Getting Started

Important! Be sure to read [About this liveProject](https://liveproject.manning.com/module/106_1_2/monitoring-changes-in-surface-water-using-satellite-image-data/introduction/start-project?) before beginning. It contains crucial information for your work.

## Objective

The objective of the first milestone is to familiarize yourself with the development environment that will be used throughout the liveProject, to understand what is meant by image segmentation and to get background information about the problem of vanishing lake area. While working on this milestone, you will:

* Install all required Python libraries
* Develop familiarity with Conda, pip, Git, Jupyter Notebook, TensorFlow, and Keras
* Develop familiarity with the problem of vanishing lake area
* Get an introduction to image segmentation with deep learning

Before starting to work on this milestone, make sure you downloaded the liveProject resources, as you will be using one of the files to understand the development environment data scientists usually adopt. <<ADD HERE LINK TO THE NEW ZIP FILE WITH THE RESOURCES!!!>>

## Importance to project

Working on this milestone will make sure that you have acquired all the skills necessary to complete the data science project you were tasked to complete:

* Conda environments allow you to start with a fresh installation of a collection of Python libraries that are mutually compatible, along with various other benefits described in the milestone.
* Familiarity with Jupyter Notebook is essential to creating project deliverables: one notebook per milestone uploaded to GitHub.
* TensorFlow is one of the main deep learning frameworks, and Keras is a high-level neural network API, an industry standard way to access low-level TensorFlow functionality.
* Understanding your data is essential to any data science project.

## Introduction

* This milestone’s tutorial starts with this document, which covers the preliminaries of Conda environments, Python packages, and running a Jupyter Notebook server. The remainder of this tutorial takes place in **Part\_1\_Getting\_Started.ipynb**, provided with the project resources. Files with the **ipynb** extension are Jupyter Notebooks. This notebook introduces this powerful development environment, as well as the libraries that will be used over the course of the project. You will submit all deliverables as a Jupyter notebook, so it is important to become familiar with Jupyter Notebook syntax and best practices in this milestone.

## Workflow

1. Get the liveProject materials.
2. Using Google Colab.
3. Working on your local machine.
   1. Conda environments and packages
   2. Install packages
4. Learn Jupyter Notebooks basics.
   1. Using markdown and HTML within cells
   2. Magic functions and optional extensions
   3. Inline plots using Matplotlib
5. Understand the basics of Tensorflow and Keras
6. Train a simple neural network that classifies images
7. Complete the deliverable applying what you have learned

## Set up an online environment.

For this liveProject, we recommend using [Google Colaboratory](https://colab.research.google.com/) (or Colab), an online, interactive editor that allows you to run your code without having to worry about installing libraries. Additionally, you will have access for free to powerful GPUs that will run your code much faster. If you are not familiar with Colab we recommend you to watch [this video](https://www.youtube.com/watch?v=inN8seMm7UI) and to read the introductory [notebook](https://colab.research.google.com/).

## Set up a local environment.

If you prefer working on your own machine, and if your hardware can support training deep learning models, you could also run all the code locally. To do so, make sure that your machine meets the following requirements:

* Central processing unit (CPU) — Intel Core i5 6th Generation processor or higher, or an AMD equivalent processor
* RAM — 8 GB or higher
* Graphics processing unit (GPU) — NVIDIA GeForce GTX 960 or higher
  + AMD GPUs can’t execute the code in this project. Read this [NVIDIA Developer guide](https://developer.nvidia.com/cuda-gpus" \t "_blank) for more information on NVIDIA GPUs for deep learning.
* Operating system (OS) — [Ubuntu Linux](https://ubuntu.com/" \t "_blank) (or Ubuntu derivatives such as [Pop!OS](https://system76.com/pop" \t "_blank)) or Microsoft Windows 10

The following setup is only required if you want to run all your code on your local machine. If you plan on using Google Colab, you can safely skip this session and jump straight to “Jupyter Notebook**”**.

