Image from NASA: https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

Geospatial Deep Learning

Introduction

Welcome to *Geospatial Deep Learning*. This course was created by West Virginia View with support from AmericaView and the United States Geological Survey (USGS). Additional support was provided by the National Science Foundation (NSF). This course was designed for geospatial professionals and students that are interested in adding deep learning to their toolkit to answer questions with a spatial component. The lecture modules focus on key concepts associated with deep learning and applying these techniques to geospatial data. The code examples make use of the PyTorch Python package. It is assumed that you can already code in the Python language. If you have no prior experience with Python, we recommend that you work through our *Methods in Open Science* course before attempting this course.

This course is broken into four broad components. The first component focuses on the basics of artificial neural networks and fully connected neural networks. The second component focuses on convolutional neural networks (CNNs) for scene labeling tasks while the third component focuses on CNNs for semantic segmentation. The last component discusses instance segmentation, generative models, variational auto encoders, and frontiers, including Transformer and Mamba architectures.

Deep learning has become a powerful tool for a variety of tasks across a wide

range of professions and disciplines. For example, deep learning has resulted in significant advances in computer vision and scene understanding for autonomous vehicles. Google and Facebook use deep learning models to refine content based on use history. The analysis of medical imagery has also greatly benefited from these advancements.

The geospatial sciences (GIScience, spatial modeling, and remote sensing) have adopted these methods to address key questions and needs.

Module Topics

1. Uses of deep learning in the geospatial sciences
2. Software, resources, and datasets
3. Supervised learning
4. Supervised learning process
5. Coding options for geospatial deep learning

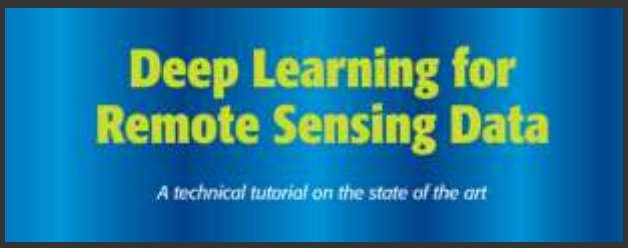
2

These are the topics covered in this module.

Introduction

Deep Learning and Geospatial Science

- Make predictive models
- Automated mapping
- Object detection
- Image classification
- Image gap filling
- Time series analysis
- Computer vision
- Self driving cars



https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7486259

https://ieeexplore.ieee.org/abstract/document/8113128

4

This slide lists some common uses of deep learning in the geospatial sciences.

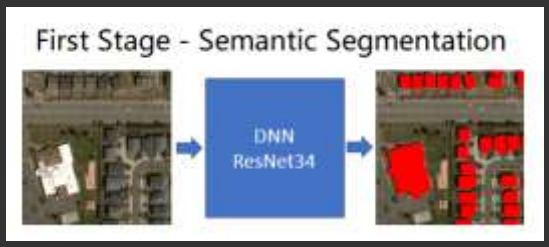
Generally, deep learning can be used to make predictions, such as the likelihood of a landslide occurring at a location, or to map features on the landscape surface, such as the location of buildings or the land cover occurring at a location.

Deep learning can be applied to vector data and raster data. It can also be applied to time series data.

In this course, we will focus on using deep learning for mapping and object detection.

The links on this page are for two review articles that explore the use of deep learning in remote sensing.

https://www.nationalgeographic.org/funding-opportunities/grants/what-we-fund/ai-earth-innovation/



https://github.com/microsoft/USBuildingFootprints

https://azure.microsoft.com/en-us/blog/microsoft-and-esri-launch-geospatial-ai-on-azure/

http://lila.science/datasets/chesapeakelandcover

This slide provides some examples of geospatial deep learning projects. For example, Microsoft used deep learning techniques to map all buildings in the United States, a dataset that has been made freely available. Microsoft and the National Geographic Society offer grants yearly to individuals interested in applying deep learning to Earth observation data and other environmental data.

There are several frameworks, libraries, or packages available for implementing deep learning in a specific environment or coding language. Here, we will make use of PyTorch.

PyTorch was originally developed by Meta, formally Facebook. The code is free and open-source. Management of PyTorch has now been handed over to the PyTorch Foundation, which is associated with the Linux Foundation.

We are using PyTorch in this course, as opposed to other available deep learning environments (e.g., Tensorflow/Keras) because it is currently the leading environment, especially in regards to research applications.

We will also provide some discussion of the torch R package and associated ecosystem. However, this is not a primary focus of this course.

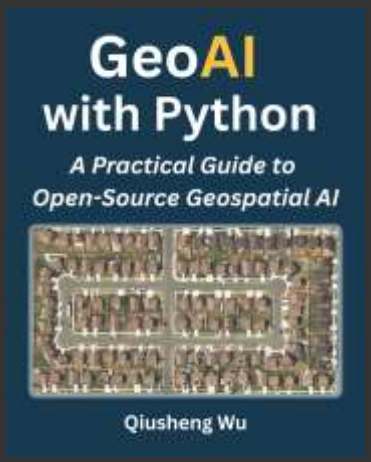
Other Packages

- Torchinfo — https://pypi.org/project/torchinfo/
- TorchMetrics — https://torchmetrics.readthedocs.io/en/stable/
- Segmentation Models — https://github.com/qubvel/segmentation_models.pytorch
- Albumentations — https://albumentations.ai/
- Torchvision — https://pytorch.org/vision/stable/index.html
- Rasterio — https://rasterio.readthedocs.io/en/latest/
- GeoAI — https://github.com/opengeos/geoai

7

This slide lists some other packages that we will make use of in this class. Torchinfo, formally Torch-summary, allows for generating summaries of model architectures. TorchMetrics provides access to a wide variety of assessment metrics. Segmentation Models allows for easier implementation of a variety of semantic segmentation methods (e.g. UNet, UNet++, and DeepLabv3+). Albumentations is used for data augmentation and is especially useful when performing semantic segmentation task. Torchvision adds additional functionality to PyTorch associated with vision or image analysis tasks. Rasterio allows for reading and working with geospatial data in Python.

GeoAI



- Environment Setup
- Geospatial Data
- Downloading Data
- Data visualization
- Preparing training data
- Image recognition
- Object detection
- Semantic segmentation
- Image translation
- Change detection
- Pixel-level regression
- SAM for Geospatial
- Vison-Language Models
- Satellite Embeddings
- QGIS plugins

https://book.opengeoai.org/

8

We will also provide some Python examples that make use of the GeoAI package, which greatly simplifies the implementation of deep learning applied to geospatial data.

Datasets

❖EuroSat https://www.kaggle.com/datasets/apollo2506/eurosat-dataset

Helber, P., Bischke, B., Dengel, A. and Borth, D., 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *12*(7), pp.2217-2226.

❖topoDL https://doi.org/10.6084/m9.figshare.25096640.v1

Maxwell, A.E., Bester, M.S., Guillen, L.A., Ramezan, C.A., Carpinello, D.J., Fan, Y., Hartley, F.M., Maynard, S.M. and Pyron, J.L., 2020. Semantic segmentation deep learning for extracting surface mine extents from historic topographic maps. *Remote Sensing*, *12*(24), p.4145.

