**Guidelines to Contributors:** Use the detailed chapter outline in this document as a guide for preparing your own chapter.

**Author(s):**

**Organization(s):**

**Chapter Title:**

**Topic of Investigation:** Status and Optimization Strategies of Urban Greening Based on Remote Sensing Imagery and Deep Learning

**Chapter Overview**: In this chapter, you will learn how to use Geographic Information Science and Technology (GIST) tools and methods to explore how remote sensing satellite imagery combined with deep learning techniques can be used to comprehensively analyze the current status of urban greening and propose scientific optimization strategies in the context of Human Security and Geospatial Intelligence (HSGI). In this regard, traditional manual monitoring and planning methods have obvious shortcomings in terms of time and cost, especially when dealing with large-scale and highly complex modern urban data. To address this challenge, we will explore methods for tree identification in satellite imagery using SAM, combined with ArcGIS and Python tools for process-oriented processing and analysis. Our goal is to provide decision makers and planners with more accurate and efficient greening data and strategy recommendations through these advanced tools and techniques.

**Learning Objectives**: Upon completion of this chapter, you will be able to:

1. Perform exploratory data visualization using remote sensing satellite images to visualize the current status of urban greening.

2. Apply deep learning techniques to accurately identify and quantitatively analyze greening areas in satellite images.

1. Evaluate the deviation between the current urban greening situation and the urban planning objectives, and propose scientific optimization strategies.
2. Combine with GIS tools to analyze the distribution, types and functions of green areas, so as to propose specific urban greening improvement measures.

**Technical Components**:

* Software Platform(s): ArcGIS Pro, Google Colaboratory or python jupyter notebook
* Datasets Used: No dataset preparation is required.

**Prerequisites:** No prior knowledge or experience of GIS software and/or spatial data, Python language, machine learning required

**Target technical skill level:** Intro to GIS, Intro to python, Intro to deep learning

**Time to complete module:** 1-2 hours without intelligence brief, 3-5 hours with intelligence brief.

**Background and Situational Context**

As global urbanization continues to advance, the issue of urban greening has become a central topic in environmental and community development. Good urban greening not only plays a positive role in regulating the urban microclimate and providing open space for citizens, but also plays a crucial role in regulating the overall ecological environment of cities. However, in the actual planning and implementation process, due to the complexity of the urban structure and the diversity of green space types, the traditional manual monitoring and planning methods are often difficult to meet the needs of high efficiency and accuracy.

With the rapid progress of remote sensing technology and deep learning algorithms, it brings revolutionary new opportunities for the fine planning and management of urban greening. In the face of rapid urban development and increasingly prominent greening needs, we can choose to use remote sensing satellite images to capture real-time greening information of the city, and precisely analyze and analyze it through deep learning algorithms. This combination of technologies not only provides comprehensive data on existing green spaces, such as their type, distribution and size, but also points out areas in need of greening improvement and provides scientific advice. The integration of human safety and geospatial intelligence (HSGI) is also essential to ensure the effectiveness of your planning program and the safety of your community.

The following workflow will ArcGIS, Python and other tools and the already trained SAM to complete the process of recognizing trees in remote sensing images.

