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

Remote Sensing

GeoAI

This module provides an introduction to uses of GeoAI in remote sensing. Note that we have an entire course focused on geospatial deep learning on the West Virginia View website: Geo*spatial Deep Learning*. Deep learning is also explored in our *Geospatial Supervised Learning* course.

This module is a very general overview. We do not cover topics in-depth, only provide a simple introduction.

Module Topics

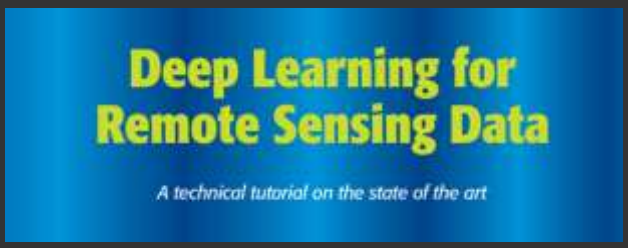
1. Machine vs. deep learning
2. Artificial neural networks
3. Image recognition or scene labeling methods
4. Object-detection methods
5. Semantic segmentation or pixel-level classification methods
6. Instance segmentation methods
7. Other use cases
8. Foundation models

2

These are the topics covered in this module.

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

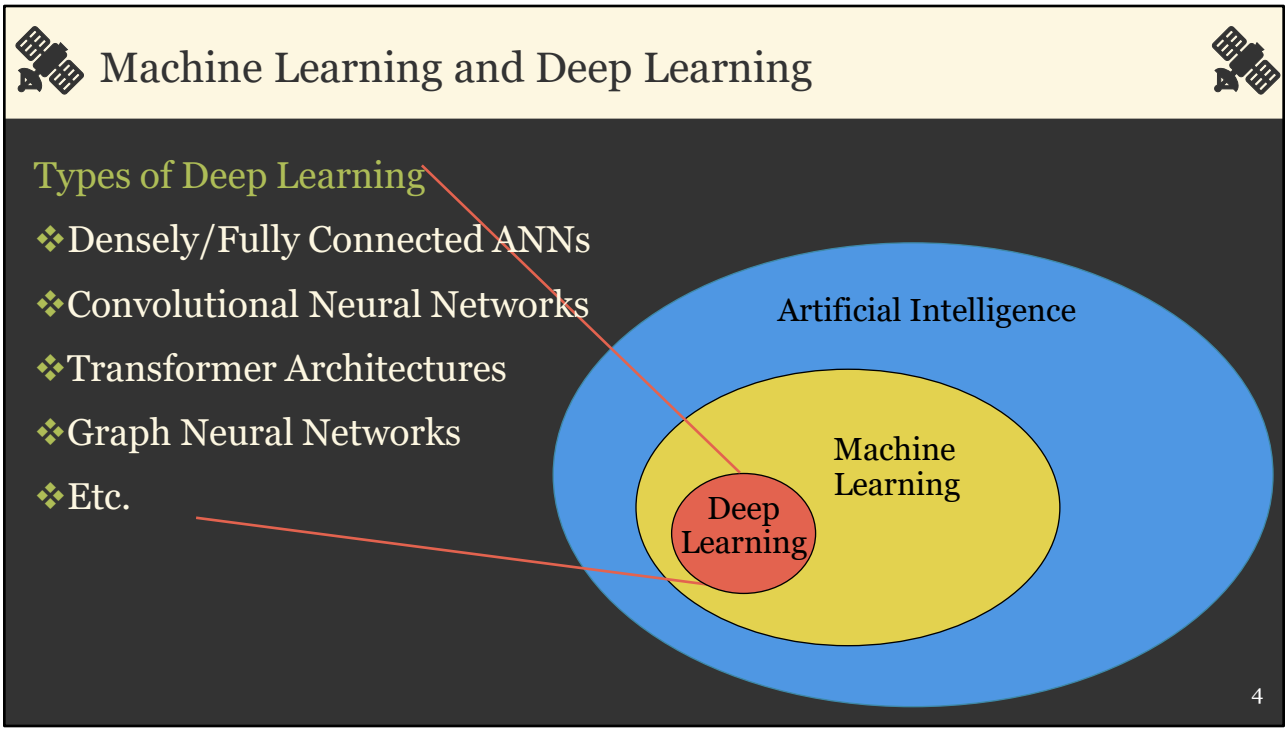
3

This slide lists some common uses of deep learning and GeoAI 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.

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

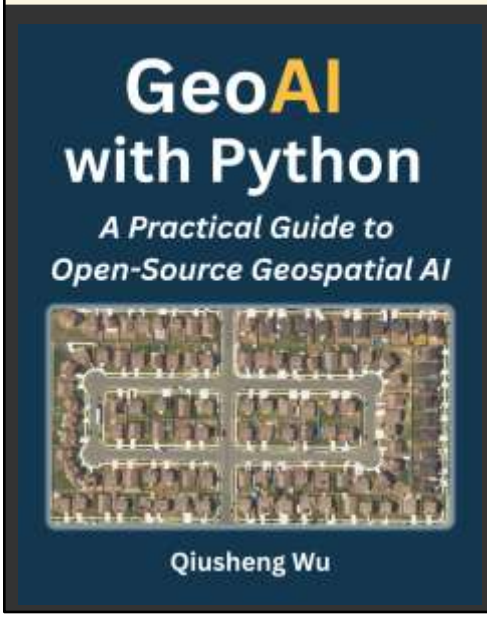


What is the difference between machine learning and deep learning? Deep learning is not different from machine learning. Instead, it is a special type of machine learning that relies on artificial neural networks (ANNs) with many hidden layers. Another common characteristics of deep learning methods 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.

Book Recommendation

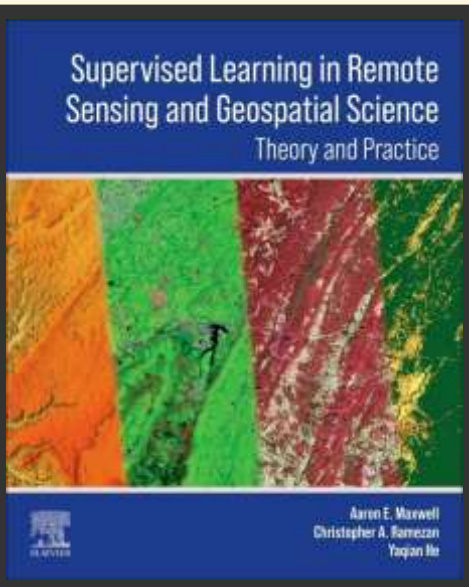


https://book.opengeoai.org/

5

This is an excellent text focused on GeoAI.

Book Recommendation



https://shop.elsevier.com/books/supervised-learning-in-remote-sensing-and-geospatial-science/e-maxwell/978-0-443-29306-1

https://www.wvview.org/gslr/

6

This text more generally discusses geospatial supervised learning and is a companion text to our *Geospatial Supervised Learning* course. The last section focuses on deep learning.

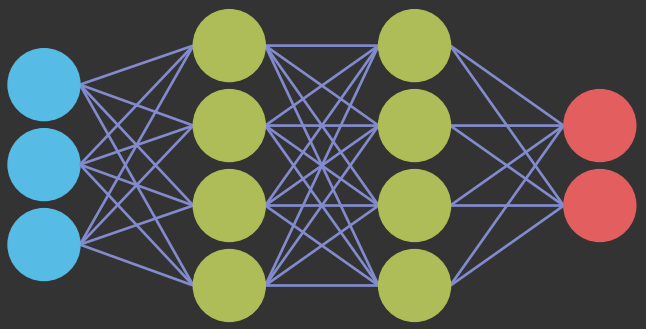
Artificial Neural Networks (ANNs)

7

Deep learning techniques build on artificial neural networks. As a result, we will discuss these architectures prior to exploring deep learning methods.

ANN Structure

- Consist of interconnected neurons or nodes
- Input layer represents predictor variables
- Output layer represents what is being predicted
- Model generated by learning a weight associated with each connection
- Referred to as fully connect architectures or densely connected architectures



8

ANNs are often described as being modeled after the human brain, as they consist of interconnect neurons or nodes. However, how these learning algorithms function is not necessarily comparable to the human brain, so I try to avoid using this analogy.

An ANN consists of neurons organized into layers as demonstrated on the slide. The Input layer represents input predictor variables, such as image bands, and has one neuron for each input, while the output layer represents the desired output, such as land cover categories, and has one neuron for each output class. For binary classification, one or two output neurons may be used. For regression, there will be one output neuron.