### Setting up a Conda environment and packages

You will use **Conda** to create a virtual Python 3.7 environment. This will help ensure that all learners use the same environment and installed packages. To ensure an environment that works well on multiple platforms, we will use both **Conda** and **pip** to install packages. You’ll use Conda to install the bare bones (an environment, Python, and a few other core dependencies that are more difficult to install with pip) and pip to install the remaining packages to complete the environment.

Refer to [Understanding Conda and Pip](https://www.anaconda.com/understanding-conda-and-pip/" \t "_blank) to understand the differences between these two package/environment management tools.

We encourage Windows and Linux users to use the [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)" \t "_blank) Python 3.9 distribution. Windows users should refer to this comprehensive [installation guide on Katie Kodes](https://katiekodes.com/setup-python-windows-anaconda/" \t "_blank). Debian-based Linux users might find the tutorial [How To Install Anaconda on Ubuntu 18.04](https://www.digitalocean.com/community/tutorials/how-to-install-anaconda-on-ubuntu-18-04-quickstart" \t "_blank) useful.

Open an Anaconda command window and make sure your **Conda** environment is up to date:

conda update -n base -c defaults conda

If your environment is up to date, running this command will not do anything. More **Conda** commands can be found on [Conda Docs](https://conda.io/docs/user-guide/tasks/manage-environments.html" \t "_blank).

### Getting the liveProject materials

If you still have not done so, download the resource file for this liveProject by clicking on this link <<ADD HERE LINK TO THE NEW ZIP FILE WITH THE RESOURCES!!!>>

Once you have obtained the file, extract his content in a folder named DeepLearningSatelliteImage. Using the Anaconda Command Window (or a terminal if you are using Ubuntu), change the current directory into the one you just created. You should use a command like this:

cd DeepLearningSatelliteImage

Next, we are going to create a **Conda environment**. While this is not mandatory, it is considered a best practice as it allows working on multiple projects with different requirements, and it avoids modifying the Python installation used by the operating system. Read more about Conda environments and the rationale behind them in the article [The Definitive Guide to Conda Environments](https://towardsdatascience.com/a-guide-to-conda-environments-bc6180fc533" \t "_blank).

Create an environment called “**liveproject**” using the yml file provided in the resources:

conda env create -f 1\_GettingStarted/conda\_env/env.yml

This command will create a self-contained Python environment and we can use it to install, update or remove the packages required to complete this liveProject.

To activate the Conda environment and clean up the downloaded packages (to save computer memory), issue the following commands:

conda activate liveproject

conda clean --all

To deactivate the Conda environment (it is recommended you do this before you close the Anaconda command window):

conda deactivate

If you want to remove the environment, you could use:

conda env remove --name liveproject

or

conda remove –name liveproject --all

### Installing necessary Python libraries

After creating the environment, we already have installed some of the required libraries, but we still need a few others, such as Sentinelsat and Tensorflow. You can install some of them using **pip** with the following command:

pip install sentinelsat numpy scipy jupyter matplotlib scikit-image scikit-learn wget requests imagecodecs

If you get timeout errors, change the timeout to a higher number, for example:

pip install sentinelsat --default-timeout=100

For installing TensorFlow, please refer to the official [Google documentation](https://www.tensorflow.org/install/pip). Please be aware that you will have to install Nvidia drivers and the latest supported CUDA for TensorFlow to take advantage of the speed-ups provided by running on the GPU.

## Jupyter Notebook

This liveProject is conducted through a series of **Jupyter** notebooks. When creating the Conda environment, you also installed the Jupyter library. Google Colab is, in essence, a web interface for running Jupyter Notebooks on the cloud, with free access to Google GPUs.

To run **Jupyter Notebook** on your PC through your browser, issue the following command from the Anaconda command window (or the terminal if on Linux), making sure that the Conda environment you created is activated:

jupyter notebook

This command should launch a new tab in your default web browser that displays a list of Jupyter notebooks (with an .ipynb extension) on your system.