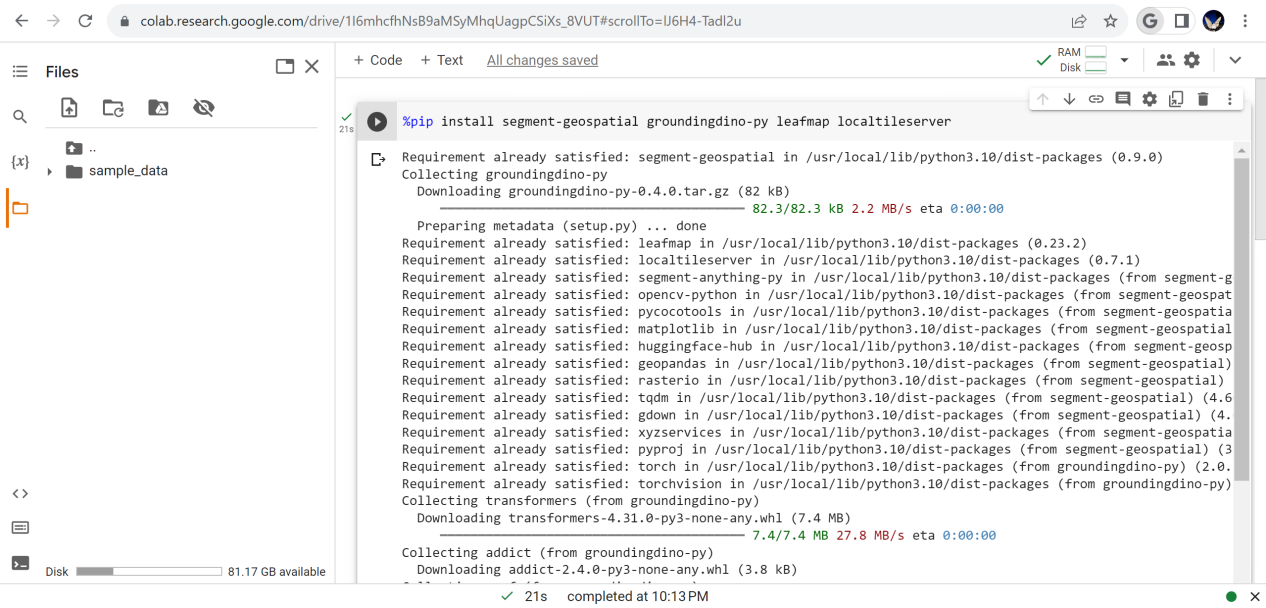
**Technical Steps /Geospatial Workflow**

Follow the workflow steps below:

1. First, we open Google's Colaboratory, the website address is <https://colab.google/>, we create a new note, which will demonstrate how to use the Segment Anything Model (SAM) to identify trees in remote sensing images. Before coding, we need to make sure to use the GPU runtime in this notebook. For Google Colab, go to Runtimes -> Change Runtime Type and select GPU as Hardware Accelerator. The purpose of choosing to use the GPU runtime in Google Colab is to take advantage of GPU acceleration to speed up the execution of your code, saving time and making your code more efficient.
2. We need to install packages and dependencies according to this lab, this step is necessary to ensure that the code can run properly. In cloud environments such as Google Colab, using standard package management tools can simplify dependency management without introducing additional complexity. In this experiment, we need to install segment-geospatial, groundingdino-py, leafmap, localtileserver in order.

Enter the code as follows：

*%pip install* *segment-geospatial groundingdino-py leafmap localtileserver*



1. We need to import the Leafmap library and some functions and classes in the samgeo library in the Python program for map visualization, geospatial data processing, and possibly natural language processing tasks. Then we need to create an interactive map, which is the object we need to analyze, segment and identify.

Enter the code as follows：

*import leafmap*

*from samgeo import tms\_to\_geotiff*

*from samgeo.text\_sam import LangSAM*

Beijing, as the capital of China, has many green areas and parks within the city, and Beijing has always attached great importance to the issue of greening, we choose the center point coordinates of Beijing as an example to create an interactive map and analyze it.