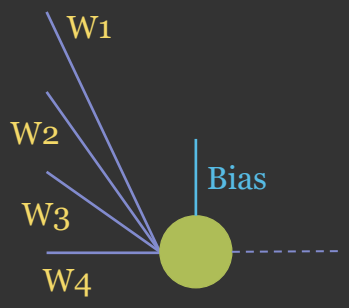
Between the input and output layers are one or more hidden layers containing multiple nodes. Within the network, all neurons in a layer are connected to the neurons or nodes in adjacent layers, and these connections have weights.

On the slide, the conceptualization contains three inputs (e.g., three image bands or predictor variables) and two hidden layers containing 4 nodes each. There are two classes being classified or predicted. Note that all nodes or neurons between adjacent layers are connected. These connections represent the weights.

Through the process of supervised learning, the weights in the network are adjusted in an attempt to improve the prediction of the output.

Weights

- Weights are adjusted during training process
- Similar to slope or coefficient in regression
- Inputs from prior layer multiplied by weights
- Multiple Input X weight combinations are summed
- Weights relate to strength of activation or signal



$$\text{Activation} = b + \sum_{i=1}^{n} x_i w_i$$

9

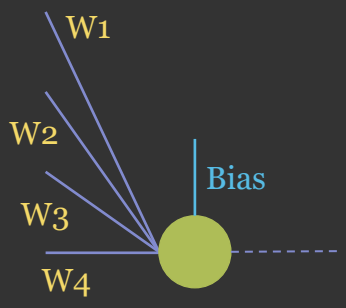
This slide further describes the concept of weights and how a signal or activation is produced at each node.

This is actually very similar to multiple linear regression. The weights are comparable to the coefficients for each predictor variable while the bias (b) is similar to the y-intercept.

So, at this point, the activation or output of the node is a linear combination of the inputs to the node, associated weights, and bias.

Bias

- A constant to shift the activation
- Similar to a y-intercept in linear regression



$$\text{Activation} = b + \sum_{i=1}^{n} x_i w_i$$

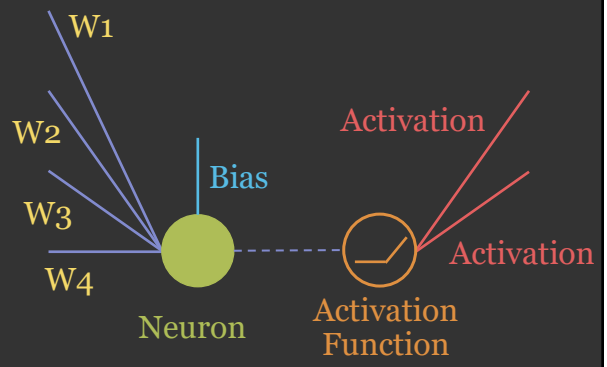
10

Again, the bias at each node is similar to the y-intercept in a linear regression equation. It is not multiplied by any input.

The bias allows the activation to be shifted to larger or smaller values, like an offset. The addition of a bias term further increases the flexibility of the model.

Activation Functions

- Allow for modeling of nonlinear relationships
- A mathematical gate between inputs and outputs



$$\text{Activation} = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

11

I left out an important component above in the discussion associated with weights, bias, and activation at each neuron or node. As I mentioned, this is similar to a multiple regression equation where the bias acts as a y-intercept, the weights act as coefficients, and the inputs act as the predictor variable values. In reality, this is more than similar to a multiple regression equation: it is a multiple regression equation.

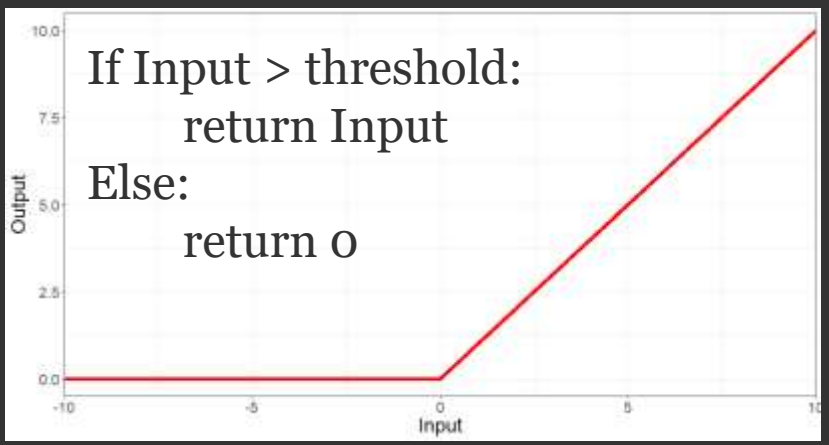
In order to be able to model complex patterns and relationships between variables and make predictions from a complex and noisy signal, it would be useful to be able to model or describe more complex patterns in the data.

This is accomplished by (1) having many interconnected nodes, associated weights, and hidden layers and (2) applying an activation function to transform the activation or signal (the output from the neuron).

Note that just having multiple neurons and multiple hidden layers is not enough to model non-linear relationships. A series of linear relationships is still linear. Thus, activation functions are required to model non-linear patterns and relationships.

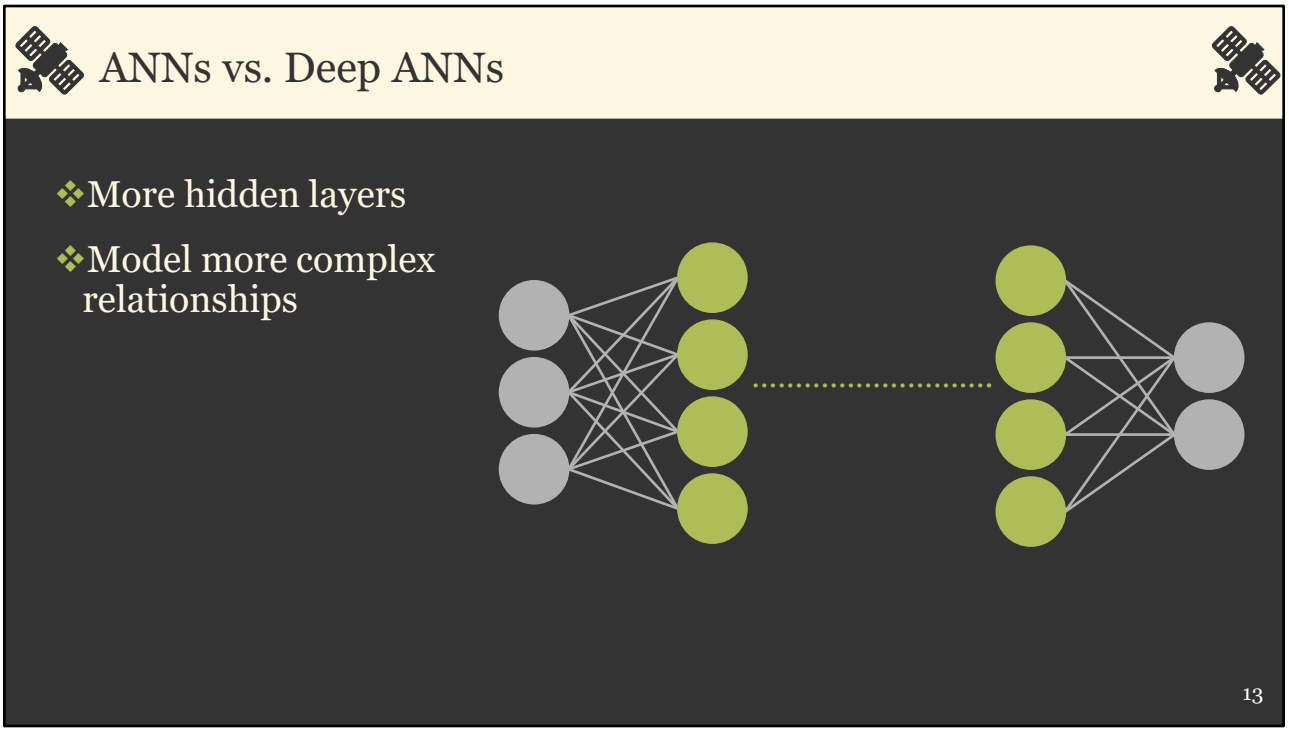
Rectified Linear Unit (ReLU)

- Nonlinear
- More computationally efficient than sigmoid or tanh
- Widely used
- Not zero-centered
- Not all neurons updated during backpropagation due to 0 gradient (dying ReLU)



12

The rectified linear unit (ReLU) activation function is currently commonly used. This allows for an activation of 0 below a certain threshold and a linear activation above a certain threshold. Generally, negative values are converted to 0 while positive values are maintained.