You can navigate through the list and launch a notebook by double-clicking on its name. Alternatively, you can launch any notebook from the command line by navigating to the notebook’s directory and issuing the following command:

jupyter notebook A\_Notebook.ipynb

Ok, this is it, now you should be ready to go. Please open (either on your machine or in Colab) the notebook Part\_1\_Getting\_Started.ipynb provided in the liveProject resources. Once you have reviewed all the content in the notebook, you are ready to create a new one, which will represent the Deliverable for Milestone 1

## Deliverable

* The deliverable for this milestone is a Jupyter notebook showing an example image or images of a satellite dataset read in using Rasterio and a demonstration of a function that carries out manipulation of that image using Keras and TensorFlow 2.0. That manipulation could be anything that alters the image, such as its size, geometry (shape), pixel intensities, or spatial projection. This will mostly test your understanding of Keras syntax, which is an essential component of the remaining milestones.
* Upload a link to your deliverable in the Submit Your Work section and click submit. After submitting, the Author’s solution and peer solutions will appear on the page for you to examine.

## Help

Feeling stuck? Use as little or as much help as you need to reach the solution!

resources

The following resources provide more background and in-depth explanation of key concepts in spatial data, spatial data processing and analysis, and geospatial Python libraries such as GDAL:

Geoprocessing with Python by Chris Garrard

Chapter 9, “Reading and writing raster data,” explains raster data.

Geoprocessing with Python by Chris Garrard

Chapter 10, “Working with raster data,” provides the raster data.

Geoprocessing with Python by Chris Garrard

Chapter 11, “Map algebra with NumPy and SciPy,” provides information on raster data.

### Additional resources

If you need more help getting to grips with geographical data, coordinate transformations, and other Earth analytics concepts, check out the following:

* [Earth Analytics Python Course](https://www.earthdatascience.org/courses/earth-analytics-python/) is a free course from the University of Colorado Boulder. Of particular relevance are the Python tutorials on [spatial data](https://www.earthdatascience.org/courses/earth-analytics-python/spatial-data-vector-shapefiles/intro-vector-data-python/), [satellite imaging](https://www.earthdatascience.org/courses/earth-analytics-python/multispectral-remote-sensing-in-python/intro-multispectral-data/), [coordinate reference systems](https://www.earthdatascience.org/courses/earth-analytics-python/spatial-data-vector-shapefiles/intro-to-coordinate-reference-systems-python/), and [exporting rasters](https://www.earthdatascience.org/courses/earth-analytics-python/multispectral-remote-sensing-in-python/export-numpy-array-to-geotiff-in-python/) using Rasterio.
* [An introduction to satellite imagery and machine learning](https://www.azavea.com/blog/2019/11/05/an-introduction-to-satellite-imagery-and-machine-learning/) is an industry-focused review of the start of the art.
* See [Reading and manipulating tiled GeoTIFF datasets](https://examples.dask.org/applications/satellite-imagery-geotiff.html) on using Rasterio and dask.

Additional Keras/TensorFlow tutorials and Python resources related to image classification and segmentation include the following:

* [Intro to Keras](https://colab.research.google.com/drive/1pMcNYctQpRoBKD5Z0iXeFWQD8hIDgzCV)
* [Intro to TensorFlow 2.0](https://towardsdatascience.com/a-quick-introduction-to-tensorflow-2-0-for-deep-learning-e740ca2e974c) with image classification examples
* [python-is-cool](https://github.com/chiphuyen/python-is-cool) GitHub page
* [Loading and manipulating imagery for deep learning](https://machinelearningmastery.com/how-to-load-and-manipulate-images-for-deep-learning-in-python-with-pil-pillow/)
* [tf.image](https://www.tensorflow.org/api_docs/python/tf/image)
* [TensorBoard](https://www.datacamp.com/community/tutorials/tensorboard-tutorial)

The following resources provide more information on specific concepts and terminology used in Milestone 1.

* [Confusion matrix](https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62)
* [MNIST database](https://en.wikipedia.org/wiki/MNIST_database)
* [Image spatial projections](https://geohackweek.github.io/raster/04-workingwithrasters/)
* [EPSG spatial projection codes](http://www.epsg-registry.org/)