Maxwell, A.E., 2024. topoDL: A deep learning semantic segmentation dataset for the extraction of surface mine extents from historic USGS topographic maps.

❖Landcover.ai https://landcover.ai.linuxpolska.com/

Boguszewski, A., Batorski, D., Ziemba-Jankowska, N., Dziedzic, T. and Zambrzycka, A., 2021. LandCover. ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1102-1110).

9

We will make use of several different datasets in the examples in this course. The EuroSat dataset will be used in the fully connected neural networks and convolutional neural networks component of the class. In the semantic segmentation component, we will use topoDL, which represent a binary pixel-level classification problem, and Landcover.ai, which represents a multiclass pixel-level classification problem.

These datasets can be obtained at the links provided on this page. We have also provided citations for the papers that introduced the datasets.

Datasets

❖vfillDL https://doi.org/10.6084/m9.figshare.22320373.v2

Maxwell, A.E., 2023. vfillDL: A geomorphology deep learning dataset of valley fill faces resulting from mountaintop removal coal mining (southern West Virginia, eastern Kentucky, and southwestern Virginia, USA). figshare.

❖terraceDL https://doi.org/10.6084/m9.figshare.22320373.v2

Maxwell, A.E., 2023. terraceDL: A geomorphology deep learning dataset of agricultural terraces in Iowa, USA. figshare. Dataset.

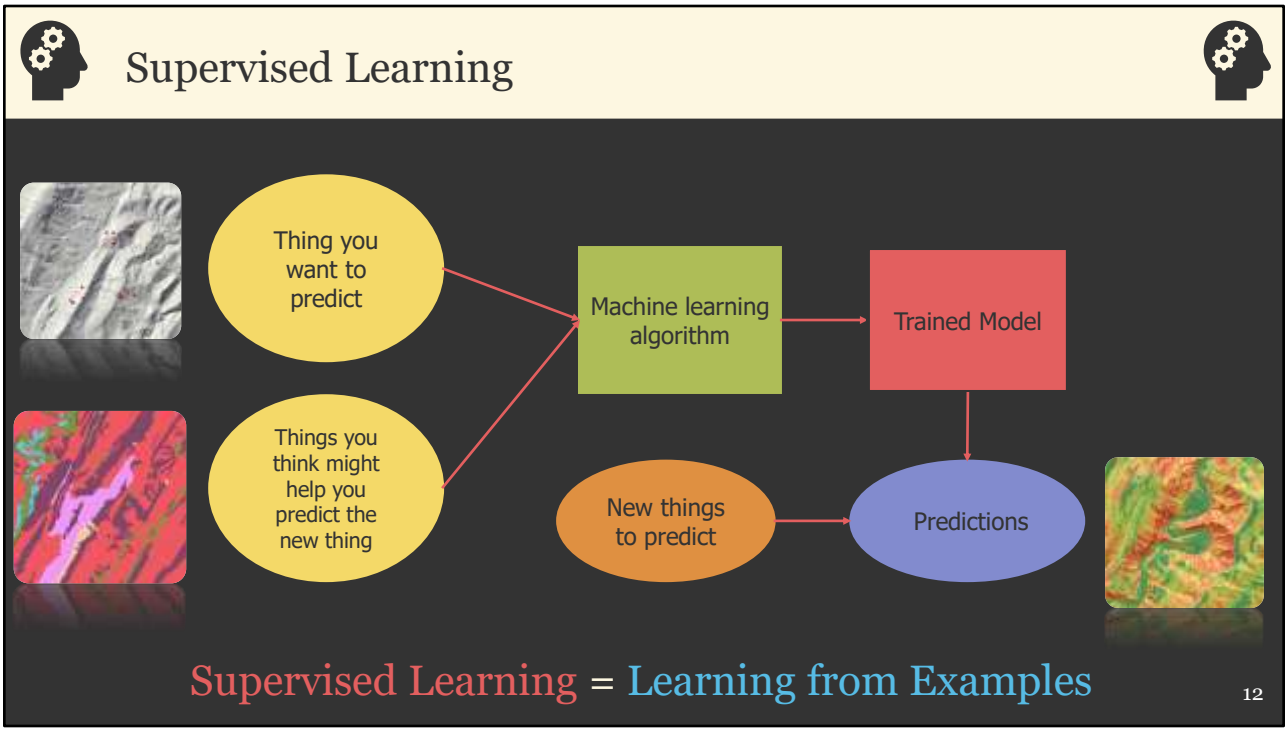
❖mineBenchDL https://doi.org/10.6084/m9.figshare.26042920.v1

Maxwell, A.E., S. Farhadpour, and M Ali., 2024. mineBenchDL: A geomorphology deep learning dataset of historic surface coal mine benches in West Virginia, USA.

10

Prof. Maxwell and his research lab have generated several additional datasets relating to geomorphic deep learning using land surface parameters (LSPs) derived from digital terrain models. These datasets are made available via FigShare.

Conceptualizing Deep Learning

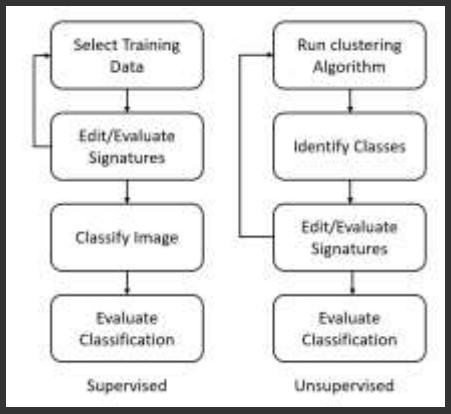


Deep learning generally makes use of supervised learning, in which the algorithm learns from example data and input predictor variables. There are some deep learning methods that rely on unsupervised learning or semi-supervised learning. However, we will focus on supervised methods in this course.

Conceptually, an algorithm is provided with predictor variables and labels. It then uses this information to produce a model, which can then be used to make predictions on new data.

For example, a dataset could consist of images, in which the predictor variables are the image bands, and associated labels for each image. Each image could be labelled as a representation of a different environment (forest, grassland, pasture, agricultural field, urban, residential, etc.). These data could then be used to train an algorithm to label new images into one of the environmental classes.

Supervised vs. Unsupervised Learning



13

Supervised learning requires that examples be provided upfront as training data or labels. The algorithm then uses the provided labels and predictor variables to "learn" or produce a model.

In contrast, unsupervised learning does not required examples upfront. Instead, the algorithm attempts to cluster the data into categories. The user then assigns the generated clusters to informational classes. So, user-input is still required in unsupervised learning; however, this happens at the end of the process by assigning the learned categories or clusters to informational classes. In contrast, supervised learning requires user input as training data upfront.

Since we will primarily work with supervised learning in this class, the examples used will consist of input labels and predictor variables. The nature of these data will depend on the problem being explored and the requirements of the algorithm. For example, the dataset could consist of images with associated categorical labels, in the case of convolutional neural networks, or images and pixel-level categorical masks, in the case of semantic segmentation. A large component of deep learning is data pre-processing to get your inputs into the right format for the algorithm to accept them.

Semi-supervised methods allow for learning from both labeled and unlabeled data. These methods will not be a focus of this course.

