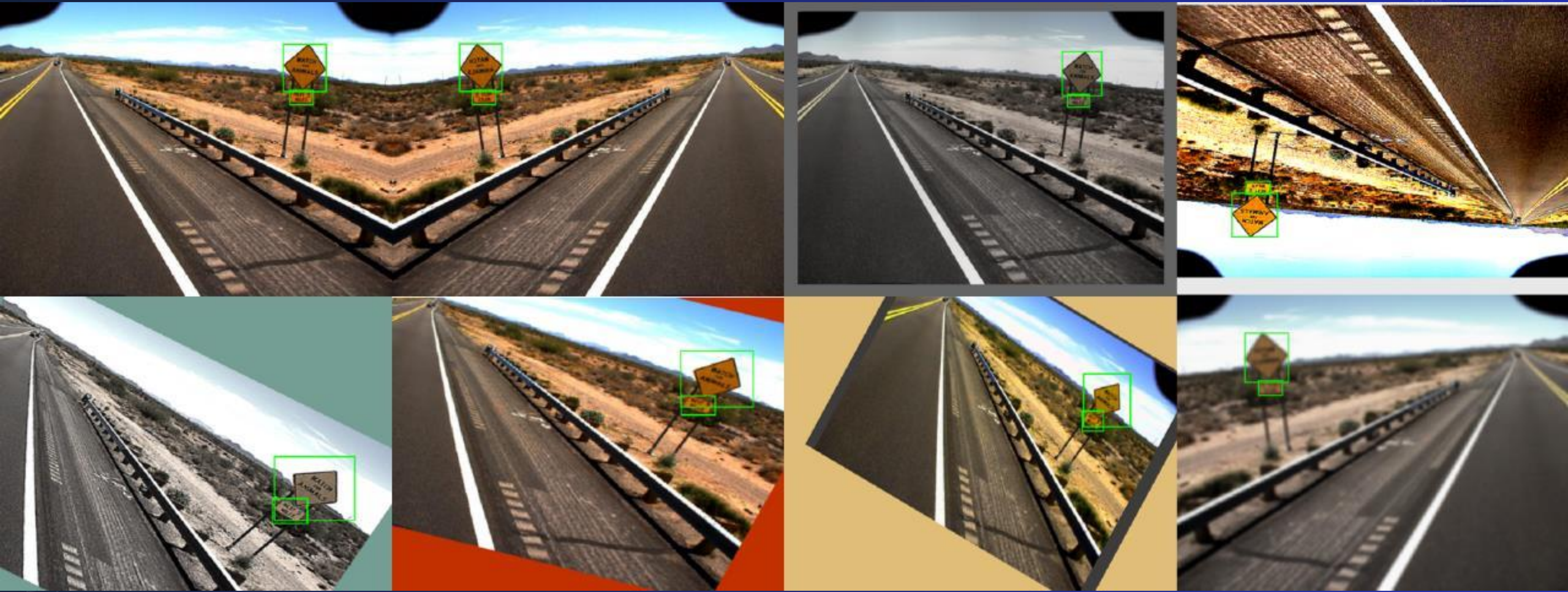
Pre-trained Model

Validation % 10

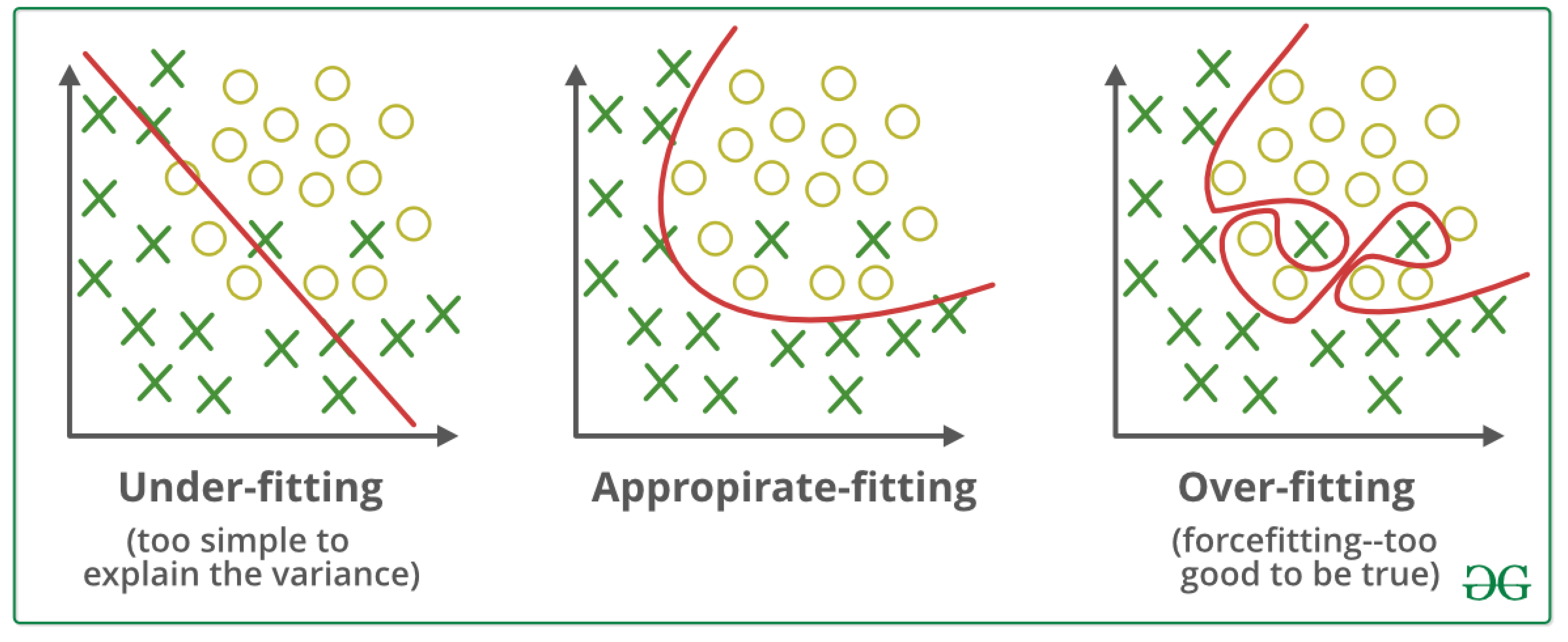
□ Stop when model stops improving

☑ Freeze Model

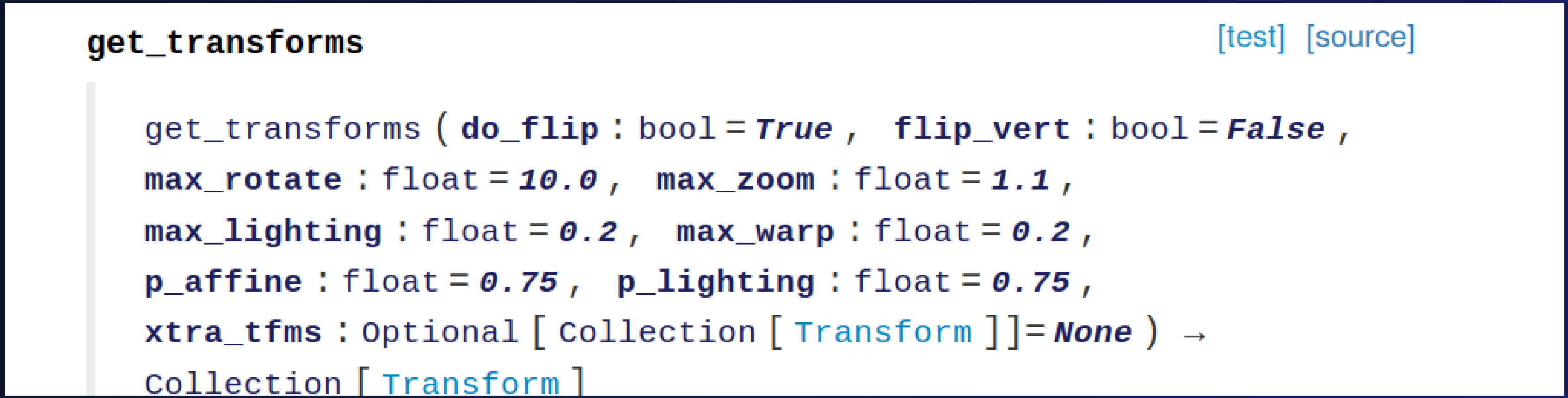
# Examples of Data Transformations



# Why Do We Need Data Transformations



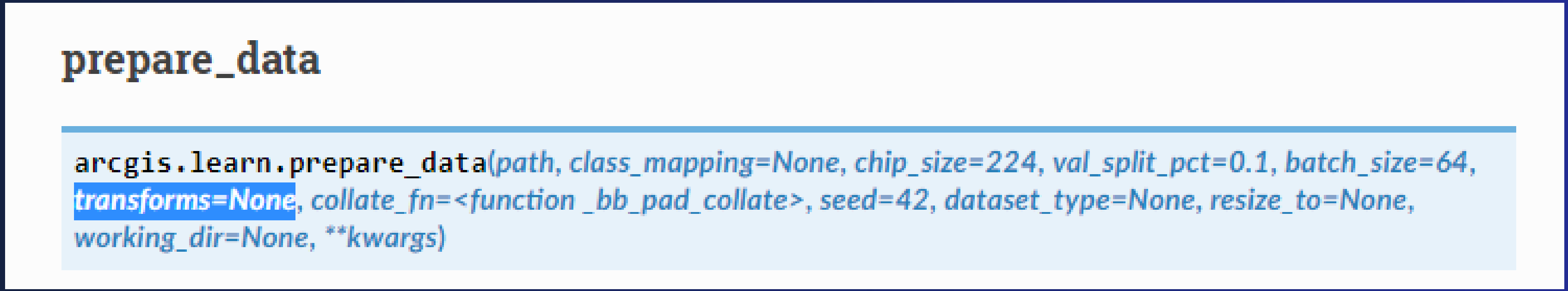
# How to Apply Data Transformations



**get_transforms** [test] [source]

```
get_transforms(do_flip:bool=True, flip_vert:bool=False,
max_rotate:float=10.0, max_zoom:float=1.1,
max_lighting:float=0.2, max_warp:float=0.2,
p_affine:float=0.75, p_lighting:float=0.75,
xtra_tfms:Optional[Collection[Transform]]=None) →
Collection[Transform]
```

Default fast.ai transforms, can try printing this Collection for default arcgis.learn transforms.

