CHAPTER 4

Training a SAMLoRA model to identify specific features

Objectives

- Use a small set of training samples to train a deep learning model.
- Extract specific parts of imagery using the trained model.

Introduction

Out-of-the-box pretrained deep learning models may not be effective for specialized needs; in these cases, the SAMLoRA (Segment Anything Model with Low-Rank Adaptation) model may be a good solution. It is based on SAM, a multipurpose foundational model that was trained on massive datasets and uses the LoRA approach to quickly teach the model how to identify specific features after being exposed to only a small number of examples.

In this chapter, you'll train the SAMLoRA model in ArcGIS Pro to identify informal settlements in Alexandra, South Africa. You'll then apply the trained model to extract informal settlement building footprints.

Requirements

- ArcGIS Pro
- ArcGIS Image Analyst
- Deep Learning Libraries for ArcGIS Pro
- Recommended: NVIDIA GPU with a minimum of 4 GB of dedicated memory

Note: For this tutorial, an NVIDIA GPU with a minimum of 4 GB of dedicated memory is recommended. Based on whether your computer has a GPU and what its specifications are, this process may take from under 2 minutes to 20 minutes or more.

If you are not sure whether your computer has a GPU and what its specifications are, see chapter 1.

Tutorial 4-1

Use samples to train a SAMLoRA model

In this tutorial, you'll learn the process of training SAMLoRA to identify features of interest in your imagery—in this case, informal settlement buildings. You'll set up the ArcGIS Pro project, review the examples provided, and export the training data to the format that SAMLoRA expects.

Set up the project

1. In File Explorer, go to **C:/GeoAI_Data** and open the **Chapter04** folder. Double-click **Alexandra_Informal_Settlements.aprx** to open the project in ArcGIS Pro.

   On the map, a drone imagery layer shows a neighborhood in South Africa. The imagery is high resolution, with each pixel representing a square of about 2 × 2 cm on the ground.

2. Explore the built-up areas shown in the image.

Many of these built-up areas are informal settlements where buildings are built very close to each other, forming intricate patterns. Their roofs are made of corrugated metal sheets of varied colors and maintenance states. For these reasons, traditional deep learning models can have difficulty identifying such buildings with a high degree of accuracy. The SAMLoRA approach that you'll learn in this tutorial is a good alternative to obtain high-quality results without requiring high computing power.

Explore informal settlement examples

You'll review the training examples that were provided with your project.

1. In the **Contents** pane, check the box next to the **Training_Area** layer to turn it on.

   An orange polygon appears on the west side of the imagery. It represents the area chosen to train a SAMLoRA on what informal settlements look like.

2. In the **Contents** pane, right-click the **Training_Area** layer and choose **Zoom To Layer**.

   The map zooms in to the training area.

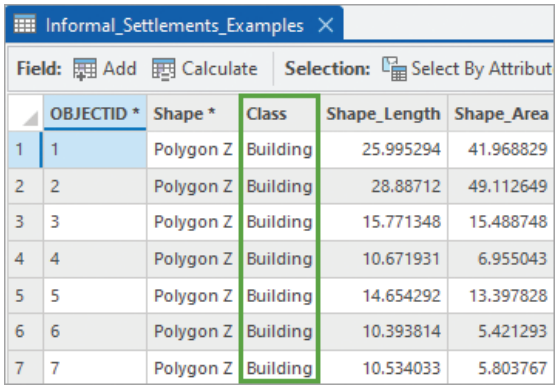
3 Turn on the **Informal_Settlements_Examples** layer.

The Informal_Settlements_Examples layer represents all the buildings in the training area as light-gray polygons. Every building in the training area was captured as a polygon. If even a few buildings were missing, it would generate confusing information and SAMLoRA would not train optimally. In the following example images, on the left, the training set is complete and ready to be used for training. In contrast, on the right, some buildings are missing, which would yield poor training performance.

Next, you'll inspect the Informal_Settlements_Examples attribute fields.

4. In the **Contents** pane, right-click the **Informal_Settlements_Examples** layer and click **Attribute Table**.

   Every line in the table represents one of the building polygons. The Class attribute has the value Building.