Predicting a Numeric or Continuous Outcome

- Linear regression
- Multiple linear regression
- Polynomial regression
- Machine learning
- Deep learning

Percent Canopy Cover NLCD 2011

https://www.mrlc.gov/index.php

14

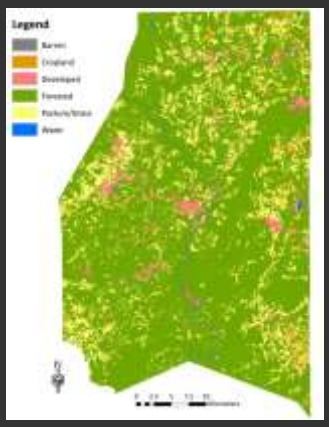
Different types of features can be predicted using deep learning.

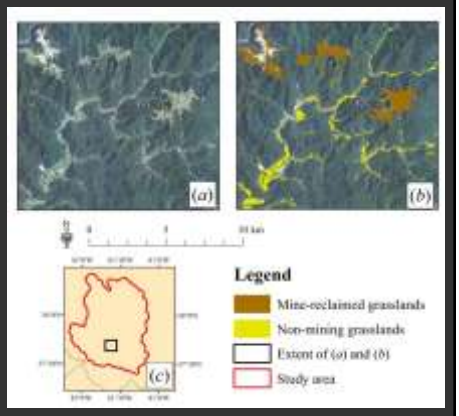
The prediction of numeric or continuous data is commonly termed regression, such as linear regression, machine learning regression, or deep learning regression.

Training data will consist of input predictor variables and a dependent variable that is continuous, such as percent canopy cover, amount of biomass, amount of carbon, concentration of a pollutant, median income, etc.

Predicting a Categorical or Binary Outcome

- Logistic regression
- Maximum likelihood
- Parallelepiped
- Minimum-Distance-to-Means
- Machine learning
- Deep learning





15

Categorical predictions involve assigning features to categories, such as types of land cover, types of forests, types of wetlands, types of animals, etc.

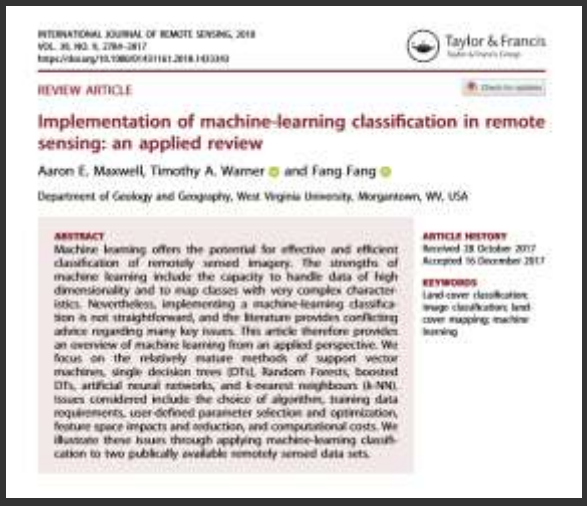
Binary classifications are a special case where there are only two classes being separate, such as forest vs. not forest or wetland vs. not wetland. It is also common to attempt to separate a feature from the background. In such circumstances, we commonly use terms such as positive vs. negative, presence vs. absence, foreground vs. background, or yes vs. no.

A variety of techniques are available to make categorical predictions. For example, the parametric methods maximum likelihood, parallelepiped, and minimum-distance-to-means have traditionally been applied to image classification and land cover mapping tasks. Logistics regression is a common technique for binary classification.

Machine learning and deep learning have also been shown to be useful for classification problems. In fact, they often outperform traditional, parametric methods.

Most of the examples that we will discuss in this course relate to classification problems.

Example: Land Cover Classification



INTERNATIONAL JOURNAL OF REMOTE SENSING, 2018
VOL. 39, NO. 9, 2784–2817
https://doi.org/10.1080/01431161.2018.1433343

Taylor & Francis

REVIEW ARTICLE

Implementation of machine-learning classification in remote sensing: an applied review

Aaron E. Maxwell, Timothy A. Warner and Fang Fang

Department of Geology and Geography, West Virginia University, Morgantown, WV, USA

ABSTRACT

Machine learning offers the potential for effective and efficient classification of remotely sensed imagery. The strengths of machine learning include the capacity to handle data of high dimensionality and to map classes with very complex characteristics. Nevertheless, implementing a machine-learning classification is not straightforward, and the literature provides conflicting advice regarding many key issues. This article therefore provides an overview of machine learning from an applied perspective. We focus on the relatively mature methods of support vector machines, single decision trees (DTs), Random Forests, boosted DTs, artificial neural networks, and k-nearest neighbours (k-NN). Issues considered include the choice of algorithm, training data requirements, user-defined parameter selection and optimization, feature space impacts and reduction, and computational costs. We illustrate these issues through applying machine-learning classification to two publically available remotely sensed data sets.

ARTICLE HISTORY
Received 28 October 2017
Accepted 16 December 2017

KEYWORDS
Land-cover classification; image classification; land-cover mapping; machine learning

Maxwell, A.E., Warner, T.A. and Fang, F., 2018. Implementation of machine-learning classification in remote sensing: An applied review. *International Journal of Remote Sensing*, *39*(9), pp.2784-2817.

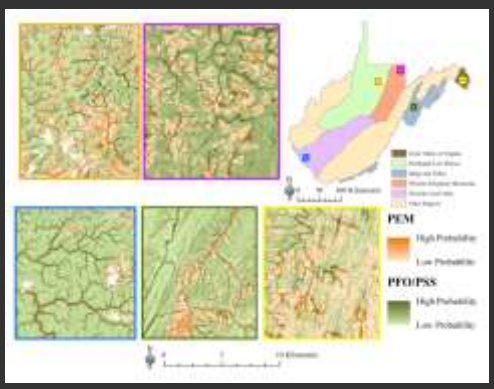
https://www.tandfonline.com/doi/full/10.1080/01431161.2018.1433343

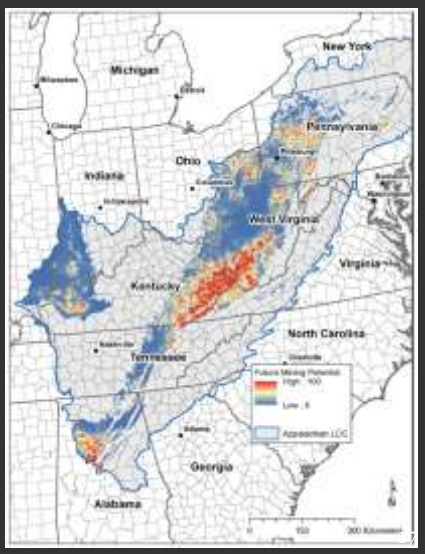
16

This slide provides a link to an open-access review article on the use of machine learning in remote sensing.

