Contact Information and Introductions

JP Metcalf
Solutions Engineer
jp.metcalf@l3harris.com

Zachary Norman
Product Manager
zachary.norman@l3harris.com
Agenda

ENVI Deep Learning Overview
What is Object Detection?
ENVI Deep Learning: Object Detection Workflow
Examples and Use Cases
Object Detection vs. Pixel Classification
Questions and Discussion
ENVI Deep Learning Recap

Applied deep learning for geospatial imagery in ENVI, the leading remote sensing and image analysis software.

Without needing to program, the capabilities include:

- Segmentation (i.e. cloud masking)
- Linear feature extraction (i.e. roads)
- Support for nearly any image format and data modality

Deep learning workflow in ENVI, built on TensorFlow and Keras.

Assess building damage after hurricanes and tornadoes.

Automated flood detection using SAR.
Hardware and Software Requirements

**ENVI 5.6.1 is needed for DL 1.2**

**NVIDIA GPU card with CUDA® Compute Capability 3.5 or higher**


CUDA 11 is required for the latest version of ENVI Deep Learning

Minimum 8GB of GPU RAM recommended
What is Object Detection?

**ENVI Deep Learning 1.2**

Returns geometry (bounding box) for features and detections are allowed to overlap.

**ENVI Deep Learning**

Pixels are classified, post processing can extract features if they don’t touch.

**Not in ENVI Deep Learning**

Returns geometry (polygon) for features and detections are allowed to overlap.

Image source: https://medium.com/onepanel/instance-segmentation-with-mask-r-cnn-and-tensorflow-on-onepanel-6a072a4273dd
ENVI Deep Learning 1.2 Toolbox

Existing capabilities have been moved under “Pixel Segmentation”

Several tools updated for object detection

New tools for object detection
When you create a new project using the Deep Learning Labeling Tool, there is a new setting to indicate whether you are labeling data for “Pixel Classification” or “Object Detection”.

A short video showing the Labeling Tool being used to identify ships for an object detection model.
Training: Pixel Classification vs Object Detection

Pixel Classification (UNet)

Object Detection
Object Detection Training Parameters

Model creation is a part of the training process and no longer requires an extra step.

Pre-trained weights are used for all models.

For features/labels less than 25 pixels in size, increase size to 25 pixels.

Augmentation during training is the same as before.

Patches per batch is *not* auto-calculated as before.

You can use 4-5 for a GPU with 8 GB or RAM.

Training and validation data is the same as before.

Parameters to control training.

For object detection, you don’t specify the number of training steps as this is done for you based on your training data.

This means object detectors train using *all* of your labels.
TensorBoard Integration

TensorBoard is a tool for visualizing the performance of your neural networks during and after training.

Logs for object detection are reset each training session.

Only loss is reported for object detection.

Loss around 0.2 indicates a model is converging well.
Classification results:

- Provided as a vector (shapefile) with class information and confidence (probability)
- When displayed in ENVI, features will have their class color applied
- Shapefiles are used for all outputs, even when images are not georeferenced
Examining Detections

Fun fact: when you are viewing the outputs of object detection in ENVI, you can colorize the vector data by the confidence of detects.

This is a great tool to quickly gauge the performance of models and see where low-probability detects are found.
Cleaning Up Results

Once you have classified an image, there are two options to cleanup results:

1. Use the “Postprocess Classification Vector” to filter detections based on confidence, IoU, and optionally intersection.

2. Within ENVI, you can use the vector editing tool to add or remove features.

Before and after results using the new “Postprocess Classification Vector” task.
Easy-to-use ENVI API

Custom Data Preprocessing

```plaintext
def process and generate labels:
    for file, process, fIdx do begin
        print, fIdx
        :build the output filename
        outRaster = file.replace('_metadata.xml', '_labels.dat')
        if file_test(outRaster) then begin
            labelRasters.add, e.openRaster(outRaster)
            continue
        endif
        :get the ROI file
        roiFile = labels[fIdx]
        :open everything
        roi = e.openROI(roiFile)
        rasters = e.openRaster(file)
        :normalize
        norm = ENVILinearRangeStretchRaster(rasters[1], MIN=0, MAX=10000)
        :build the label raster
        task = ENVIENVIObjectModelTask('BuildLabelRasterFromROI')
        task.INPUT_RASTER = norm
        task.INPUT ROI = roi
        task.OUTPUT_RASTER URI = outRaster
        task.execute
        :clean up and save
        norm.close
        foreach r, rasters do r.close
        foreach r, roi do r.close
        labelRasters.add, e.openRaster(outRaster)
    endforeach
```