	OBJECTID *	Shape *	Class	Shape_Length	Shape_Area
1	1	Polygon Z	Building	25.995294	41.968829
2	2	Polygon Z	Building	28.88712	49.112649
3	3	Polygon Z	Building	15.771348	15.488748
4	4	Polygon Z	Building	10.671931	6.955043
5	5	Polygon Z	Building	14.654292	13.397828
6	6	Polygon Z	Building	10.393814	5.421293
7	7	Polygon Z	Building	10.534033	5.803767

> **Note:** Although this is not an essential part of this workflow, note that the value **Building** is a label for the underlying numeric value **1**. In a case in which the training examples represent several feature types, the **Class** field would list these several types—for instance, **Building**, **Road**, or **Tree**. This approach enables SAMLoRA to learn how to recognize different feature types in your imagery.

5. Close the table.

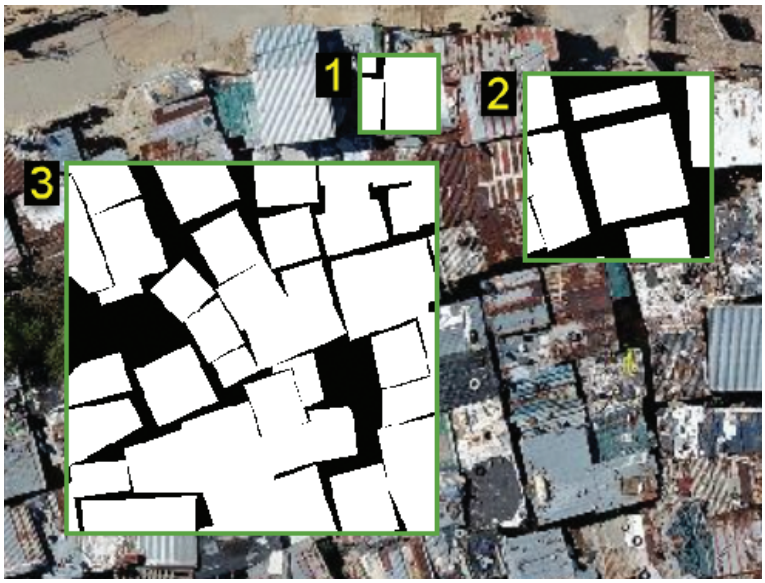
6. In the **Contents** pane, uncheck the box next to **Informal_Settlements_Examples** to turn the layer off and reduce clutter on the map.

Learn about training chips and cell size

You'll use the example polygon layer and the imagery layer to generate training data in a specific format. When producing training chips, one important thing to determine is their optimum size. To detect objects that are close to each other, as is the case with informal settlement buildings, a good guideline is that a chip should include 6 to 12 features. Another way to think about it is that a chip should contain at least one or two complete features in its center and show a good amount of context (or background) around it.

The following example image contains three chips of different sizes:

- Chip 1 is too small; it contains no complete features and almost no context.
- Chip 2 is a good size; there are two complete features in the center, as well as a few incomplete features and context around them.
- Chip 3 is too large; it contains many features (more than 25).



It is customary to have chips that measure 256 × 256 pixels (or cells), so you'll keep that default value. However, you can obtain chips of different sizes by varying the size of the cells that compose them. The optimum chip size will depend on the size of the features you want to identify.

Export training data

Next, you'll generate the training chips using the Export Training Data For Deep Learning tool.

1. On the ribbon, click the **Analysis** tab. In the **Geoprocessing** group, click **Tools**.
2. In the **Geoprocessing** pane, search for and open the **Export Training Data For Deep Learning (Image Analyst Tools)** tool. Apply the following settings:
   - For **Input Raster**, select **Alexandra_Orthomosaic**.
   - For **Output Folder**, type Informal_Settlements_256_5cm. (This creates a subfolder in your project to hold the training data.)
   - For **Input Feature Class Or Classified Raster Or Table**, select **Informal_Settlements_Examples**.
   - For **Class Value Field**, select **Class**.
   - For **Input Mask Polygons**, select **Training_Area**.