How is deep learning (DL) different from a traditional ANN? The key difference is the number of hidden layers, or layers between the input layer and output layer.

Deep learning algorithms generally have many hidden layers and associated nodes, as opposed to just one or two hidden layers.

So, you can think of "deep" as referring to the depth of the model or number of hidden layers.

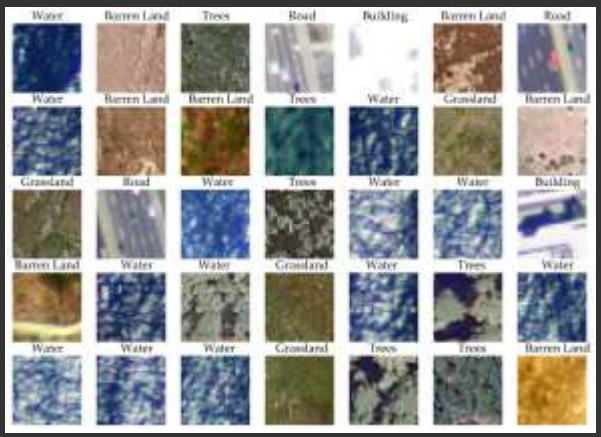
Interestingly, it has generally been shown that incorporating more layers is more effective than having fewer layers with a large number of associated neurons. One reason why this might be the case it that a larger number of layers allows for more data abstraction than a small number of layers with a lot of nodes or neurons.

Image Recognition

Problem Description

- Label an entire image to a single label

		Reference						
		Barren	Building	Grassland	Road	Tree	Water	UA
Classification	Barren	18,203	0	47	0	0	0	0.997
	Building	0	3,678	0	4	0	0	0.999
	Grassland	162	0	12,542	0	15	0	0.986
	Road	0	36	2	2066	0	0	0.982
	Tree	0	0	5	0	14,170	0	1.000
	Water	0	0	0	0	0	20,068	1.000
	PA	0.991	0.990	0.996	0.998	0.999	1.000	



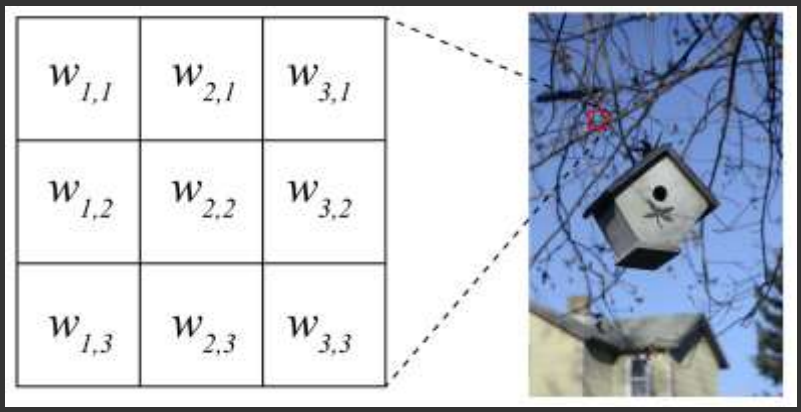
https://csc.lsu.edu/~saikat/deepsat/

15

Image recognition, labeling, or classification relates to labeling an entire image, as opposed to individual pixels, to one of a predefined set of classes.

What is Convolution?

- CNNs or ConvNets
- Learn spatial patterns and context
- Based on moving windows
- Apply moving windows to create feature maps



16

Convolutional neural networks, ConvNets, or CNNs are a sub-type of DL architectures that allow for the incorporation of spatial patterns into the learning process.

Instead of learning weights/parameters associated with links between neurons, as is the case for fully connected layers, convolutional layers learn weights to build a moving window or kernel that is passed over the image to manipulate the data and learn spatial patterns and abstractions.

Once the filter or moving window is learned, it is used to alter the values in the array and produce a feature map.

The basic idea behind CNNs is to learn spatial patterns at different scales.

This type of DL has led to many of the recent (i.e., over the last decade) improvements in computer vision tasks.

Convolution in Geospatial Science

- Moving windows, kernels, neighborhood analysis
- Examples in GIS = Focal Statistics, Majority Filter, Pixel Aggregation
- Examples in Remote Sensing = Texture, Edge Detection, Sharpening, Blurring, Statistical Filters, Focal Analysis

-1	-2	-3	-2	-1
-2	-3	-4	-3	-2
0	0	0	0	0
2	3	4	3	2
1	2	3	2	1

17

As a geospatial scientist or professional, you are likely already familiar with the concept and use of moving windows, as they are commonly applied to raster data, including images, to manipulate the data in some way. Such techniques are termed moving window analysis, kernel-based analysis, neighborhood analysis, or convolution.

For example, raster-based focal statistics techniques implemented in GIS allow for the calculation of a statistical measure within local windows while majority filtering is used to generalize data by returning the most commonly occurring value in a local neighborhood. Pixel aggregation is used to decrease spatial resolution by merging cells.

In remote sensing, moving windows are used to calculate textural measures, such as the measures after Haralick, detect edges, sharpen or blur images, and calculate local statistics.

Moving windows are a key component of working with raster data and images in general. For example, many digital photo effects make use of moving windows.

The example on the slide represents a Sobel filter that is used to highlight edges in a particular direction.

Convolution in Geospatial Science

-1	-2	-3	-2	-1
-2	-3	-4	-3	-2
0	0	0	0	0
2	3	4	3	2
1	2	3	2	1

18

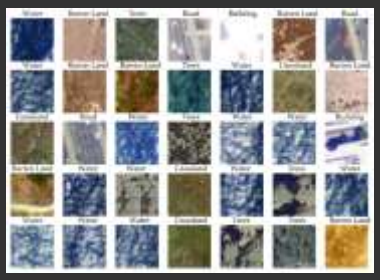
Here are some examples of window-based tool in remote sensing.

Note here that users can define their own filters by changing the values in the filter array or the shape or size of the array.

The key difference between these uses of filters and ConvNets is that the CNN learns these values or weights as part of the training process as opposed to the user defining them manually.

CNN Building Blocks

(1) Input image chips



(2) Learn weights to build kernels

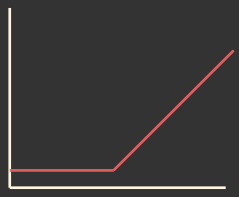
$w_{1,1}$	$w_{2,1}$	$w_{3,1}$
$w_{1,2}$	$w_{2,2}$	$w_{3,2}$
$w_{1,3}$	$w_{2,3}$	$w_{3,3}$