Classifier Generation

```plaintext
; create our task
task = ENVITask('TrainTensorflowObjectModel')

; specify training time
task.EPOCHS = epochs
task.FEATURE_PATCH_PERCENTAGE = dataPerEpoch
task.BACKGROUND_PATCH_RATIO = bpr

; set some properties for our project
task.MODEL_NAME = 'Crosswalks'
task.MODEL_DESCRIPTION = 'Crosswalks from aerial data'

; set our rasters
task.TRAINING_RASTERS = training
task.VALIDATION_RASTERS = validation
task.PATCHES_PER_BATCH = patchesPerBatch

; set our outputs
task.OUTPUT_MODEL_URI = bestUri
task.OUTPUT_LAST_MODEL_URI = lastUri

; execute
task.execute
```
Creating Custom Training Data

When the out-of-the-box tools for data preparation are not enough, you have several options via the ENVI API to choose from to get your data in the right format.

Out-of-the-box tools allow you to create training data from ROIs, annotations, and vectors (after conversion to ROIs).

Recommended

Advanced usage
Verifying Custom Training Data

Use the “View Object Detection Raster Labels” tool to verify your bounding boxes are correct before training.

**Important note!**
There is an ENVI bug you may encounter while using this tool. It will be fixed in the next release of ENVI (likely Q1 2022).
Object Detection Examples
Aircraft Detection

Data Source: WorldView panchromatic
Ship Detection: Optical

Data Source: PlanetScope VNIR, surface reflectance
Ship Detection: SAR

Data Source: Sentinel 1 (SAR)
Turn Arrows

Data Source: RGB aerial imagery
Vehicle Detection

Data Source: RGB aerial imagery
## Object Detection vs. Pixel Classification

<table>
<thead>
<tr>
<th>Feature</th>
<th>Object Detection</th>
<th>Pixel Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Label types</strong></td>
<td>• Bounding box</td>
<td>• Points, polylines, or polygons</td>
</tr>
<tr>
<td><strong>What is used for training?</strong></td>
<td>• Source data, bounding boxes, and box classes</td>
<td>• Source data and classification image (i.e. mask)</td>
</tr>
<tr>
<td></td>
<td>• Each bounding box is a training sample, so there are few to learn from</td>
<td>• Each pixel is a training sample, so there are many to learn from</td>
</tr>
<tr>
<td><strong>Epoch for training</strong></td>
<td>• Recommended 60-75 minimum</td>
<td>• Depends on feature and training data, can get good models at 20 epochs</td>
</tr>
<tr>
<td></td>
<td>• Target loss should be &lt;=0.2; loss=0.5 will likely produce bad results</td>
<td>• Loss can vary, lower the better</td>
</tr>
<tr>
<td><strong>Size of features</strong></td>
<td>• Minimum: 25 pixels across (architecture limitation)</td>
<td>• Minimum: At least one pixel (no sub-pixel detections)</td>
</tr>
<tr>
<td></td>
<td>• Maximum: 400 pixels across (patch size limitation)</td>
<td>• Maximum: Can learn features larger than your patch size</td>
</tr>
<tr>
<td><strong>Relative size of classes</strong></td>
<td>• Don’t train a model with classes that vary dramatically in size</td>
<td>• Size is not as important</td>
</tr>
<tr>
<td></td>
<td>• For example: Not the best idea to detect parking lots and cars</td>
<td>• Multi-class size differences can impact performance because of class balance</td>
</tr>
<tr>
<td><strong>Model output</strong></td>
<td>• Bounding box, class, and confidence/score</td>
<td>• A class or probability for each pixel</td>
</tr>
</tbody>
</table>
Release details: https://www.l3harrisgeospatial.com/Support/Maintenance

JP Metcalf  
Solutions Engineer  
jp.metcalf@l3harris.com

Zachary Norman  
Product Manager  
zachary.norman@l3harris.com

L3Harris Geospatial
www.L3HarrisGeospatial.com  
geospatialinfo@l3harris.com  
303-786-9900