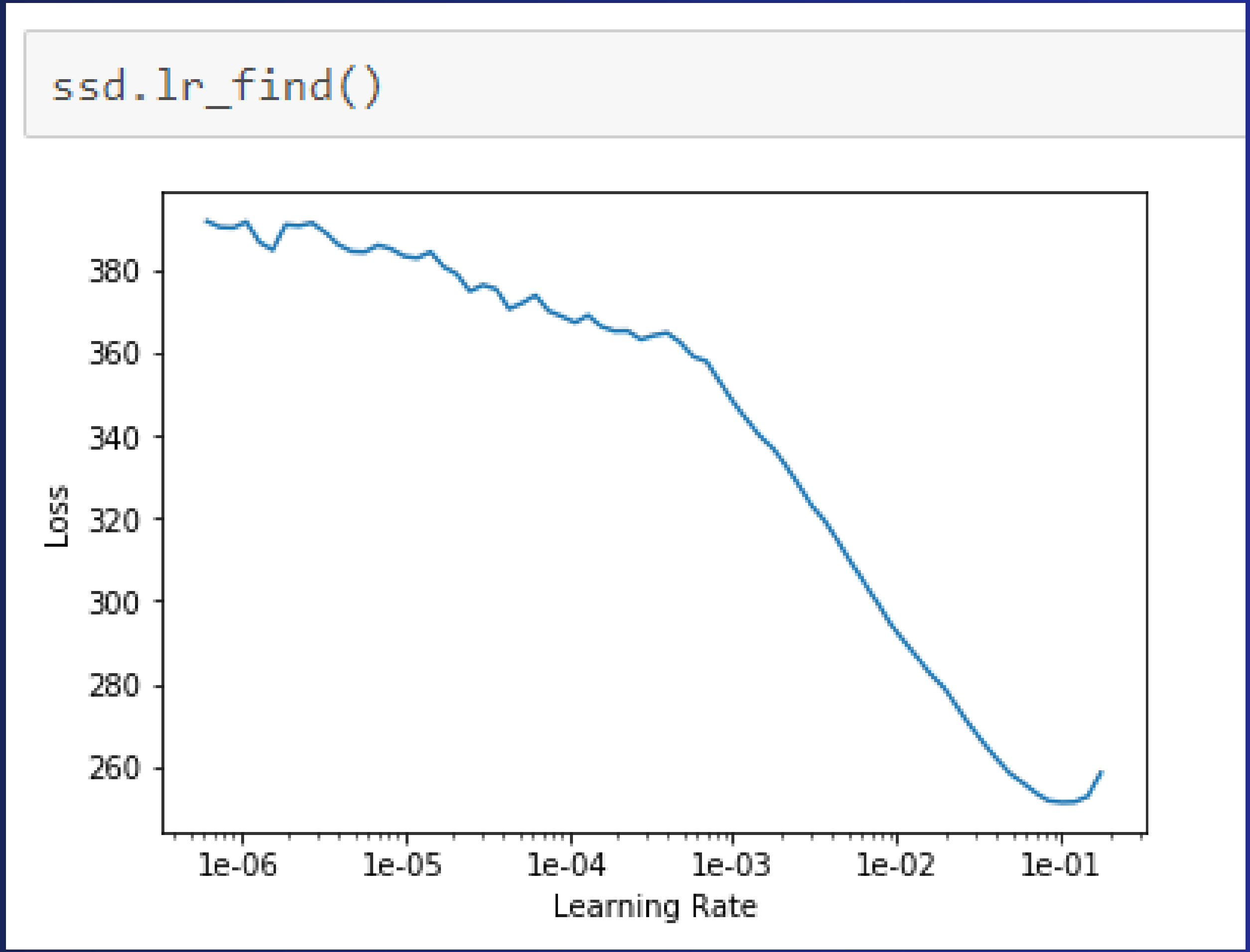


**prepare_data**

```
arcgis.learn.prepare_data(path, class_mapping=None, chip_size=224, val_split_pct=0.1, batch_size=64,
transforms=None, collate_fn=<function _bb_pad_collate>, seed=42, dataset_type=None, resize_to=None,
working_dir=None, **kwargs)
```