(3) Max pooling to learn at multiple scales

3	5	1	7
5	6	3	6
1	3	1	1
2	4	2	1

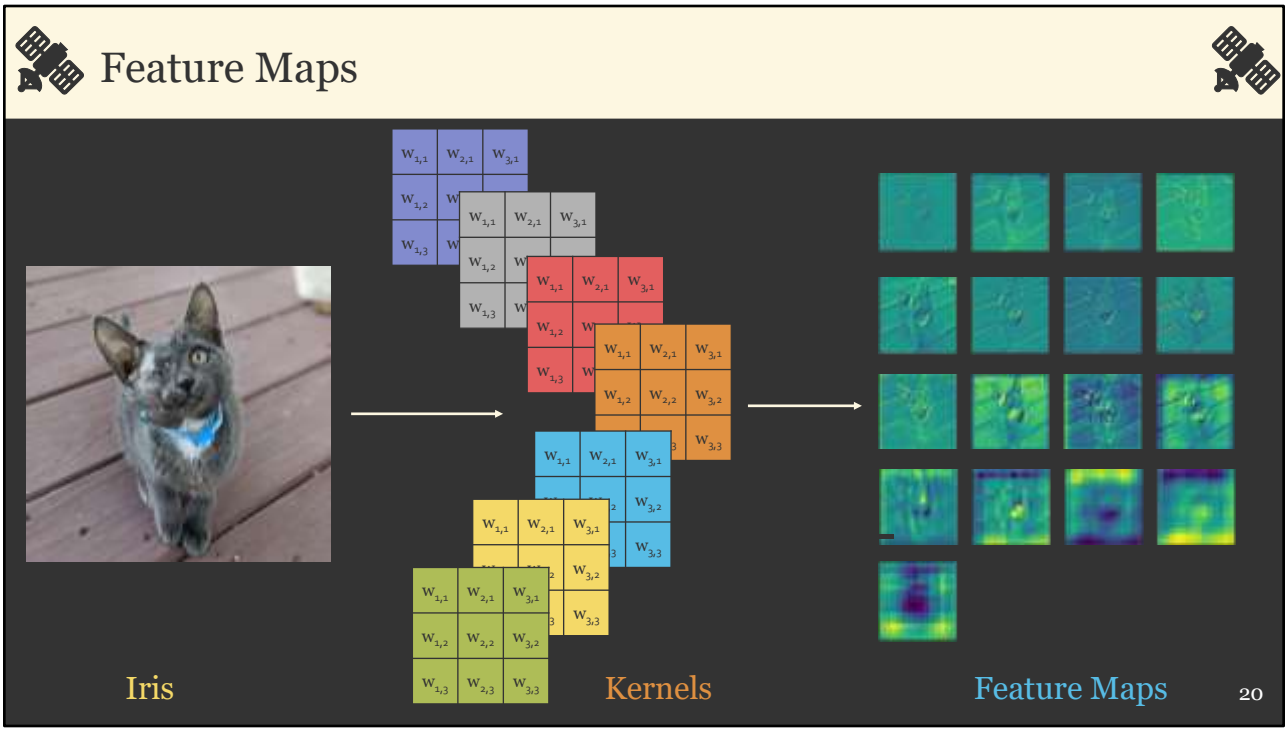
6	7
4	2

(4) Activation functions for non-linearity



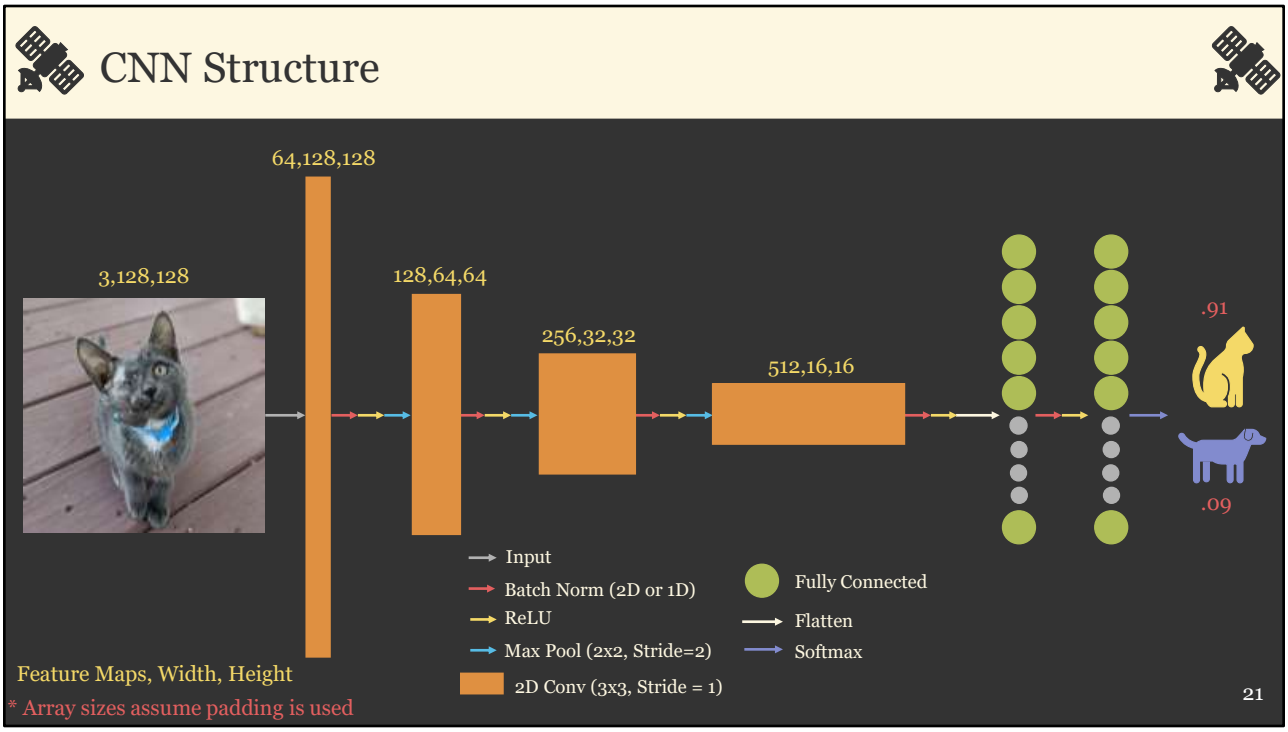
19

CNNs are generally built using a fairly simple set of operations or components including 2D convolutional, max pooling, and activation layers.

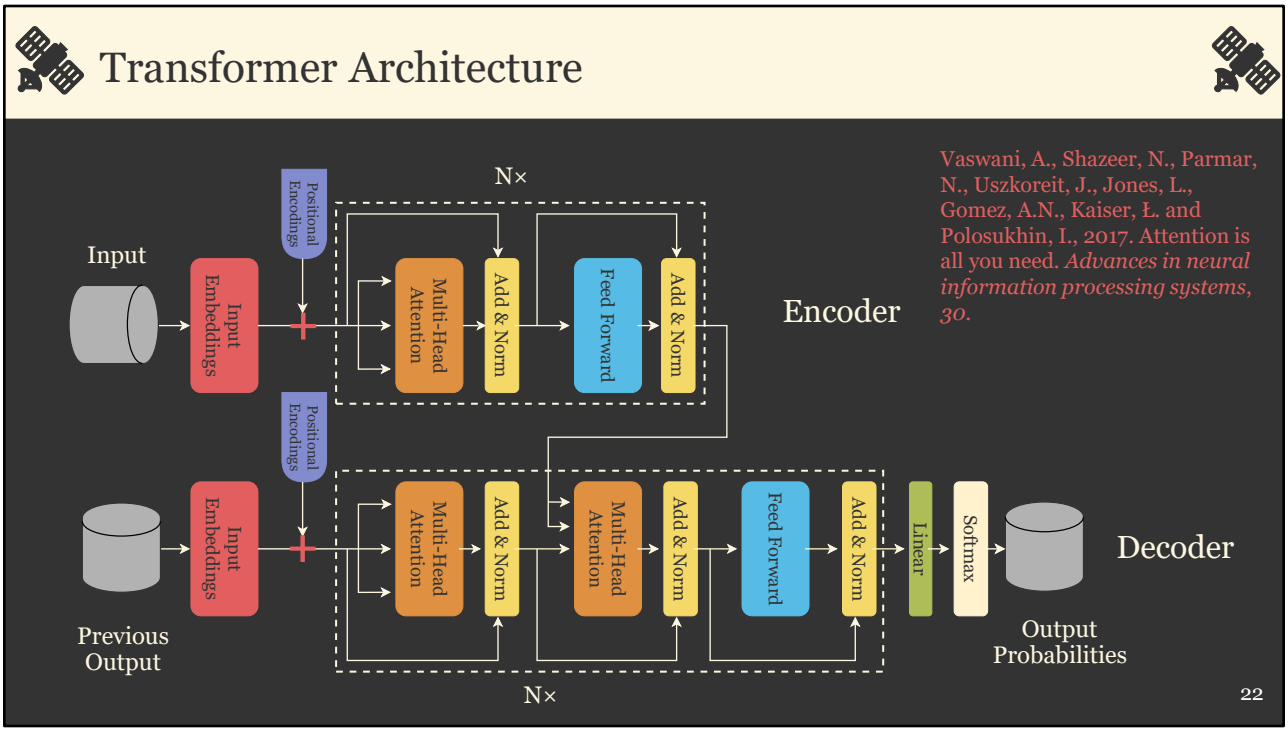


This slide visualizes some feature maps derived by multiplying the input image or prior feature maps produced by prior layers in the model by the trainable kernel weights and adding a trainable bias term. Again, the goal here is to model useful spatial information that can aid in the classification.