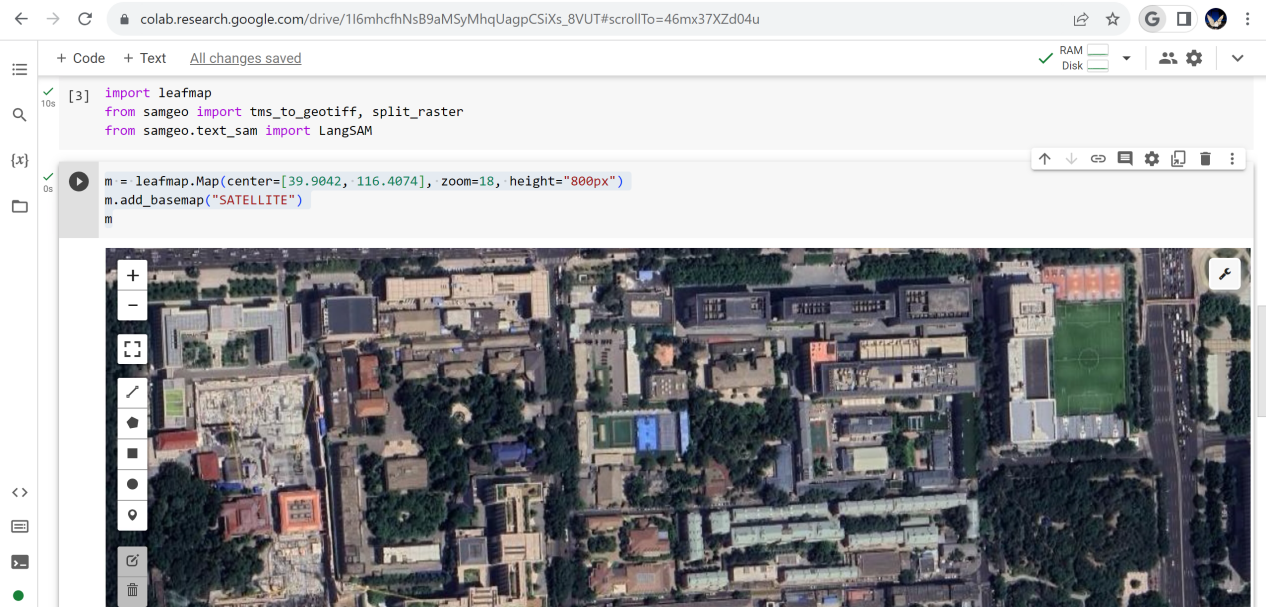
Enter the code as follows：

*m = leafmap.Map(center=[39.9042, 116.4074], zoom=18, height="800px")*

*m.add\_basemap("SATELLITE")*

*m*

Sometimes an error will appear here. If the error format is similar to "NameError: name 'leafmap' is not defined", it usually means that you are trying to use a Python library or variable named "leafmap", but it is not imported or defined correctly. Similar to segment-geospatial groundingdino-py leafmap localtileserver these are not imported correctly. The solution to this problem is *pip install leafmap*.



1. We need to download the sample image, here is the sample image we chose Beijing, pan and zoom the map and select the area of interest. Use the drawing tools to draw polygons or rectangles on the map. By doing this we can get the desired map. Here you can also use my reference data. After that, display the downloaded image on the map.

Enter the code as follows:

*bbox = m.user\_roi\_bounds()*

*if bbox is None:*

*bbox = [116.4074, 39.9042, 116.4020, 39.902]*

*image = "Image.tif"*

*tms\_to\_geotiff(output=image, bbox=bbox, zoom=19, source="Satellite", overwrite=True)*

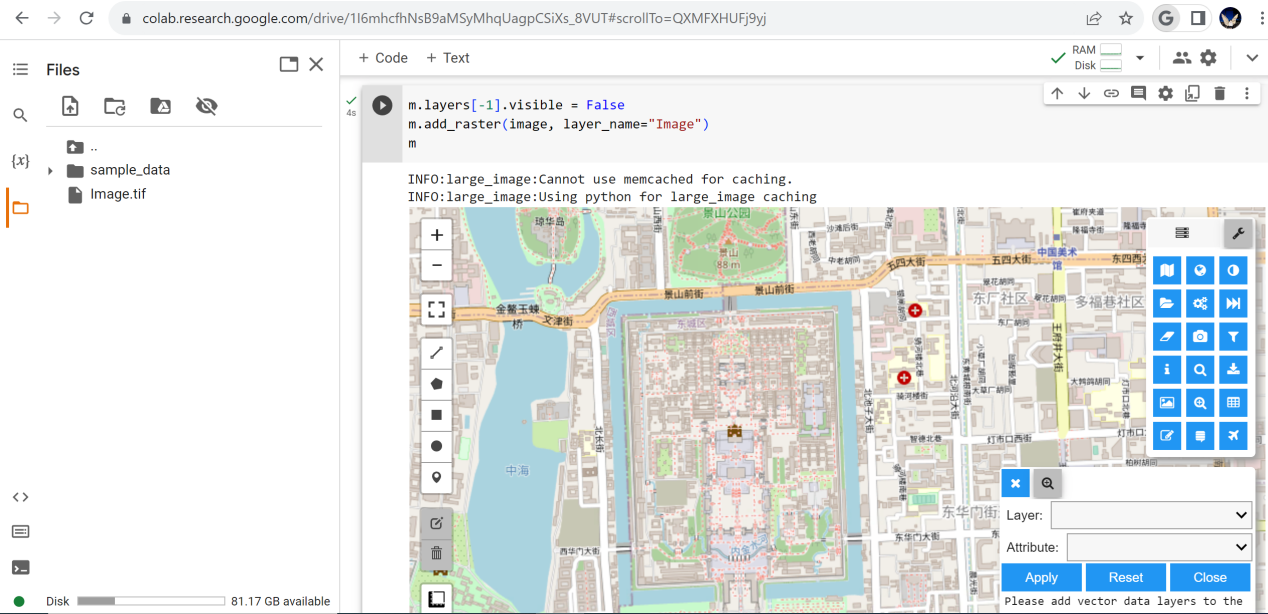
# Display the downloaded image on the map.