Predicting a Probabilistic Outcome

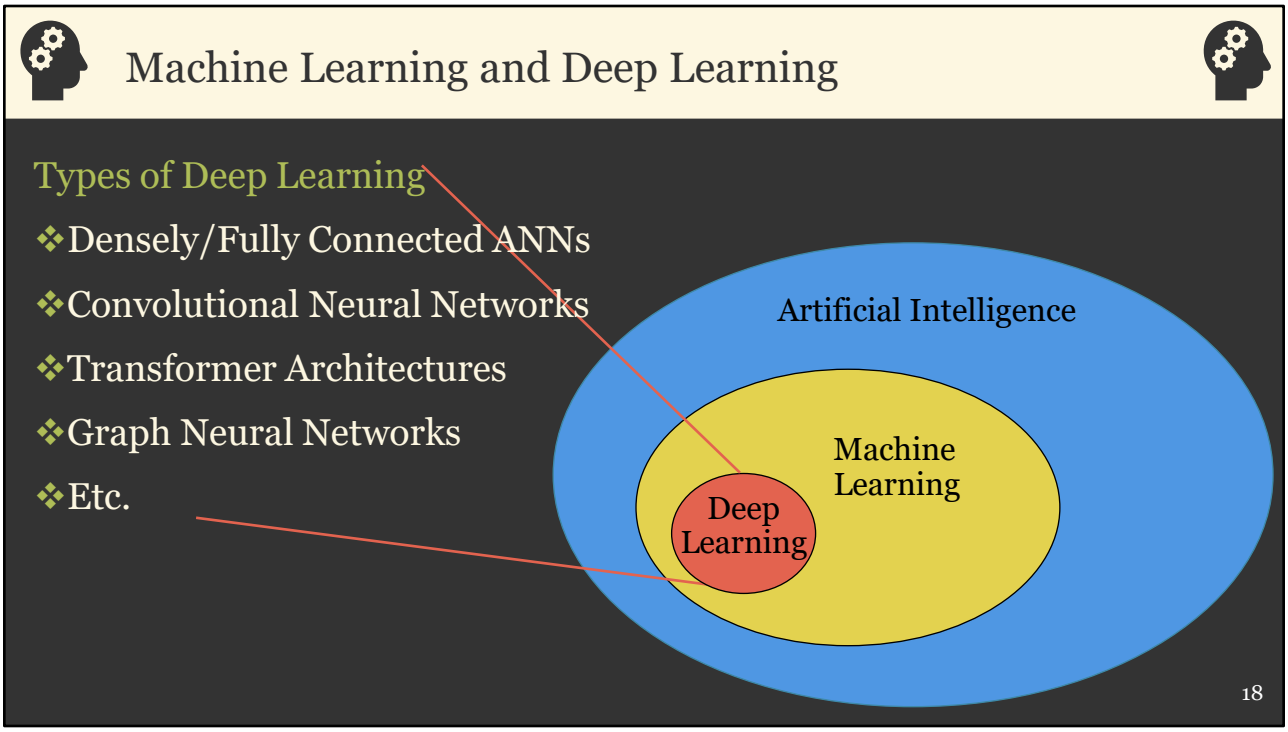
- Logistic regression
- Maxent
- Machine learning
- Deep learning





Instead of predicting classes, it is also possible to predict the probability or likelihood of a feature belonging to a specific class. Such techniques are similar to classification problems, except that a probability, or probability estimate, is returned as opposed to a "hard" classification.

When deep learning models predict categories, the result is generally a probabilistic prediction as opposed to a hard classification. The hard class is then derived from the probabilities using a defined probability threshold or returning the class with the highest predicted probability or logit. So, deep learning is well suited for probabilistic modeling or prediction.



What is the difference between machine learning and deep learning? Deep learning is not different from machine learning. Instead, it is a special type of machine learning that relies on artificial neural networks with many hidden layers. Another common characteristics of deep learning methods is that model parameter updates are made using the backpropagation approach. In short, deep learning is a branch of machine learning.

There are many machine learning methods that do not make use of deep artificial neural networks, such as tree-based methods (e.g., decision trees, boosted decision trees, and random forests) and kernel-based methods (e.g., support vector machines).

There are also sub-types of deep learning, such as fully connected neural networks, convolutional neural networks, graph neural networks, and Transformers.

The Supervised Learning Process

The Process

1. Develop training data
2. Develop predictor variables
3. Develop testing data
4. Do any pre-processing
5. Optimize and run algorithm
6. Use model to predict to validation data
7. Validate/assess the model
8. Predict to entire spatial extent
9. Save out the results

20

This slide outlines the process used to apply machine learning and deep learning methods to make spatial predictions.

Since deep learning relies of supervised classification, you will need to develop training data, such as labels and associated predictor variables. The nature of these data will depend on the methods applied.

In order to validate your models, you will also need a second set of withheld testing data. It is also common to use a separate withheld validation set to assess the model at the end of each training epoch, or iteration over the training set.

Data must then be preprocessed so that they are suitable as input to the specific modeling technique. This could consist of centering and scaling data, generating tables, or generating image chips and associated labels or masks.

Once a final model is generated by the learning process, it needs to be evaluated using the testing data.

Once a suitable model is obtained and validated, it can be applied to new data to make predications and maps.

We will discuss this process in more detail in later modules, and you will see examples of this workflow as implemented with PyTorch.

Developing Training Data

- Spatially explicit
- Representative of the population
- Adequate number of samples
- Adequate number of samples per category
- Accurate

21

Supervised learning requires training samples. These samples represent examples of the classes or values you are trying to predict.

The algorithm will then use these examples and the predictor variables at these locations to make a model that can be used to predict new locations.

I have found that the quality of the training samples has a large impact on the quality of the prediction.

When creating training samples, they need to be spatial explicit, or you need to know where they exist in the map space.

They also must be representative of the population. For example, if you are trying to predict where a certain tree might grow, and it is known to grow on both ridges and floodplains, then you need to give the algorithm examples of known locations where the tree has been found to grow in both floodplains and along ridges.

You need to provide an adequate number of samples and an adequate number of samples per class. The number of samples required will vary based on the complexity of the problem and the methods used. I generally try to provide as many quality training samples as possible given limitations and constraints.

Collecting a large number of training samples can be difficult due to time, cost, and access constraints.

Lastly, the data should be accurate. If the algorithm is given mislabeled or poor examples, it will have difficulty creating a useful model. Garbage in, garbage out.

A Note on Pseudo-Absence Data

- Some algorithms require both presence and absence data
- If you don't have absence data, you can develop pseudo-absence data
- These can be random locations that are unlikely to be presence locations

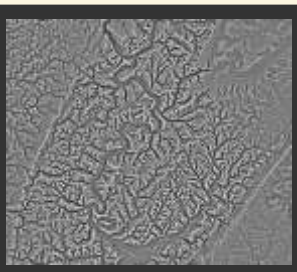
22

For binary classifications, it is common to only have examples of the truth or presence class. However, most algorithms require examples of both true and false, or presence and absence data.

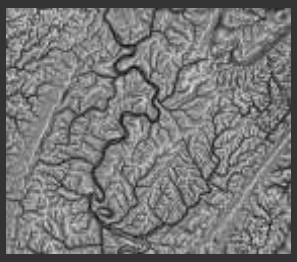
If absence data are not available, it is sometimes necessary to generate pseudo-absence data, such as random background points.

This generally requires some assumptions and can be difficult to implement.

Developing Predictor Variables



Slope Position



Dissection

Roughness

- Spatially explicit
- Accurately georeferenced
- Accurate/precise
- Timely
- Adequate spatial resolution
- Available

23

It would make sense to provide as much information or as many predictor variables as possible. For example, if you are buying a car, more information will make you a more informed consumer.