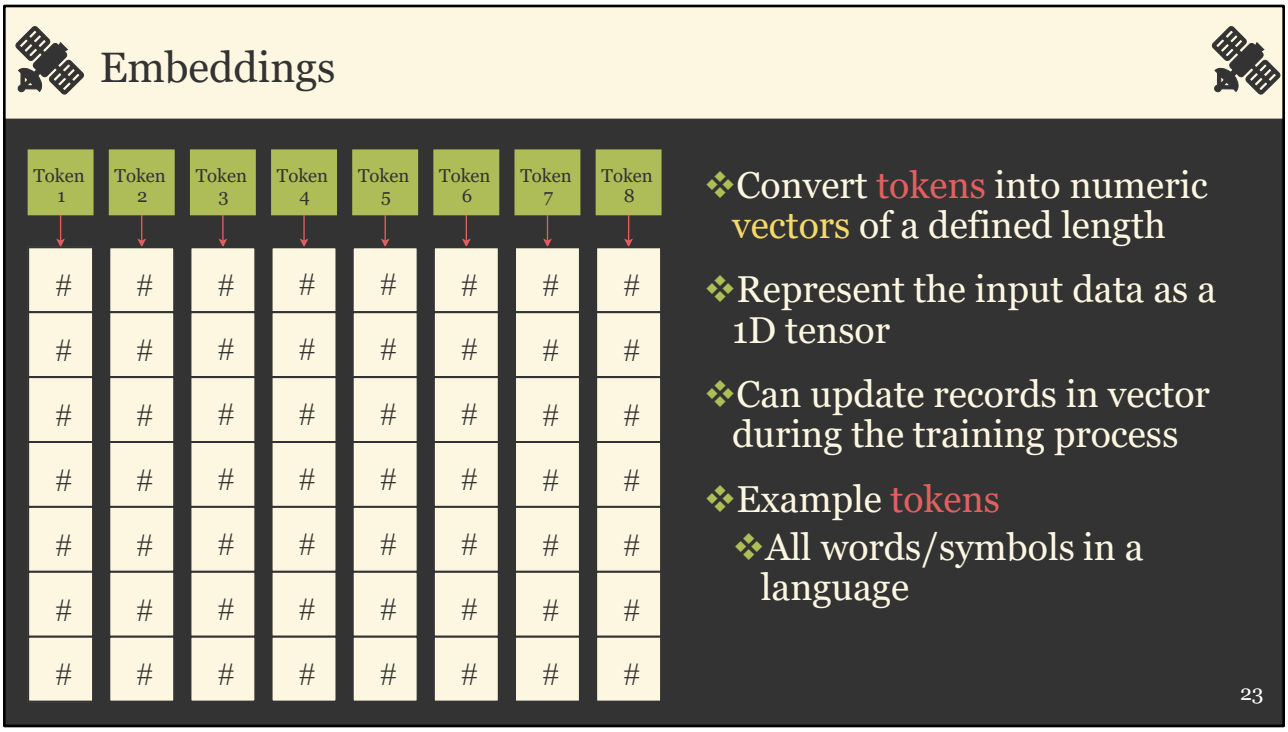
This is the central idea behind CNNs: learning weights associated with kernels allows for creating spatial abstractions of the data that can be useful for differentiating classes.



Scene classification CNN models use a series of convolutional layers to create spatial abstractions of input data. Between these layers, the resolution of the data are decreased in the spatial dimensions. This allow for learning spatial patterns at different spatial scales. The final set of pixel values are collapsed to a one-dimensional vector and provided as input to a fully connected layer. A series of fully connected layers are used to generate estimates of class membership at the scene level.

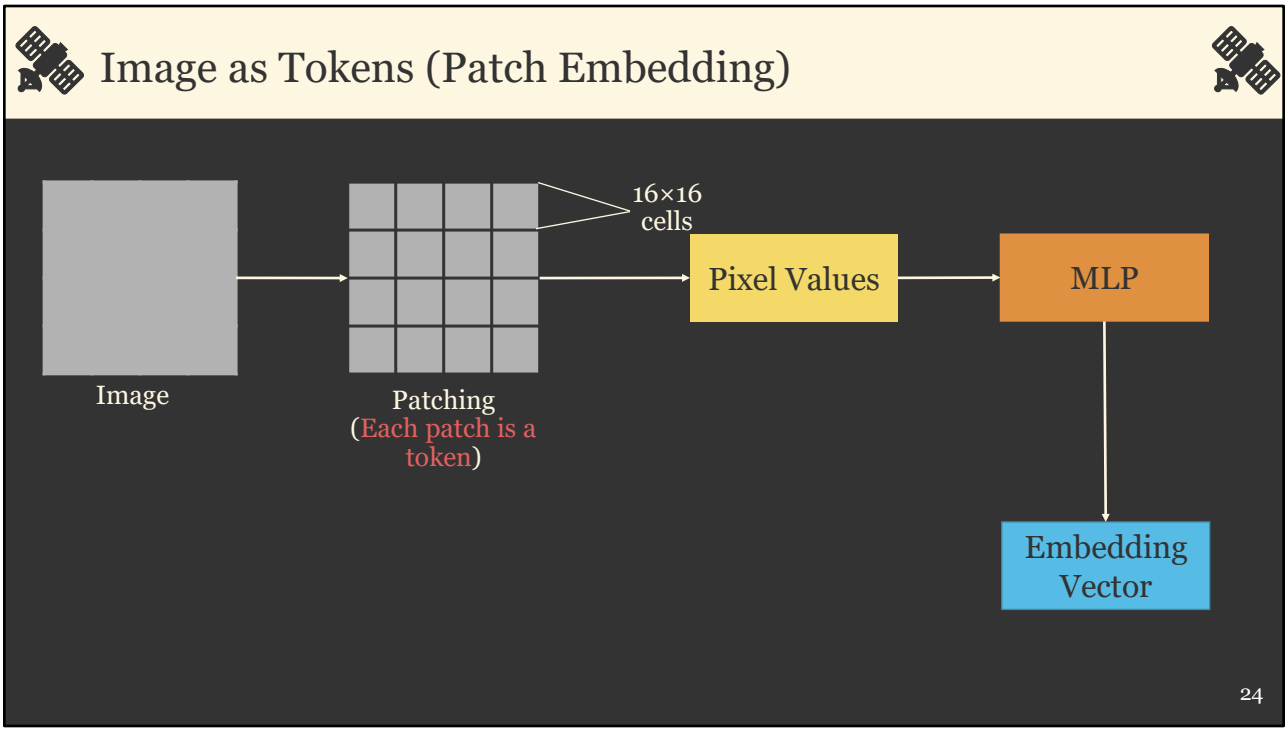


More recently, Transformer-based architectures have been adapted for vision tasks generally and scene classification tasks specifically. One of the key benefits of these architectures is the ability to model relationships between a larger number of image patches. The small size of convolutional filters is a limitation of CNNs, but limited receptive fields is generally not an issue for Transformer-based architectures.