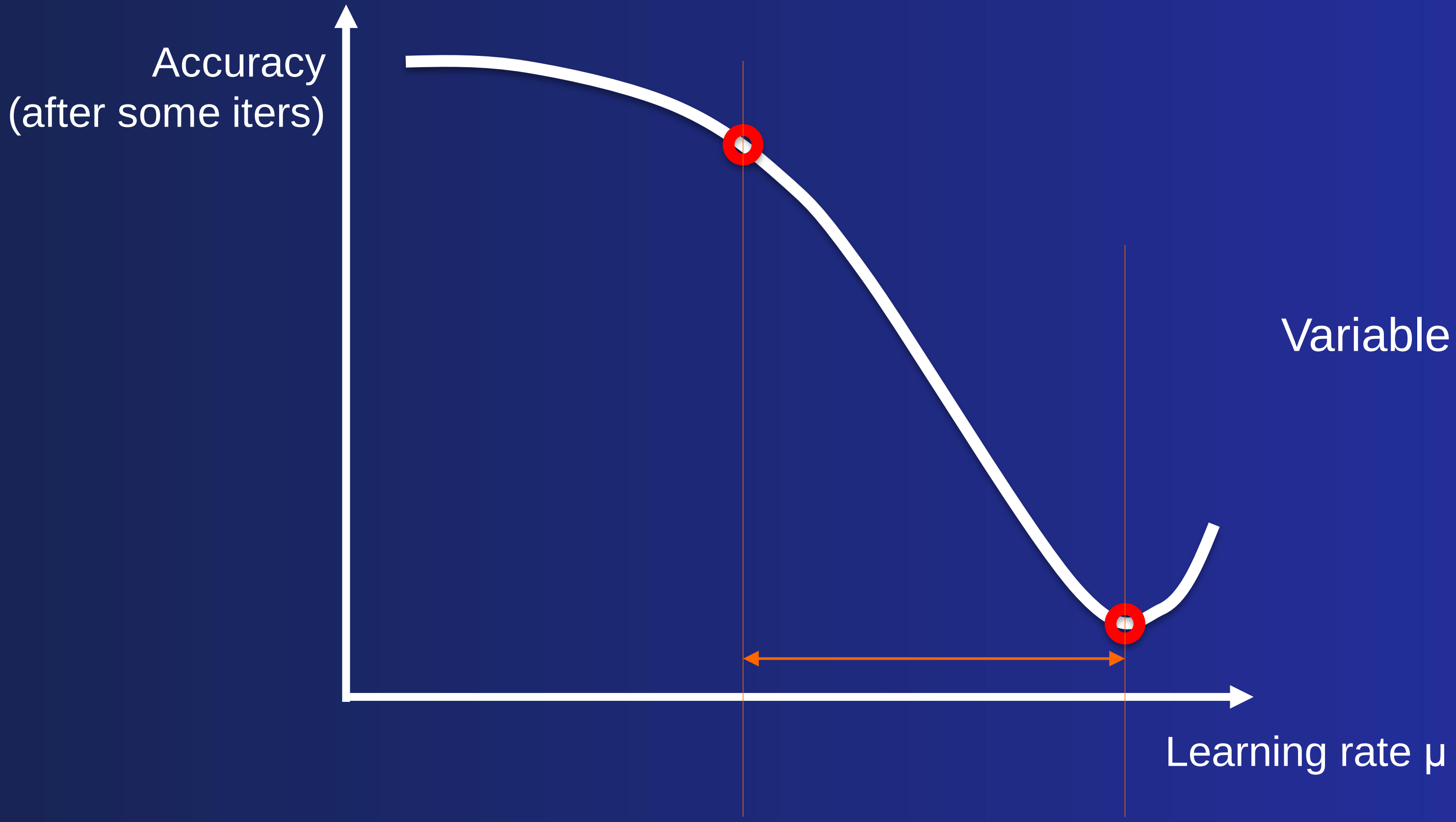
# Before You Train: Learning Rate Finder