However, this has generally been found to not be the case in predictive modeling. This is known as the Hughes Phenomenon or "Curse of Dimensionality". This suggests that adding more predictor variables can decrease the performance of the model. This is because, even though more information is being provided, the complexity, or dimensionality, of the problem increases.

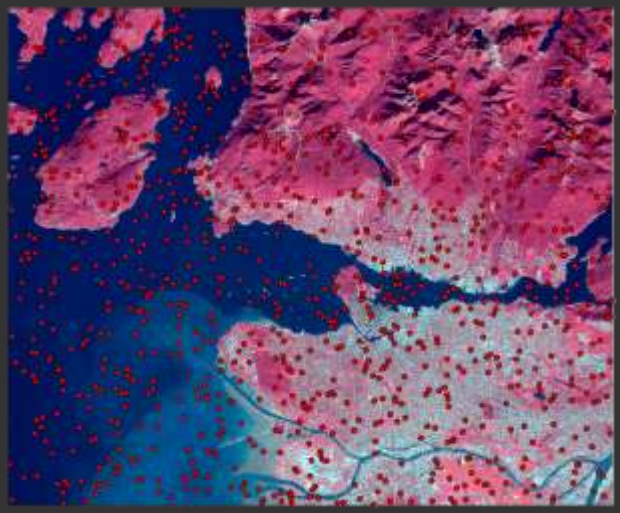
This problem is generally more pronounced when a small training set is used. This is because more samples will need to be provided to deal with the complex dimensionality of the problem.

Fortunately, some algorithms are robust to this problem. Also, methods are available to reduce the number of variables or select the most useful variables. This is outside the scope of this course.

Deep learning algorithms are generally not provided a large number of predictor variables or a large feature space. Instead, the deep network is expected to learn data abstractions from the input predictor variables.

Developing Testing or Validation Data

- Unbiased
- Randomized
- Non-overlapping with your training data
- Correctly proportioned
- Accurate



24

Testing or validation data must also be provided to assess the model performance and map output. In deep learning, validation data refer to withheld samples used to assess the model at the end of each training epoch while testing data are used to assess the final model.

In order to produce an unbiased estimate of model performance, the testing and validation data must be randomized in some way.

Also, the training and validation data should not overlap with each other or the testing samples, or the same samples should not be included for both training and assessment. This is because algorithms tend to do a better job of predicting the training samples as opposed to new locations, a phenomenon known as overfitting. So, including training samples as testing or validation samples could inflate the reported performance.

It is also recommended that testing and validation samples be correctly proportioned relative to the map. For example, if you are mapping land cover and 70% of the mapped area is forest, then 70% of the testing and validation samples should also be forest.

Lastly, testing and validation data should be accurate. The goal here is to compare the map product to reference data of higher quality. It is generally

assumed that no data are perfect. So, even the testing and validation data will have some error. We try to avoid using the terms “ground truth” or “ground truthing” for this reason.

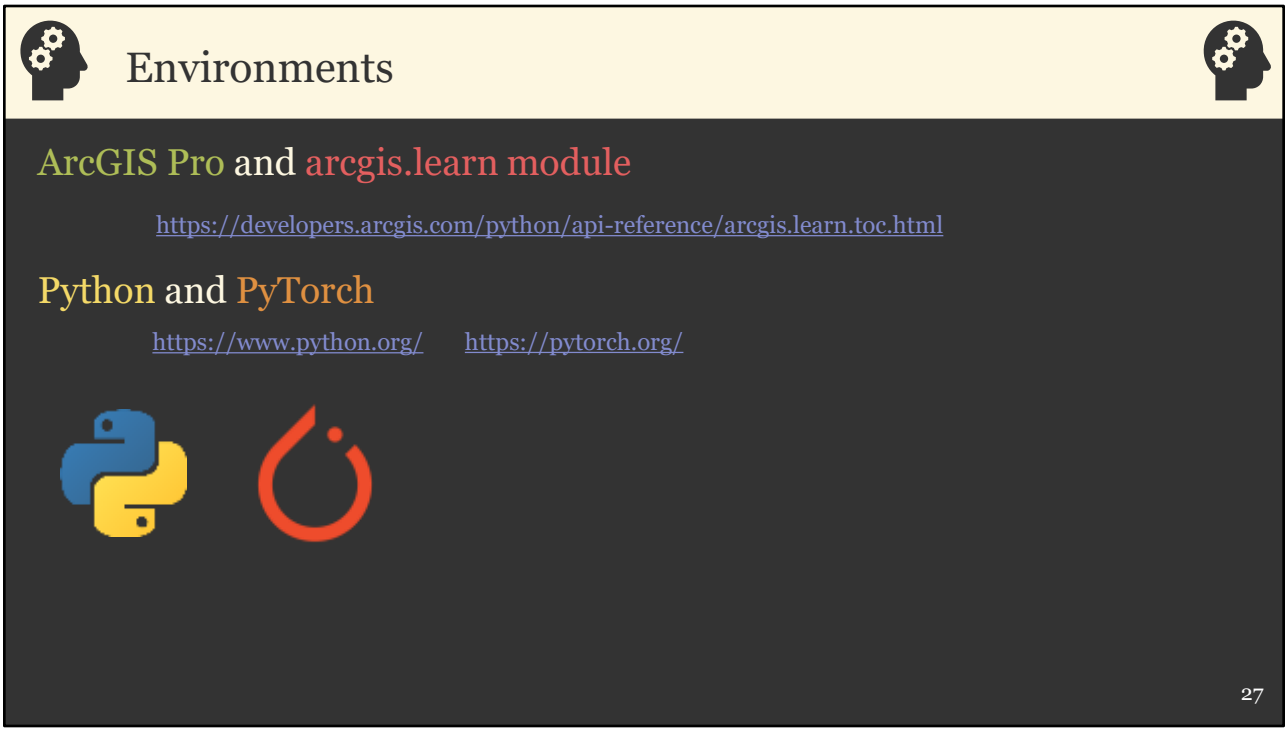
In later modules, we will discuss model training and assessment methods specific to deep learning.

Working in Code

Deep learning data preparation, training, validation, and implementation are commonly conducted using scripting or code. Currently, Python is the most used language for deep learning research and application. R can also be used, but it generally relies on Python for deep learning implementation and is not as robust. We will also explore some examples using ArcGIS Pro that do not require any coding.

When you work in code, it is generally a good idea to comment and document your code. If you use code generated by another analyst or scientist, you should cite it.

To support reproduceable science, it is generally good to share your data and code if possible. For example, you could make a GitHub repository for your project.