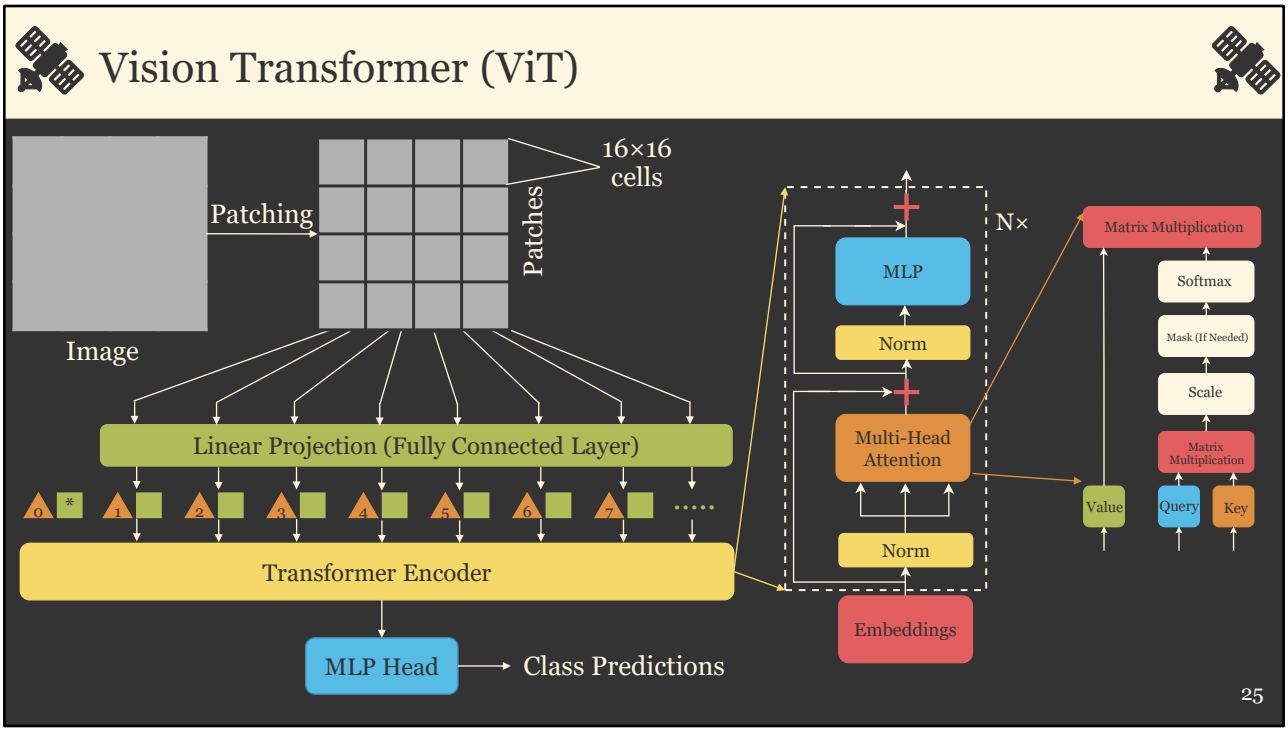


Input data for a transformer architecture must be represented as a vector. So, input data must be converted to embeddings. For natural language processing tasks, each individual word or token will be assigned a set of values, or a vector, of a defined length that will be used to represent it. The length of this vector will vary. In my example, each token or word would be represented using a vector containing 7 values. In a real model, the number of values stored in the vector would be much larger. These initially random values are updated during the training process to characterize key information about the token.

As already mentioned, each token represent a single item. For natural language processing, these are commonly words; however, other tokens are possible, such as punctuation, <start>, and <end> tokens. In short, all elements must be represented as a unique token, which is subsequently represented in the model as a vector encoding.



Patches of cells in an image can be represented as individual tokens with associated embedding vectors to serve as input to transformer-based models. The cell values within each patch are collapsed to a vector. This vector is then passed through a multilayer perceptron (MLP) or fully connected layer(s) to convert it into the embedding vector.



This slide conceptualizes the vision transformer, or ViT, architecture proposed in a 2020 paper. ViTs allow transformers to be applied to image or scene labeling tasks.

First, the image is broken into patches, each constituting 16×16 cells or pixels. These patches are treated as individual tokens that can be represented with a vector embedding. Additional positional information is added to these embeddings using an index.

Note that a new token is added with a position of 0. This is the classification token and represents the classes that are being predicted. All other tokens and their associated positional information correspond to image patches.

The tokens are then fed through a transformer architecture that captures spatial context information and models relationships between the image patches.

The multilayer perceptron (MLP) layer will accept the representation modeled by the transformer encoder to predict logits, which can be re-scaled to predicted class probabilities for each predicted class using a softmax function. A prediction will be made for each of the differentiated classes.

Object Detection

Problem Description

- Place bounding boxes around objects in an image

27

Object detection entails predicting a bounding box for each instance of the class or classes of interest occurring within an image extent.

Faster R-CNN Components


Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Abstract

1 Introduction

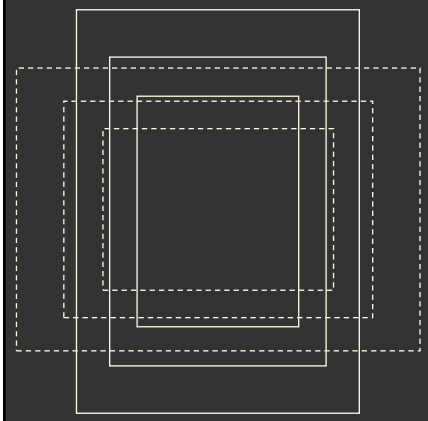
1. Convolutional Neural Network = learn feature maps
2. Region Proposal Network (RPN) = generate candidate regions within the image
3. ROI Align = improve alignment between ROI pooling layers and ROIs
4. Fully Connected Layers = for bounding box regression and class prediction

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

28

Faster region-based convolutional neural networks (R-CNNs) are a CNN-based architecture designed for object detection tasks. This slide describes the components of this architecture.

Anchor Boxes



- Serve as starting point bounding boxes
- Can have varying centers within image, sizes, and aspect ratios
- Anchor boxes that are predicted to contain an object of interest can be further refined by object detection algorithms

29

Anchor boxes serve as starting points for generating region proposals or bounding boxes to be refined.

A set of anchor boxes are defined with varying sizes, orientations, and aspect ratios. The image on the slide shows six boxes; however, faster R-CNN commonly uses nine boxes. These anchor boxes are then passed over the input image with a defined stride to crop out subsets of the image with different centers, sizes, and aspect ratios.

Region Proposal Network (RPN)

- Goal: Generate and refine region proposals
- Anchor boxes are compared with the reference bounding boxes
- Boxes that have an IoU larger than a defined threshold are defined as having an object within them
- Predicts whether a bounding box has an object within it
- Refines the anchor boxes by predicting bounding box offsets

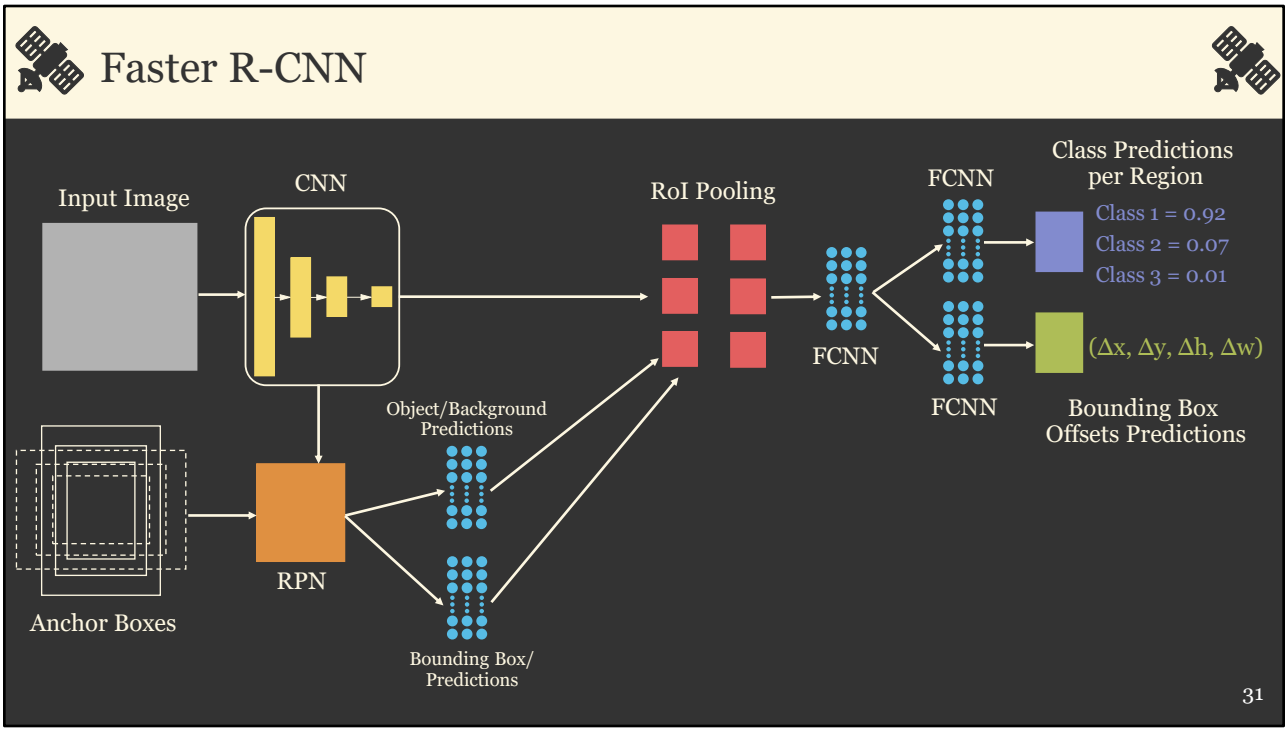
30

The large set of anchor boxes are passed to the region proposal network (RPN).