https://arxiv.org/abs/1708.07120



```
ssd.lr_find()
```

# Gradient Descent



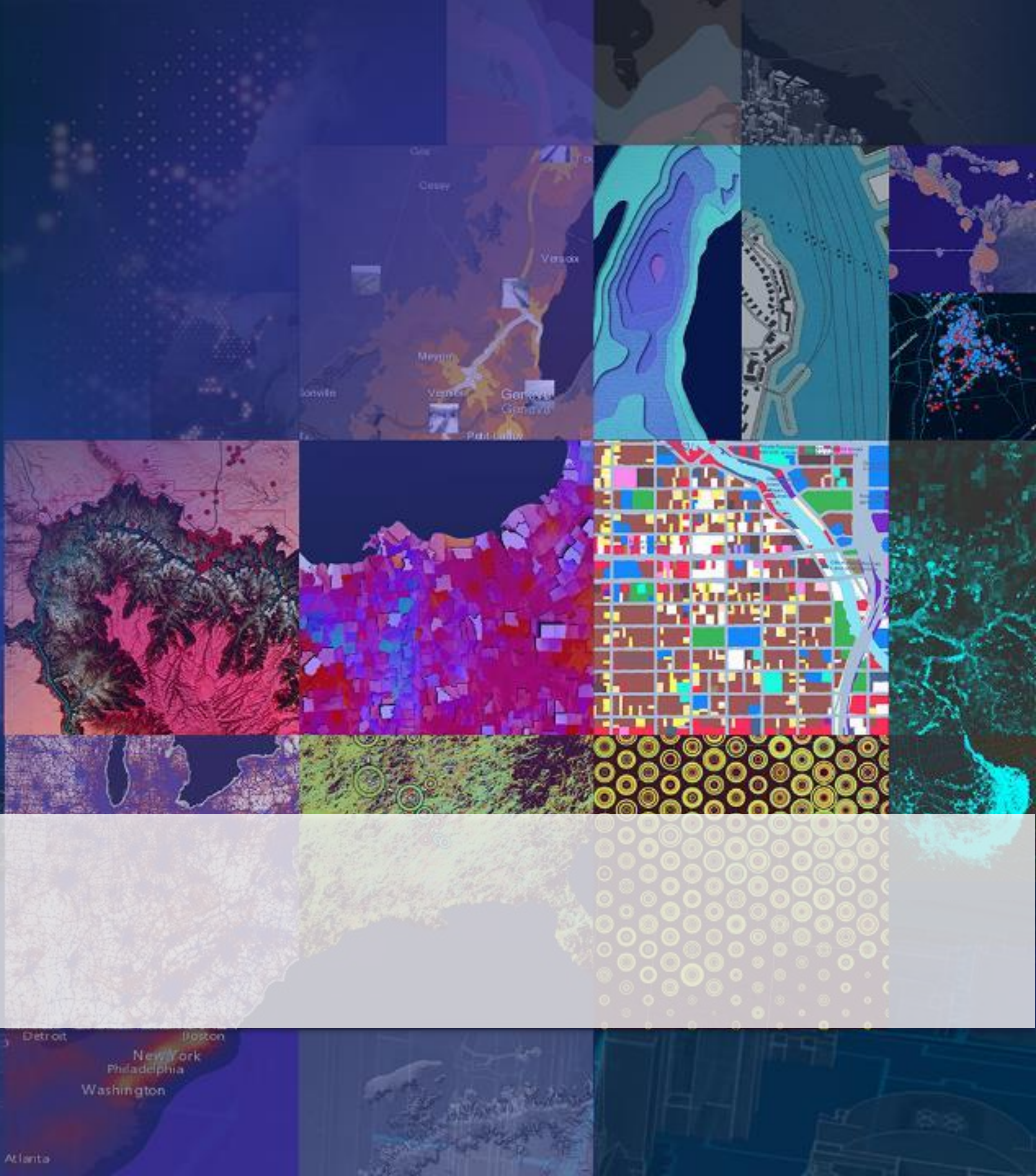
Variable Learning rate μ

# Model Training, Saving and Loading

```
model = FeatureClassifier(data, backbone='MobileNetV2', backend='tensorflow’)

model.fit(20, lr=lr)

model.save('20e_nofpn’)

model.load('20e_nofpn’)  # Must have the same model defined!

model.fit(25, lr=lr)

model.save('Plant-identification-25-tflite', framework="tflite")  # by default ‘PyTorch’



model2 = FeatureClassifier.from_emd(data, ‘20e_nofpn’)
```

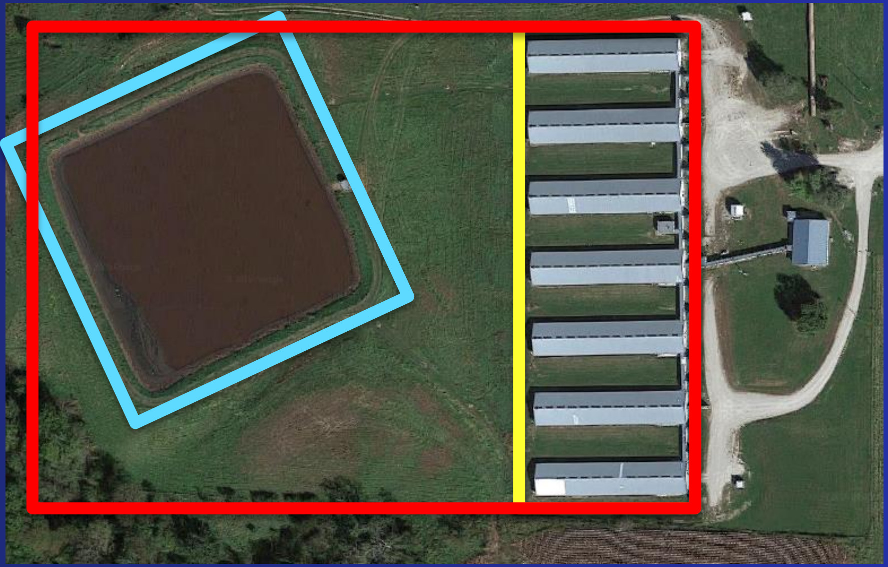
# Few Best Practices

+ FAQ



# Preparing an ideal training dataset

- Need a balance of classes
- Label Accurately including context
- Apply image augmentation
- Size of chips >= 400 px

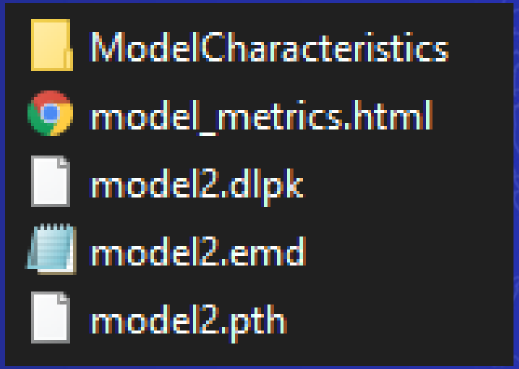


**Universal question - How many image chips do I need?**

Ideally between 400 and 40,000 chips !