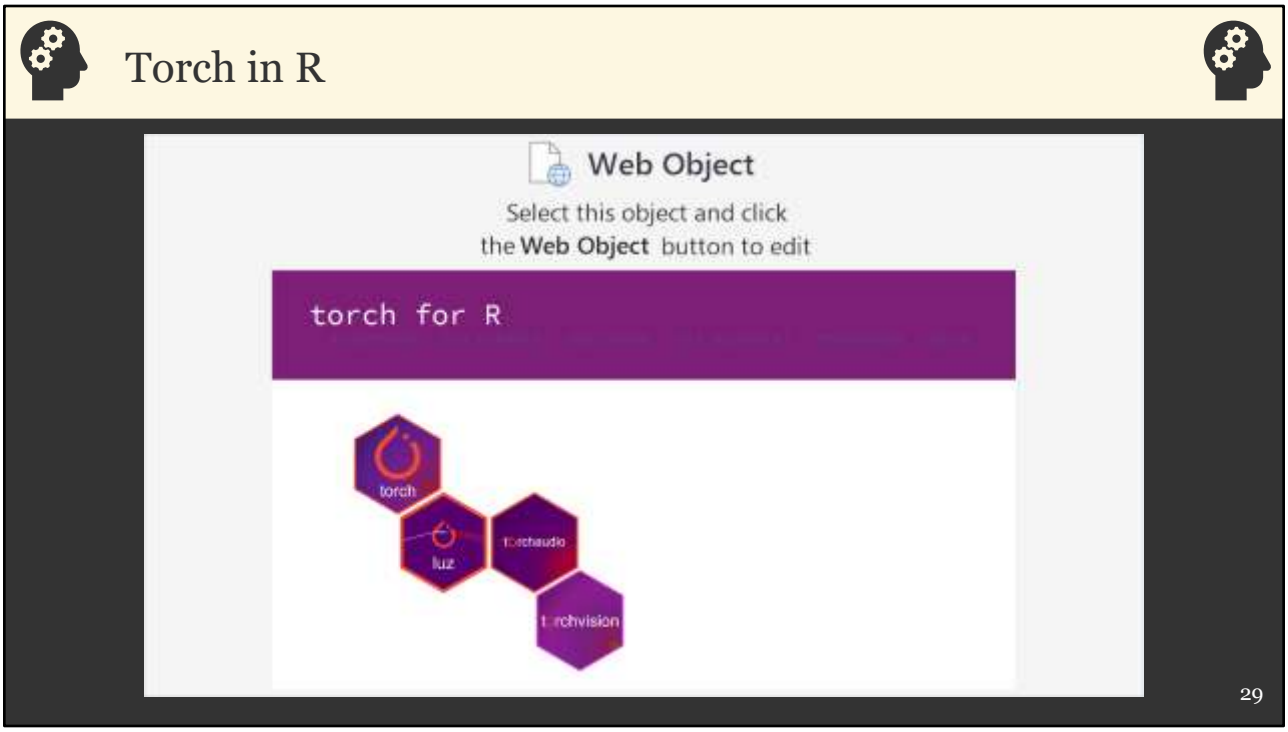
Again, I will provide examples using ArcGIS Pro and Python/PyTorch.

When using Python, it is common to use Anaconda to set up a research environment that has access to all the required packages, libraries, or modules needed to support deep learning, such as PyTorch, NumPy, and Pandas. I will provide a video that steps through this process. Luckily, all of these technologies are open-source and free.

PyTorch Ecosystem

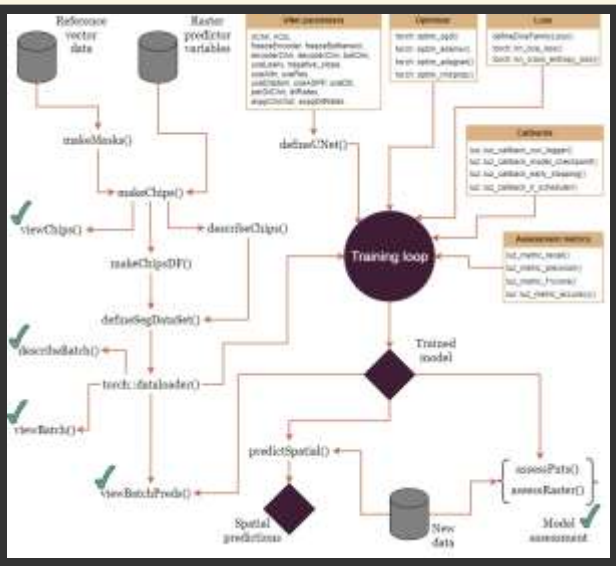
28

A large set of packages or libraries have been developed to build upon the functionality of PyTorch and/or make it easier to apply deep learning for specific tasks. This website provides links to these packages.



At the time of this writing, deep learning is more developed in Python than it is in R. However, the deep learning options in R continue to grow. For example, the torch package is being developed to provide a PyTorch-like implementation in R. Since the Python/PyTorch environment is more developed, we will focus on Python in this class.

geodl in R



https://github.com/maxwell-geospatial/geodl

https://mlverse.github.io/luz/

https://torch.mlverse.org/packages/

30

The author of this course is working on developing an R package to support geospatial semantic segmentation. This package is called geodl and is built on top of the torch, luz, and terra packages.

The torch package implements deep learning in the R environment and does not require using a Python Environment. Instead, it uses R and a C++ backend: libtorch. The luz packages makes training deep learning models easier. As you will see throughout this course, the training process can be complex and error prone. Using luz helps reduce opportunities for issues or errors. The terra packages is used to read and work with raster geospatial data in R.

We will not cover geodl in this course. Instead, it is discussed in our *Geospatial Supervised Learning* class.

Keras and Tensorflow

- Python API for deep learning
- Code in Python as opposed to C++
- Also available for R
- Run on CPU or GPU
- Building blocks to create, train, and use a variety of DL models
- Interface with multiple backends including Tensorflow and PyTorch

https://keras.io/ https://www.tensorflow.org/ https://github.com/microsoft/CNTK

https://keras.rstudio.com/ https://github.com/Theano/

31

Keras is an application programming interface (API) written in Python. It allows users to develop their models and experiments using Python as opposed to a more complex language, such as C++.

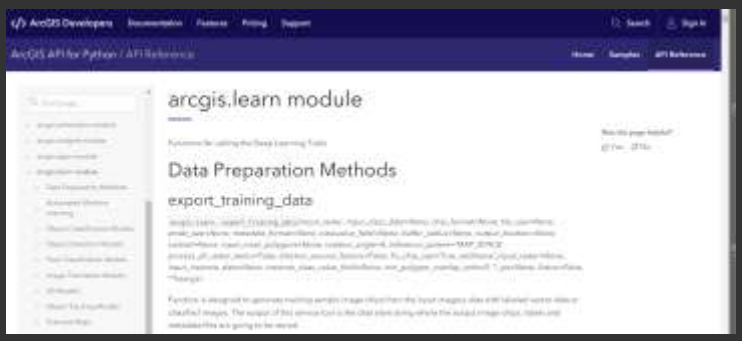
Keras provides an interface to multiple backends, including Tensorflow and PyTorch.

Keras was originally developed for Python, but there is now a Keras API for R that relies on the Python API. So, you can now use Keras in R, but you will need to set up a Python environment that R can connect to.

I have chosen to use PyTorch as opposed to Keras/Tensorflow here based on current trends and PyTorch's focus on research applications.

ArcGIS Pro

- Image Analyst/Spatial Analyst
- Generate training data
- Train models
- Predict data with models
- Raster functions
- arcgis.learn module



https://developers.arcgis.com/python/api-reference/arcgis.learn.toc.html

32

If you do not want to learn to code or further develop your coding skills to apply deep learning to your data, ArcGIS Pro now includes deep learning tools to create training data, train models, and apply models. These tools are available with a valid Image Analyst extension license.