The anchor boxes are compared to the reference bounding boxes, and boxes that have an intersection-over-union (IoU) larger than a defined threshold are defined as having an object within them.

The RPN then further refines the boxes by predicting whether a box has an object within it and augmenting the anchor box by predicting bounding box offsets. Both of these tasks are accomplished using fully connected layers.

The result is a set of region proposals.



This slide conceptualizes the faster R-CNN architecture.

First, input data are passed through a CNN architecture. This architecture is often pre-trained on a large datasets. It can also be refined using the current dataset and task.

These feature maps are passed to the region proposal network (RPN), which was described in the prior slide. Regions are cropped out using a large set of anchor boxes, and the RPN is used to generate region proposals.

The region proposals generated from the RPN and the feature maps produced by the CNN are then passed to an RoI pooling module that standardizes the shapes and sizes/aspect ratios of the region proposals.

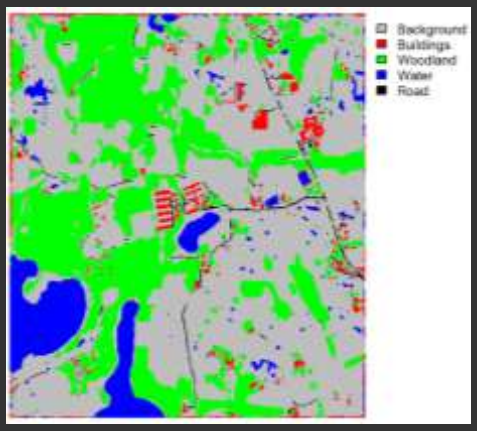
The outputs of the RoI pooling module are flattened and passed to a series of fully connected layers.

The output features that are generated by the last fully connected layer are passed to two separate fully connection architectures. One is used for classification and the other is used for further bounding box refinement.

Semantic Segmentation

32

Problem Description



- Label each pixel in an image to a specific class
- Pixel-level labeling

33

Semantic segmentation is another name for pixel-level prediction or labeling. In this section, I discuss two deep learning, CNN-based architectures that are widely used for semantic segmentation: UNet and DeepLabv3+.

UNet

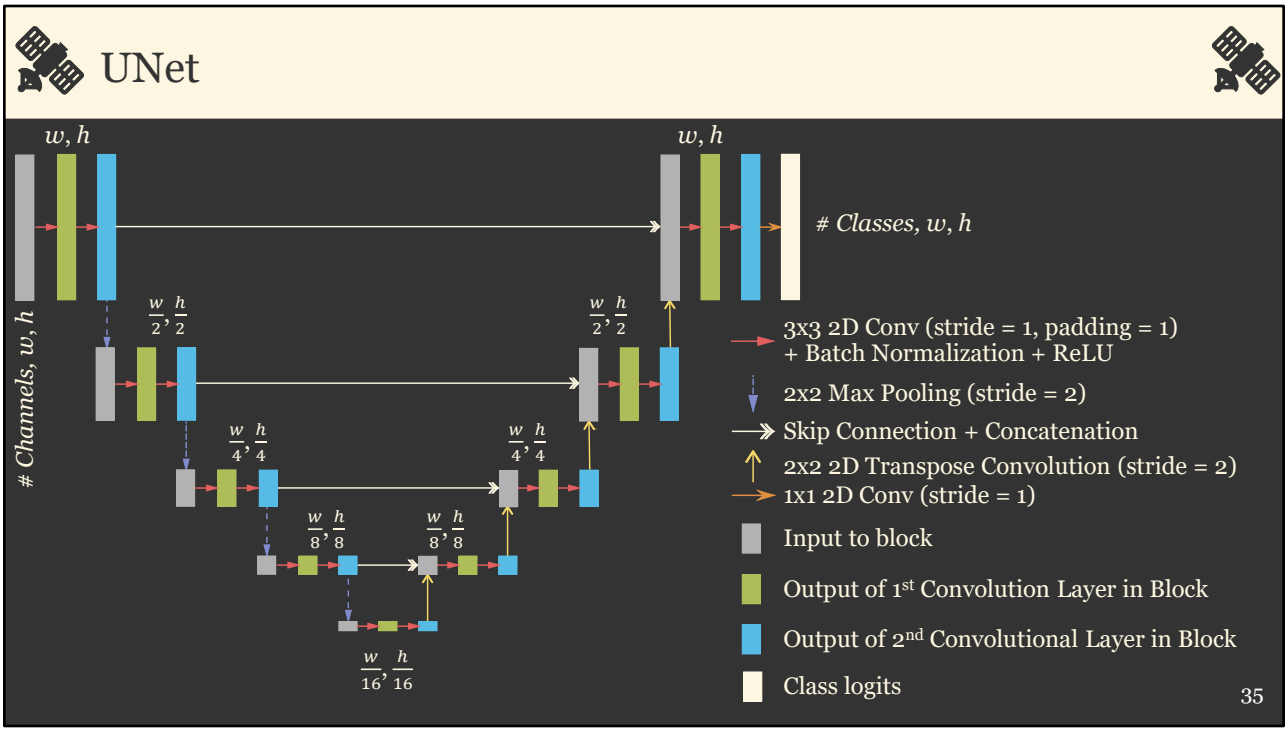

U-Net: Convolutional Networks for Biomedical Image Segmentation

1 Introduction

Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

https://arxiv.org/abs/1505.04597

The UNet method was first introduced in 2015 and builds upon CNNs to support pixel-level labeling or semantic segmentation. This slide provides a reference to the paper that introduced UNet. Although originally developed for medical image segmentation, it has been shown to have many applications.

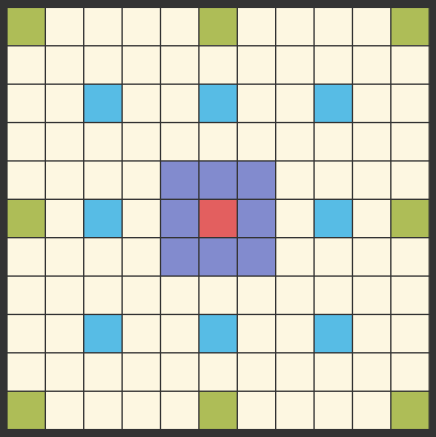


UNet consists of an encoder that generates spatial abstractions of the data at progressively lower resolutions. The bottleneck marks the point at which the feature maps reach their lowest resolution while capturing the greatest concentration of semantic information. The decoder then reconstructs spatial resolution through successive upsampling steps, using convolution operations to refine the learned representations. A classification head generates predictions from the final feature maps. A defining characteristic of UNet is its skip connections, which bridge corresponding encoder and decoder stages to preserve and reuse spatial information that would otherwise be lost during downsampling.

This architecture is designed for pixel-level prediction tasks and can be configured for multiclass classification, binary classification, probabilistic prediction, or regression.

Convolution with Dilation

- Atrous convolution
- Add zeros in kernel to expand kernel size
- Increases the size of the receptive field
- Learn spatial patterns at varying scales
- Dilation Rate = 1
- Dilation Rate = 3
- Dilation Rate = 5



36

In order to increase the effective field of view or receptive field of convolutional operations and in order to learn spatial context information over larger extents, some semantic segmentation architectures make use of dilation or atrous convolution. The idea is to insert zeros into the kernel filters to increase the distance between cells included in the operation. This is generally controlled using a dilation rate.

In the example image, the red cell represents the cell that is being processed, which is included in all the kernel windows. The purple cells are included when using a dilation rate of 1. This is equivalent to a normal 3x3 kernel. The blue cells are included when the dilation rate is 3, and the green cells are included when the dilation rate is 5. In all cases, the number of cells with trainable weights is 9. The difference is in the spacing between the included cells, which again is controlled by the dilation rate. The dilation rate determines the number of inserted zeros.

It is also possible to use dilated convolution with more then 9 trainable weights in the kernel. In such configurations, the dilation rate within the kernel controls how many zeros are inserted between cells that have trainable parameters.

In, summary, the key idea behind atrous convolution is to change the effective

field of view, or receptive field, by incorporating offsets in the moving window, which is controlled by a dilation rate. So, windows can be made up of pixels that are not adjacent, allowing for a larger effective field of view.