     Specifying the Training_Area layer as a mask is crucial because it will ensure that the image chips will be created only within the area where all the buildings are labeled.

   - For **Metadata Format**, select **Classified Tiles**.

     Classified Tiles is the metadata format that is expected to train a SAMLoRA model.
3. In the tool pane, click the **Environments** tab. For **Cell Size**, type 0.05.

   This cell size is 0.05 meters, or 5 centimeters. As indicated in the previous section, this cell size will ensure that the tiles generated are of a suitable size to identify informal settlement buildings.

   Note: Choosing the cell size for your image chips will not change the original resolution of your input imagery. The tool will resample the data on the fly to produce image chips of the desired cell size.
4. Click **Run**.

   The SAMLoRA deep learning model will use the labeled tiles to learn how the buildings look and where they are located. You can examine the training in the Catalog pane.

Train a SAMLoRA informal settlement model

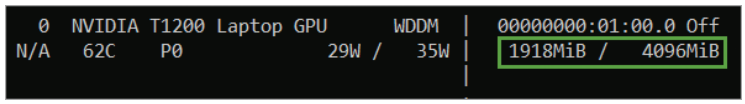
Next, you'll use the training data you generated to train the SAMLoRA deep learning foundational model and teach it to identify the informal settlement buildings in your imagery. Then, you'll review the trained model to better understand it.

1. Search for and open the **Train Deep Learning Model** tool and apply the following settings:
   - For **Input Training Data**, click the browse button. Browse to **Folders** > **Alexandra_Informal_Settlements**, select **Informal_Settlements_256_5cm**, and click **OK**.
   - For **Output Folder**, type Informal_Settlements_256_5cm_SAMLoRA.

     This parameter creates a subfolder in your project (under Folders > models) to hold the resulting trained model.
   - For **Max Epochs**, type 50.

     An epoch refers to one complete pass of the entire training dataset.
   - For **Model Type**, select **SAMLoRA (Pixel classification)**.
2. Expand **Data Preparation**. For **Batch Size**, if your NVIDIA GPU dedicated memory is 4 GB, type 2. If it is 8 GB, type 4.

   Note: You can examine your GPU memory usage live by opening the command prompt from the Windows start menu and pasting the command `nvidia-smi -l -5`.

   If you aren't reaching the maximum, you can increase the batch size the next time you run the tool.
3. Expand **Advanced** and apply the following settings:
   - For **Backbone Model**, select **ViT-B**.

     B stands for basic. This option trains the SAMLoRA model on a smaller (basic) neural network.

- For **Monitor Metric**, confirm that **Validation loss** is selected.

  This metric measures how well the model generalizes what it has learned to new data.

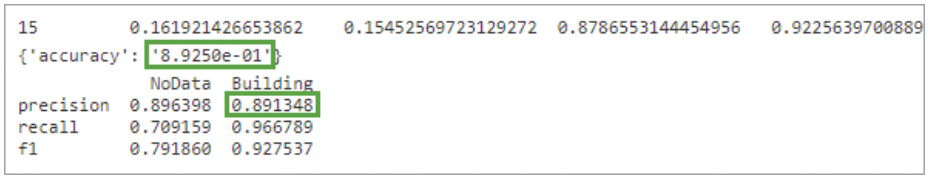
- Confirm the **Stop when model stops improving** option is checked to avoid model overfitting.

  You are now ready to run the tool.

4. Click **Run**.
5. Click **View Details**, and in the details window, click the **Messages** tab and monitor the **Validation loss** metric.

   The smaller the validation loss value, the better. It gradually decreases with each epoch as the model improves its ability to successfully identify building areas. When its value no longer changes significantly, the training stops. In parallel, the accuracy and Dice metrics (third and fourth columns) steadily increase. These metrics measure the model's performance.

6. When the training ends, review the **accuracy** and **Building precision** numbers.



   The values you obtain might be different. In the example image, the overall accuracy is 8.9250e-01, or 89.25 percent. For this use case, it should be between 85 and 95 percent. Because you are specifically interested in identifying buildings, the precision value for Building is the best measure of the model's performance. In this case, it is 0.8913 or 89.13 percent.

7. Close the details window.