You can also use ArcGIS Pro deep learning functions in Python scripts by referencing ArcPy and the arcgis.learn module, which provides functions and classes for deep learning.

QGIS

- GeoAI plugin for QGIS

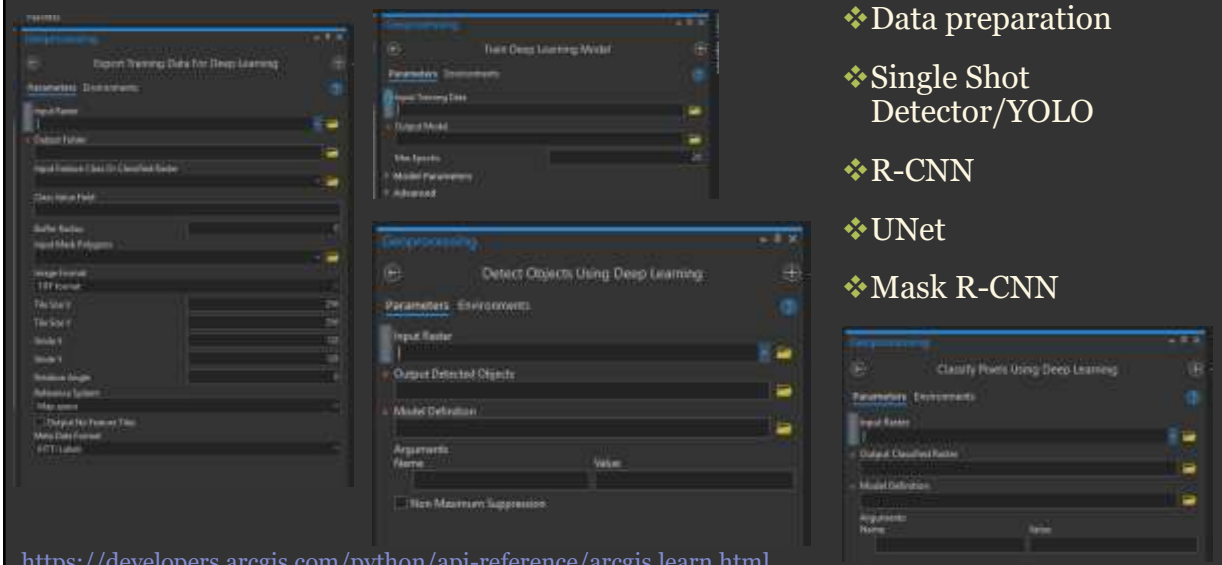
https://opengeoai.org/qgis_plugin/

https://plugins.qgis.org/plugins/geoai/

33

There is also a free and open-source GeoAI plugin for QGIS. It will not be explored in this course; however, I recommend that you check it out.

ArcGIS Pro



- Data preparation
- Single Shot Detector/YOLO
- R-CNN
- UNet
- Mask R-CNN

https://developers.arcgis.com/python/api-reference/arcgis.learn.html

34

At the time of this writing, ArcGIS Pro and the argis.learn module include functions, classes, and tools to perform object detection, semantic segmentation, and instance segmentation.

There are also tools available to train models, apply models, and develop training data, such as image chips and associated labels or masks.

Hardware

- Solid state storage
- Decent CPU
- Lots of RAM (>16 GB)
- CUDA-Enabled NVIDA GPU

https://developer.nvidia.com/cuda-toolkit
https://developer.nvidia.com/cuda-gpus

https://www.nvidia.com/en-us/geforce/graphics-cards/rtx-2080-ti/

35

Again, deep learning can be very slow on CPUs, even high-end, multi-core CPUs.

If you plan to work with deep learning, and especially if you want to use convolutional neural networks, a graphics card is required. Currently, NVIDIA is the only graphics card company that has invested in deep learning. So, you will specifically need an NVIDIA GPU that is CUDA-enabled. The links on this page provide a list of CUDA-enabled graphics cards and also links to required toolkits, which are currently free.

GPUs can be very expensive. However, it is possible to do some deep learning on fairly inexpensive GPUs built for gaming computers. It is also possible to install multiple GPUs. For example, you can purchase deep learning workstations that include 4 GPUs.

Other than the GPU, it is generally recommended to use a solid-state hard drive, since this will improve transfer and processing speed. You should have at least 16GB of RAM available and a decent CPU.

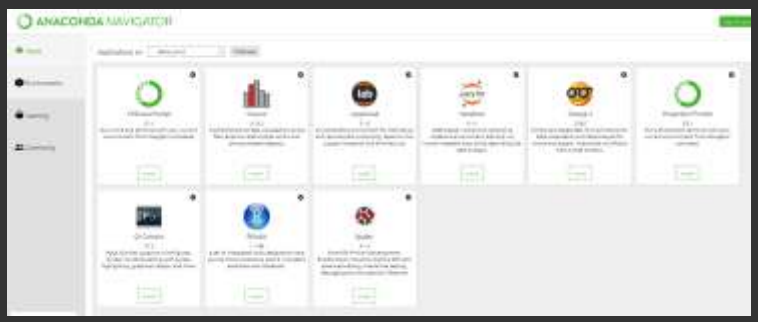
If you do not want to set up your own hardware, another option is to set up a virtual machine from a service provider, such as Microsoft Azure or Amazon AWS. Unfortunately, this can get expensive since pricing is generally based on

computational time.

Lastly, Google Colab provides free access to GPU resources; however, availability is based on demand and there are processing time/computational load limitations.

Working in Python

- Set Up Anaconda environment
- Data science libraries/modules = pytorch, numpy, pandas, matplotlib, scipy, scikit-learn, scikit-image
- Geospatial libraries/modules = geopandas, rasterio