# Am I certain that the model is “good”?

- Test the model over multiple regions with large variability.
- Visually inspect the inference results

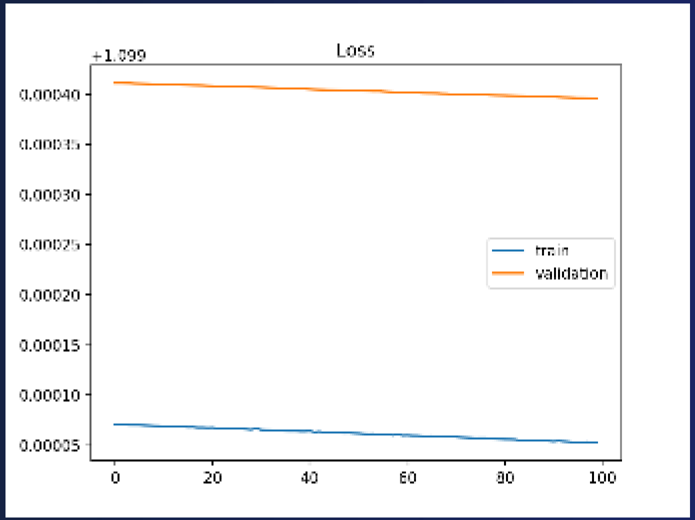


ModelCharacteristics
model_metrics.html
model2.dlpk
model2.emd
model2.pth

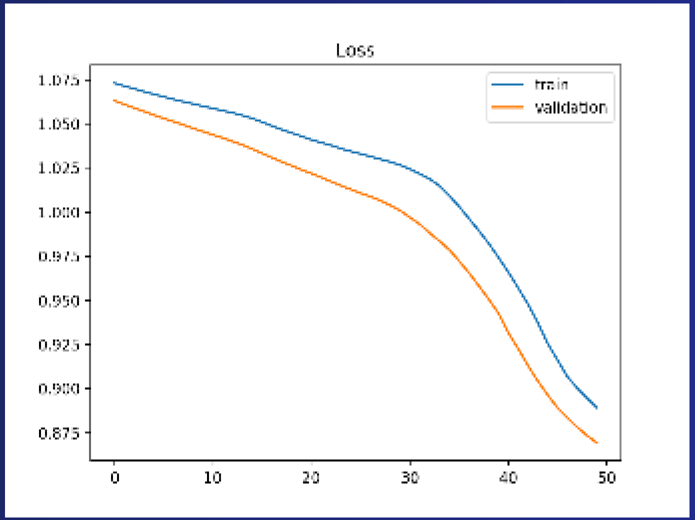
**Look at the training metrics**

- Confusion matrix, validation accuracy
- Avoid “standardized indicators”
  - Only useful when training happens over a known benchmark
  - Or if quantitative comparisons need to be made between different model iterations
  - They do not tell you how “well” a model will perform in practice

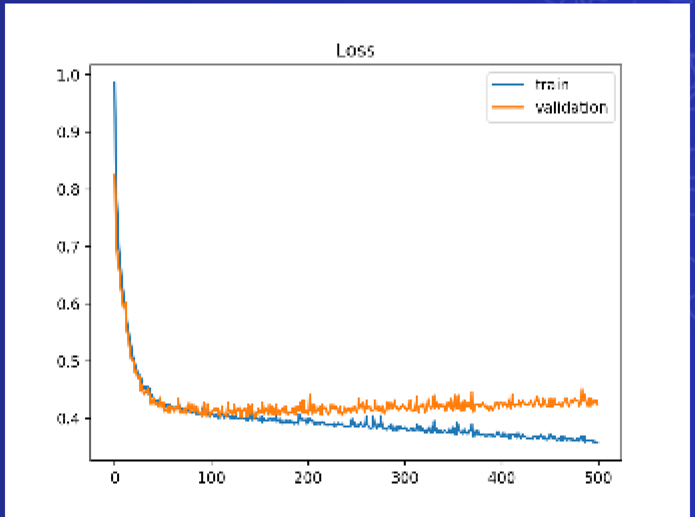
# How to interpret training/validation loss curves?



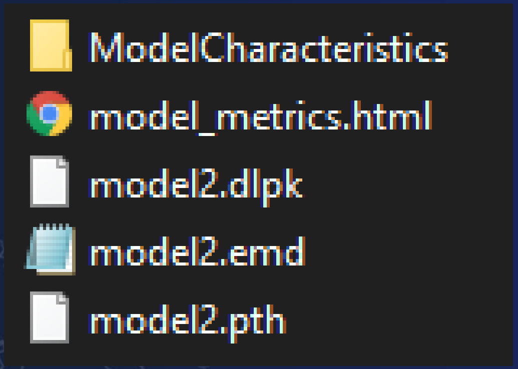
Model not complex enough – Try to increase the backbone size.



Both train/valid losses are decreasing, but have not yet converged – keep training.



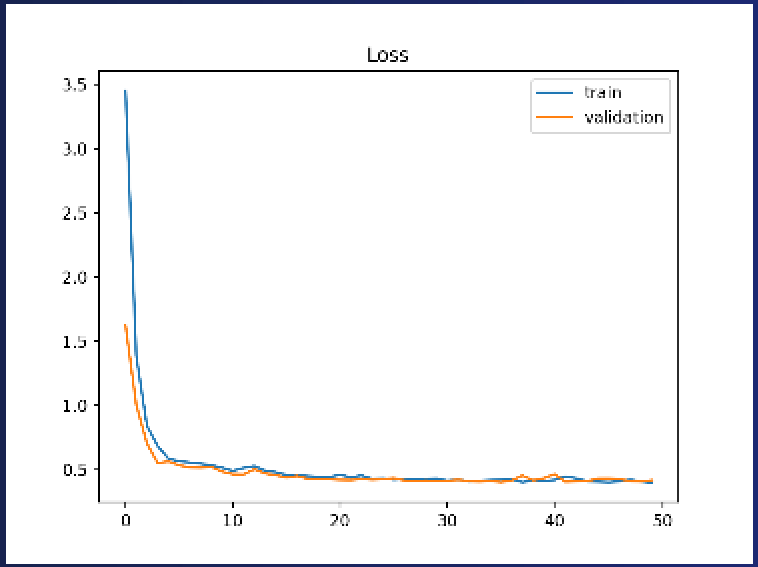
Model has overfit – implement early stopping.