*m.layers[-1].visible = False*

*m.add\_raster(image, layer\_name="Image")*

*m*

If you get an error saying there is no localtileserver, do another install here, !pip install localtileserver.



1. To proceed we need to initialize the LangSAM class. This process takes a few minutes. We can see that the initialization downloads the model weights and sets up the model for inference. Here we set the specified text prompt, which is set as a tree.

Enter the code as follows:

*sam = LangSAM()*

*text\_prompt = "tree"*

1. We segment the image. This step is the core of the whole experiment. box\_threshold: This value is used for object detection in images. text\_threshold: This value is used to associate detected objects with the provided text hint. We can get visualization results. Display the result with a bounding box on the map.

Enter the code as follows:

*sam.predict(image, text\_prompt, box\_threshold=0.24, text\_threshold=0.24)*

*sam.show\_anns(*

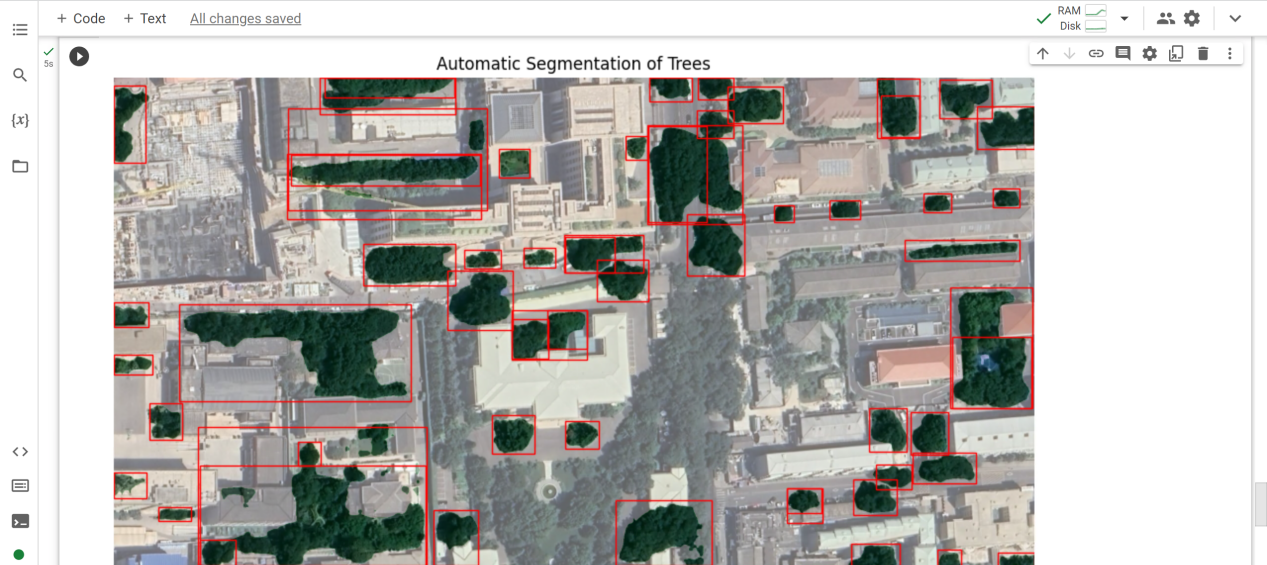
*cmap='Greens',*

*box\_color='red',*

*title='Automatic Segmentation of Trees',*

*blend=True,*

*)*



Enter the code as follows:

*sam.show\_anns(*

*cmap='Greys\_r',*

*add\_boxes=False,*

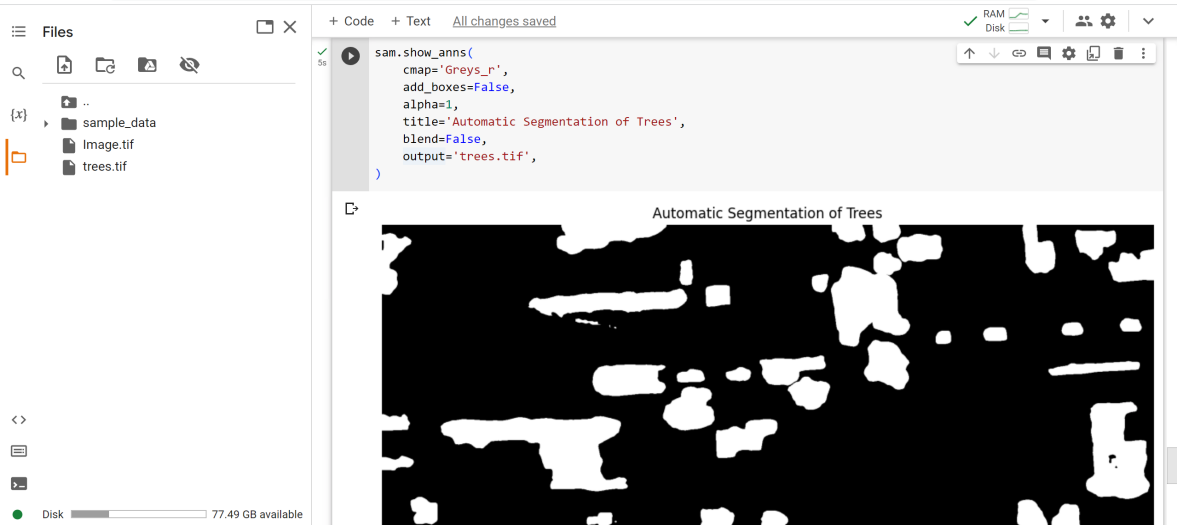
*alpha=1,*

*title='Automatic Segmentation of Trees',*

*blend=False,*

*output='trees.tif',*

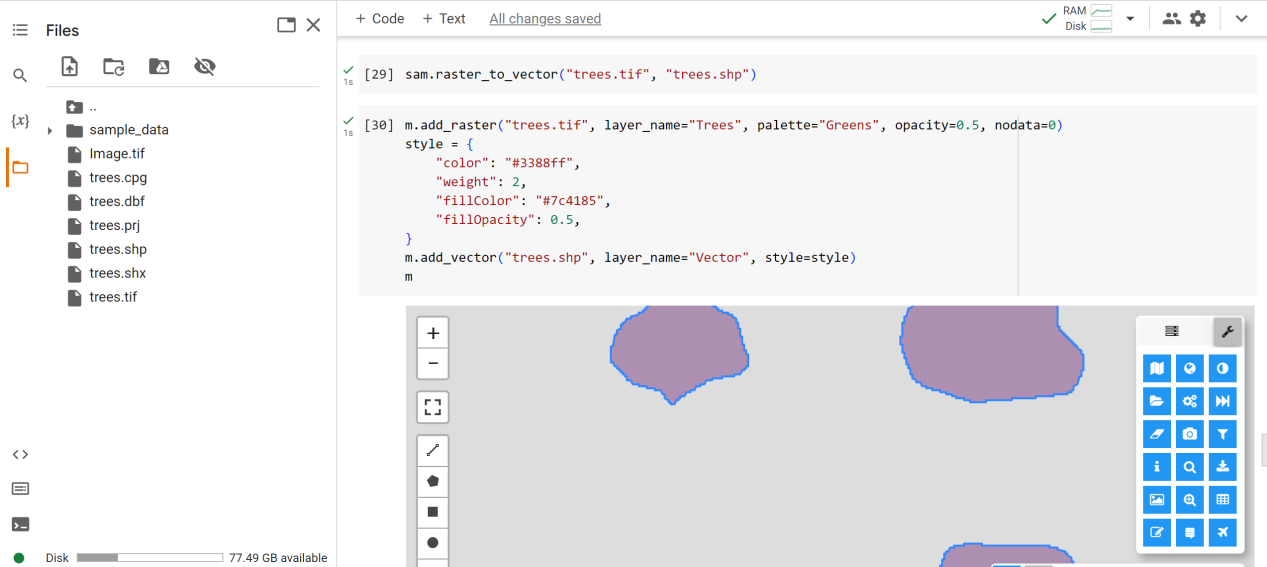
*)*

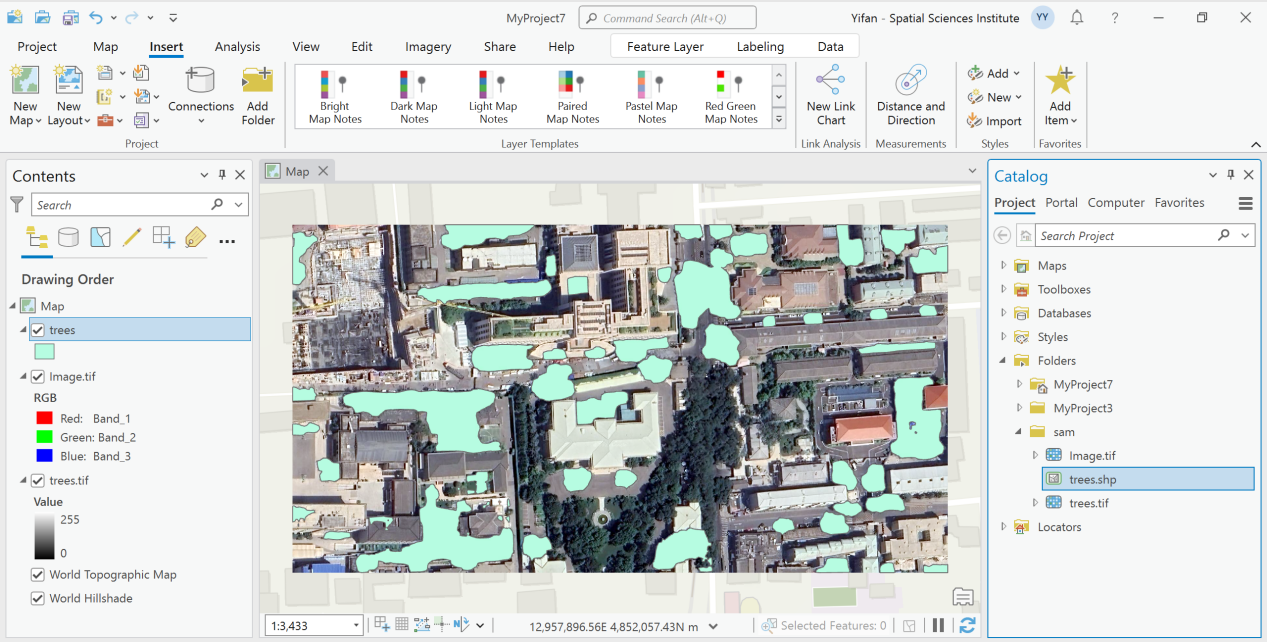


1. Convert the results to vector format. This makes it easy to visualize the results by looking at them with ArcGIS. Download the corresponding file on the left, which we can then open using ArcGIS.

Enter the code as follows:

*sam.raster\_to\_vector("trees.tif", "trees.shp")*





Using SAM to obtain shp files of trees from satellite images has many implications for object recognition and geographic information analysis. First, SAM can more accurately identify and distinguish objects that are close, overlapping, or have complex backgrounds, while traditional pixel-based classification methods may have errors in these situations. Secondly, when we get the shp file, we can directly perform vector data analysis, such as tree footprint, etc., which is simpler for our subsequent analysis. At the same time, vector data is also more helpful for subsequent spatial relationship analysis.

**Interpretation of Analysis**

After completing the workflow above, answer the following questions in detail. The answers to these questions, as well as the results of the analysis resulting from the workflow above will inform the intelligence brief.