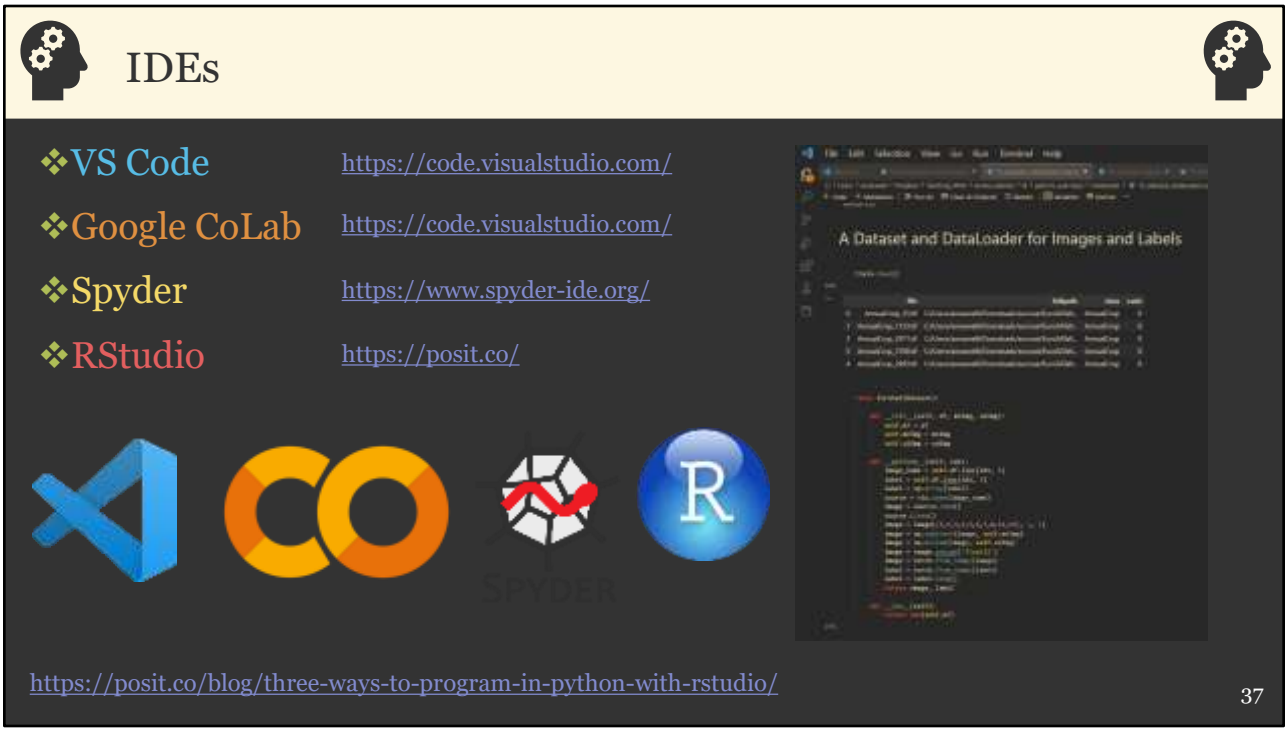


36

As mentioned above, it is generally best to create an Anaconda Environment if you plan to use Python for data science and machine learning/deep learning.

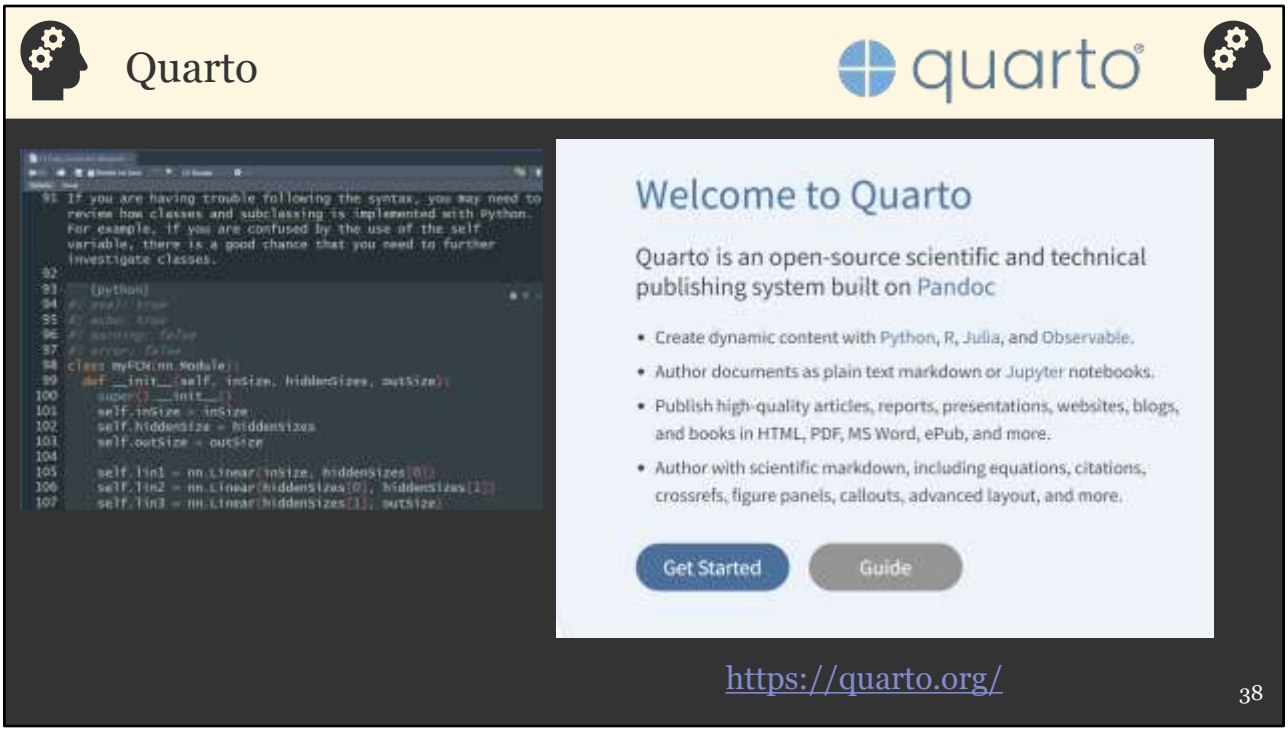
Several packages are required for data science with Python including NumPy, Pandas, and matplotlib. For deep learning, you will also need to install PyTorch. If you need to work with and process geospatial data, GeoPandas and Rasterio are useful.

One headache when conducting deep learning with Python is making sure that all the needed packages are installed into your environment and that they are the correct version. You will also need to make sure that Python is able to access your GPU. I have provided a video on this topic.



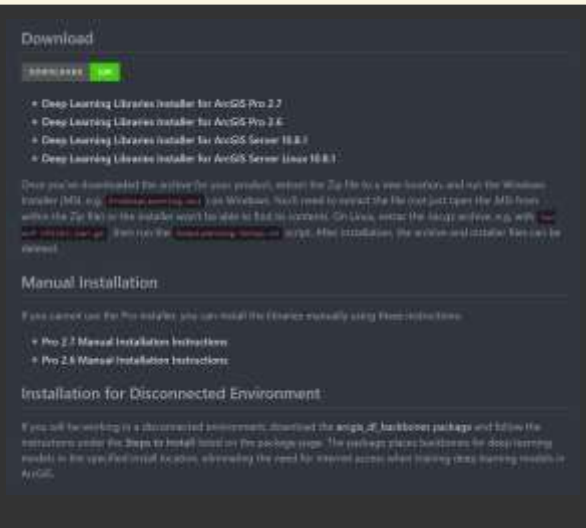
There are a variety of integrated development environments (IDEs) available for Python. I prefer either VS Code or RStudio, both of which are free. If you do not want to set up a local Python environment, you can make use of Google Colab; however, there are usage caps, as noted in a prior slide.

As you work through the course materials, feel free to use the IDE of your choosing.



Quarto is a tool that allows for rendering code and Markdown to final products, such as PDFs, Microsoft Word documents, HTML webpages or websites, books, and presentations. I used Quarto to render the PyTorch example modules for this class and have made the Quarto documents available for download on the course webpage.

Deep Learning in ArcGIS Pro



https://github.com/Esri/deep-learning-frameworks

39

It is also possible to use the built-in Python environment that installs with ArcGIS Pro. However, you will need to update it to include the needed deep learning libraries. I have provided a video on this topic.

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

Thanks! Hope you found this useful.