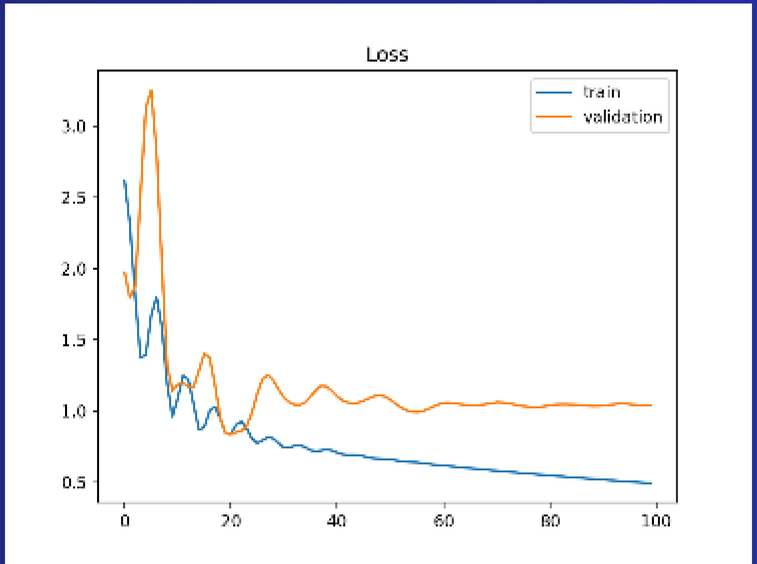
ModelCharacteristics
model_metrics.html
model2.dlpk
model2.emd
model2.pth

https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

# How to interpret training/validation loss curves?



Ideal model behavior, but converged a while ago – implement early stopping.



Loss oscillating wildly – try a smaller learning rate.

https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

# Imagery and Hardware Considerations

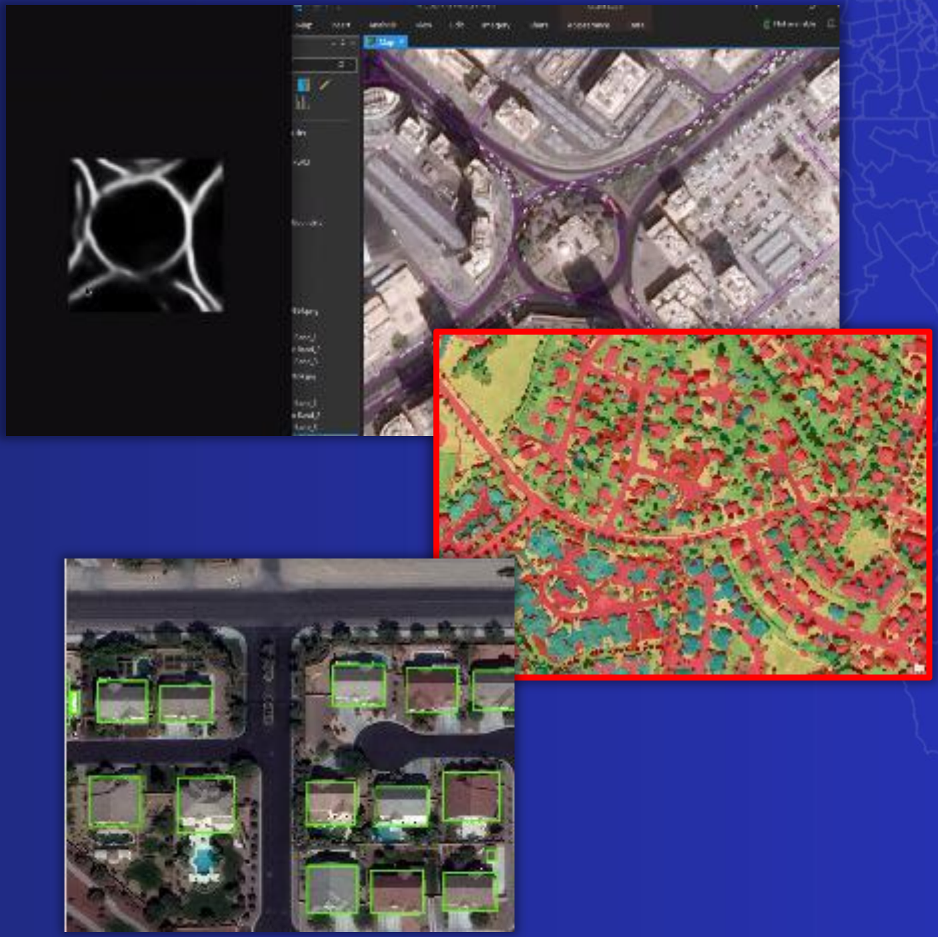
- Confirm your imagery is suitable (Deep Learning in not magic!)
  - Identifiable objects
  - Image properties match the model requirements
- 8-bit 3 band imagery is no longer a limitation
- Multispectral imagery inputs supported on Pro 2.7

- Recommended desktop GPUs:
  - RTX2XXX, RTX3XXX, P4000, GV100….
- Cloud GPUs:
  - T4, V100
- AWS instances:
  - G4, P3
- Azure instances:
  - ND6s

# Key takeaways

ArcGIS has powerful deep learning capabilities!