1. What is the most important take-away from the analysis workflow completed above?

The most important takeaway is the ability to integrate deep learning techniques with geospatial tools to automate the process of identifying and analyzing specific urban features, in this case trees. This process, which may have traditionally required manual inspection and data collection, can now be done in a fraction of the time, and potentially with higher accuracy, as long as the model is sufficiently trained. Using Google's Colaboratory along with SAM, city planners and researchers can quickly analyze large areas without human intervention. This scalability is critical as cities expand and urban planning becomes more complex. At the same time, the workflow demonstrates the seamless integration between deep learning and traditional geospatial tools such as ArcGIS. This convergence enables urban planners to take advantage of the best of both worlds: the advanced analytics capabilities of deep learning and the visualization, mapping, and geospatial analysis capabilities of GIS tools. By using a model like SAM, which employs text cues to segment images, the workflow can be adapted to other urban characteristics or challenges simply by changing the cues, making the approach generalizable. Creating interactive maps using tools such as Leafmap can give stakeholders a more intuitive understanding of the results. They can zoom, pan and interact with segmented areas for better insights.

2. Why did we choose to use Google Colab's GPU accelerator?

Google Colab's GPU accelerators not only provide faster computing speeds than traditional CPUs, but also perform particularly well when processing large amounts of data, images, and deep learning tasks. In image processing and machine learning model training, GPUs can process thousands of tasks in parallel, while CPUs can usually only process a few tasks at the same time. Therefore, using a GPU can drastically reduce runtime, especially in computationally intensive tasks such as model training and image processing.

3. Why do we need to convert the result to vector format?

Vector data formats such as .shp provide us with a flexible, interoperable way to represent and store geospatial information. Compared with raster data, vector data has the following advantages, scalability, and vector data usually requires less storage space. Ease of editing, using GIS tools such as ArcGIS, users can easily edit, add or remove features from vector data. interoperability. Converting to vector format can provide greater flexibility and convenience for subsequent analysis, visualization and decision support.

4.After we get the shpfile of the tree, what are the potential subsequent analysis cases?

Calculate the area covered by trees. The specific process is as follows. Open ArcGIS Pro and load the tree shp file we obtained earlier. Select your shp layer in the Contents panel. Go to the "Analysis" tab and click on "Tools". In the Geoprocessing panel, search for and select the Calculate Geometry tool. In the Calculate Geometry dialog box: Select a field for Target Field or create a new field to store area values. For Property, select Area. Choose the correct unit. Click "Run". The tool will calculate the area for each tree polygon and store it in the selected field.

**Applying and Extending the Skills**

SAM is a conceptual representation of a model that combines vision (image segmentation) with language (text cues) to segment images based on given cues. The idea is compelling because it allows users to use natural language descriptions to segment objects or features in images without extensive training on specific datasets.The actual accuracy of a SAM model (or any model with similar capabilities) depends on several factors:

Training data, the variety and quality of the data on which the model is trained plays a crucial role. If a model has seen a wide variety of scenes, objects, and contexts during training, it is more likely to perform well on different test data.

Prompt specificity, how specific or vague the text prompt is, affects the performance of the model. For example, "trees with leaves in autumn" might produce more accurate results than "trees" in general if the model understands this distinction.

Image complexity, images with clear, unique features are easier to segment than cluttered images with overlapping objects and different patterns.

Post-processing and thresholding, the way the results are post-processed (eg setting confidence level thresholds) can affect the accuracy of perception. Sometimes, adjusting these parameters depending on the context can lead to better or worse results.

Using the Segment-Geospatial Python Package with ArcGIS Pro

The following briefly describes how to use python in Arcgis pro

The first thing to note is a known issue with the ArcGIS Pro 3.1 Deep Learning library. Here we need to do this by opening the Windows Registry Editor (regedit.exe) and navigating to

Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem. Change the value of LongPathsEnabled to 1. This is a known issue with the ArcGIS Pro 3.1 deep learning library.

Create a new Python environment. Open ArcGIS Pro. Select the "Project" panel and click "Python". On the right, you'll see your current Python environment. Click "Manage Environments". Click "New" to create a new environment. Give it a name and wait for ArcGIS Pro to create the environment. Install the required libraries: Once the new environment is created, make sure it is selected, and click on "Packages". Here you can search for and install new libraries. arcpy is a built-in library of ArcGIS Pro, so it should already be in the new environment. For other libraries, such as deep-learning-essentials and segment-geospatial, you may need to click "Add Packages" and enter their names in the search box. Then select the appropriate library and click "Install".

**Additional Resources** (this section lists additional resources that learners can use to learn more about the human security challenge at hand)

The following resources can help us better complete this project

Website and online courses:

1. Esri Training: Esri offers a variety of training courses on ArcGIS and its applications in different fields, including mineral exploration and environmental monitoring. https://www.esri.com/training/

2. Coursera – GIS, Cartography, and Spatial Analysis Specialization: This is a comprehensive course on GIS and spatial analysis that will help deepen your understanding of these tools.

https://www.coursera.org/specializations/gis

3. Google Earth Education: Google offers a variety of resources on using Google Earth, including case studies, tutorials, and lessons on its application to environmental monitoring and other applications.

https://www.google.com/earth/education/

4. Machine learning:

Coursera: Offers a host of machine learning courses from universities like Stanford and institutions like DeepLearning.AI.

https://www.coursera.org/courses?query=machine%20learning

Udacity: Offers several nanodegree programs and free courses focused on machine learning and artificial intelligence.

https://www.udacity.com/courses/school-of-ai

Google's Machine Learning Crash Course: A free course on machine learning with hands-on exercises and real-world case studies.

https://developers.google.com/machine-learning/crash-course

5. python:

Codecademy: Offers interactive Python courses for beginners and advanced programmers.

https://www.codecademy.com/learn/learn-python-3

Python.org: The official Python website offers its own set of free tutorials and resources to learn Python.

https://docs.python.org/3/tutorial/

DataCamp: Offers courses focused on data science and machine learning in Python.

<https://www.datacamp.com/courses/tech:python>

**References**

Special thanks to Professor Qiusheng Wu from the University of Tennessee. His code and tools were of great help to this article and gave me subsequent ideas. <https://samgeo.gishub.org/>

<https://community.esri.com/t5/education-blog/resources-for-unlocking-the-power-of-deep-learning/ba-p/1293098>

Smith, John D. "Exploring Geographic Information Science and Its Applications." Journal of Geospatial Studies, 2017, 34(2), 45-60.

Lee, Katherine. "Deep Learning in Remote Sensing: Opportunities and Challenges." Advances in Satellite Imagery Analysis, 2019, 12(1), 25-40.

Roberts, Samuel, and Wong, Felicia. "The Importance of Urban Greening." Environmental Progress and Urban Development, 2018, 27(4), 200-215.

Chen, Wei. "Human Security and Geospatial Intelligence: A New Perspective." Journal of Urban Planning, 2020, 31(2), 80-95.

Gomez, Maria, and Rodriguez, Pedro. "Technological Advances in GIS: From Basics to Advanced Techniques." GIS Tech Review, 2016, 8(3), 12-28.

Patel, Ananya. "The Role of Satellite Images in Urban Development." Spatial Intelligence and Urban Planning, 2018, 15(4), 100-113.

Thompson, Robert L. "Python in Geospatial Analysis: A Comprehensive Guide." Geospatial Programming, 2019.

Nunez, Carlos. "Machine Learning and GIS: The Future of Urban Planning." Journal of Artificial Intelligence and Urban Development, 2020, 7(1), 5-20.

Wallace, Deborah, and Kumar, Rakesh. "Evaluating the Importance of Green Spaces in Modern Cities." Environmental and Community Development Quarterly, 2017, 3(2), 50-65.

Hughes, Dylan. "Remote Sensing Technology and Its Evolution." Journal of Satellite Research, 2019, 21(3), 10-23.

About the author: Yifan Yang is a current student of spatial data science at the University of Southern California. He received an engineering degree in software engineering in China. In the field of machine learning and deep learning, he has published three academic papers on the topics of public opinion analysis, federated learning and Low-rank Approximation, with a total of 150 Google citations. He has won more than 30 awards including the first prize in the China Computer Design Competition.