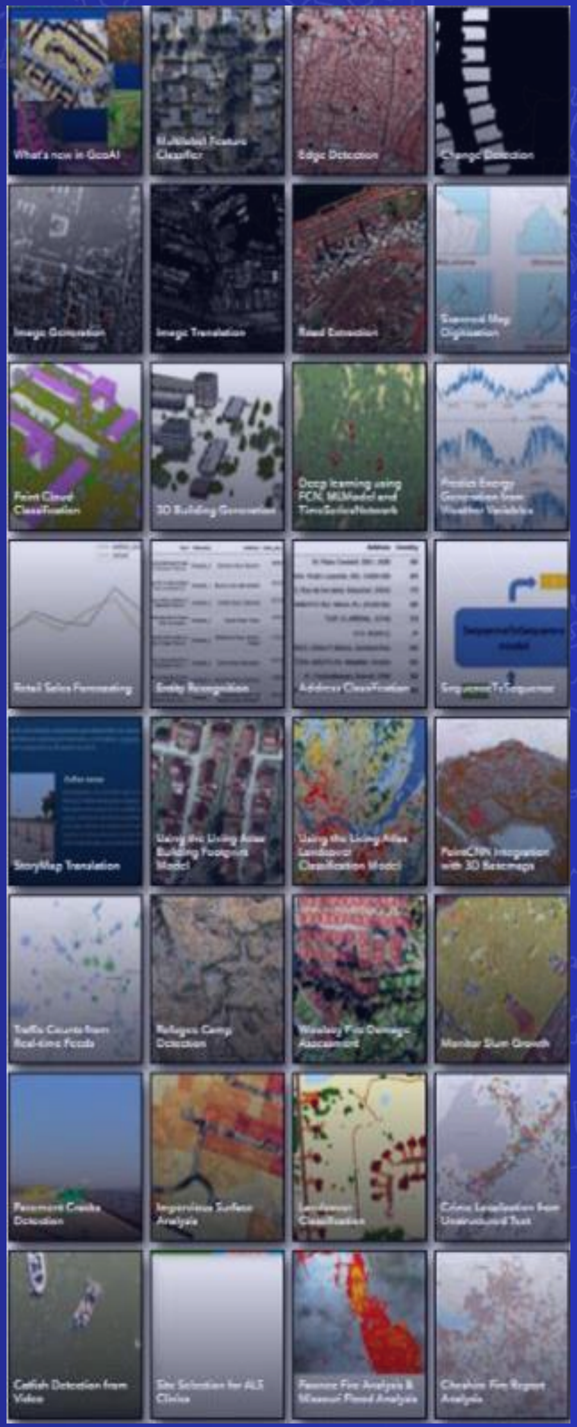
- Suite of tools for Deep Learning workflows
- Powerful API for developers and data scientists
- Variety of clients
- Support all imagery categories/tasks and more..
- Massively scalable
- Robust image management to complement deep learning
- Rich tools to perform down stream analysis
- Pre-trained AI models



# Resources

We will send the following resources:

- DL Libraries Installer: https://github.com/Esri/deep-learning-frameworks
- Sample Notebooks: https://developers.arcgis.com/python/sample-notebooks/
- Esri Community: https://community.esri.com/
- GitHub Repo: https://github.com/ESRI/arcgis-python-api
- GeoAI Hub Demo Resources: https://demos-geoai.hub.arcgis.com/notebooks/
- GeoAI Medium (Technical Blogs): https://medium.com/geoai
- Ready-to-use Geospatial Deep Learning Models (blog)


What's new in GeoAI
Multilabel Feature Classifier
Edge Detection
Change Detection
Image Generation
Image Translation
Road Extraction
Scanned Map Digitization
Point Cloud Classification
3D Building Generation
Deep learning using FCN, MLModel and TimeSeriesNetwork
Predict Energy Generation from Weather Variables
Retail Sales Forecasting
Entity Recognition
Address Classification
SequenceToSequence
StoryMap Translation
Using the Living Atlas Building Footprint Model
Using the Living Atlas Landcover Classification Model
PointCNN integration with 3D Basemaps
Traffic Counts from Real-time Feeds
Refugee Camp Detection
Woolsey Fire Damage Assessment
Monitor Slum Growth
Pavement Cracks Detection
Impervious Surface Analysis
Landcover Classification
Crime Localization from Unstructured Text
Catfish Detection from Video
Site Selection for ALS Clinics
Pawnee Fire Analysis & Missouri Flood Analysis
Cheshire Fire Report Analysis

Thank you and please fill in the exit survey

esri | THE SCIENCE OF WHERE

To connect, please reach out to Canserina at:

ckurnia@esri.com

Copyright © 2020 Esri. All rights reserved.