DeepLabv3+

1. Use backbone network with dilation to generate feature maps
2. Atrous Spatial Pyramid Pooling to obtain multi-scale spatial context information. Consists of:
   - 1x1 convolution
   - 3x3 convolution with different dilation/atrous rates
   - Image pooling for global context
3. 1x1 convolution and 3x3 convolution to learn additional filters
4. Upsampling to resize feature maps
5. Concatenation to stack feature maps


Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation

Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).

https://towardsdatascience.com/review-deeplabv3-atrous-convolution-semantic-segmentation-6d818bfd1d74

https://developers.arcgis.com/python/guide/how-deeplabv3-works/

https://github.com/tensorflow/models/tree/master/research/deeplab

37

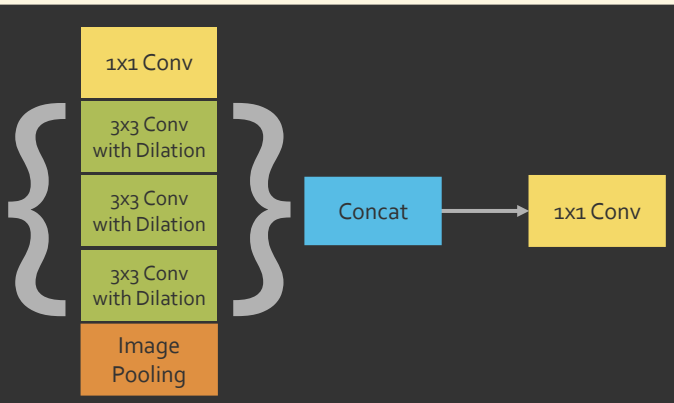
DeepLabv3+ is an example of a CNN-based semantic segmentation architecture that incorporates dilated or atrous convolution.

First, an encoder or backbone is used to extract feature maps at vary spatial resolutions. Some of the kernels in this architecture use atrous convolution. There is also a second atrous spatial pyramid pooling (ASPP) component that is designed to further capture spatial context information at multiple scales. We will discuss this component on the next slide. 1x1 and 3x3 convolution are used to learn additional filters, and upsampling and concatenation are used to standardize the sizes of feature maps in the spatial dimensions and merge them into a stack of feature maps, respectively.

Please have a look at the links provided for additional details. The paper that introduced the method is also referenced here.

Atrous Spatial Pyramid Pooling

- ASPP
- Allows for multi-scale context information to be combined/learned without significantly increasing the architecture size and number of parameters



Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).

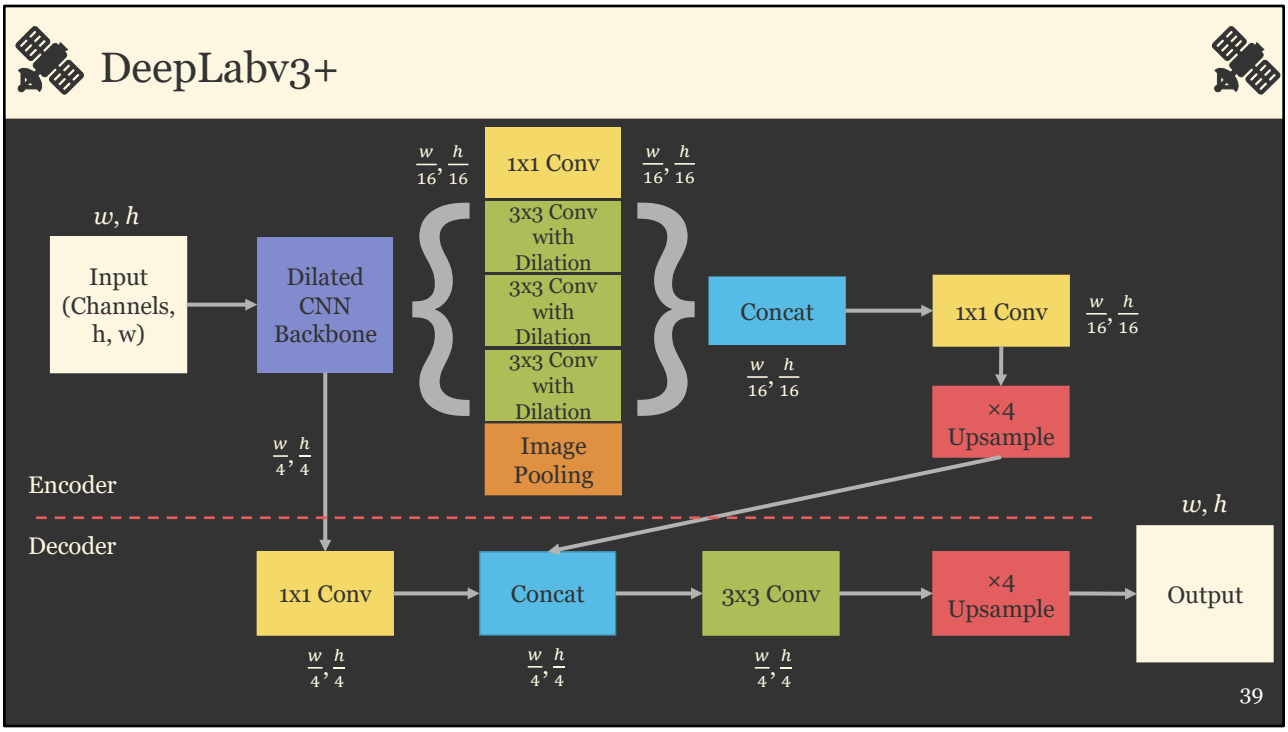
38

The atrous spatial pyramid pooling (ASPP) module allows for the aggregation of spatial context information across scales and also the incorporation of global information from the entire input extent.

Feature maps from prior layers are passed through a series of operations including 1x1 2D convolution, multiple 3x3 2D convolutions with different dilation rates, and image pooling to incorporate global context information.

The learned feature maps are then concatenated and passed to a 1x1 2D convolution layer to further abstract the data and augment the number of feature maps.

Again, the goal here is to learn spatial context information at different scales and increase the size of the receptive field.



This slide conceptualizes the components of DeepLabv3+.

The input data are passed through a backbone encoder network. The feature maps learned in the backbone are then passed to the ASPP module. The learned feature maps from the ASPP module are then concatenated and passed through a 1x1 2D convolution layer to reduce the number of feature maps. Lastly, upsampling by a factor of 4 is applied so that the feature maps have the same size in the spatial dimensions as the feature maps to which they will be concatenated in the decoder component of the model.

In the decoder component of the model, the feature maps from the encoder backbone are passed through a 1x1 2D convolution layer. The feature maps from the ASPP module that have been upsampled are then merged with these feature maps via concatenation. The layers then pass through a 3x3 2D convolutional layer and are then upsampled by a factor of 4 to obtain the pixel-level prediction at the spatial resolution of the input data.

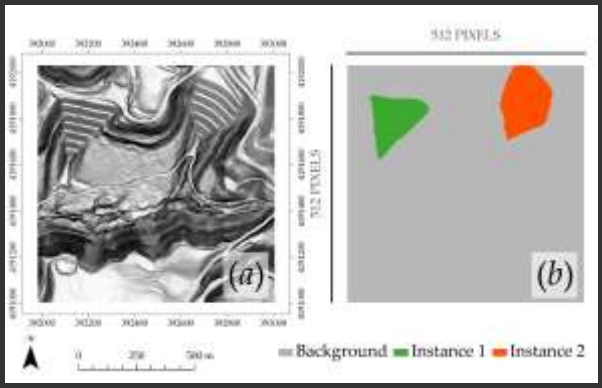
This architecture has the same purpose as the UNet architecture: to make predictions at the pixel-level. However, the architecture is different. As described here, one of the key differences is the use of atrous convolution and the ASPP module.

Instance Segmentation

40

Problem Description

- Differentiate Each Instance of each feature in an image with a bounding box and/or a pixel-level mask



41

Instance segmentation entails differentiating each unique instance of the class(es) of interest as bounding boxes and pixel-level masks or predictions. It combines object detection and instance segmentation.

Mask R-CNN

1. Convolutional Neural Network = learn feature maps
2. Region Proposal Network (RPN) = generate candidate regions within the image
3. ROI Align = improve alignment between RoI pooling layers and RoIs
4. Fully Connected Layers = for bounding box regression and class prediction
5. Fully Convolutional Neural Network = for pixel-level semantic segmentation within the RoIs

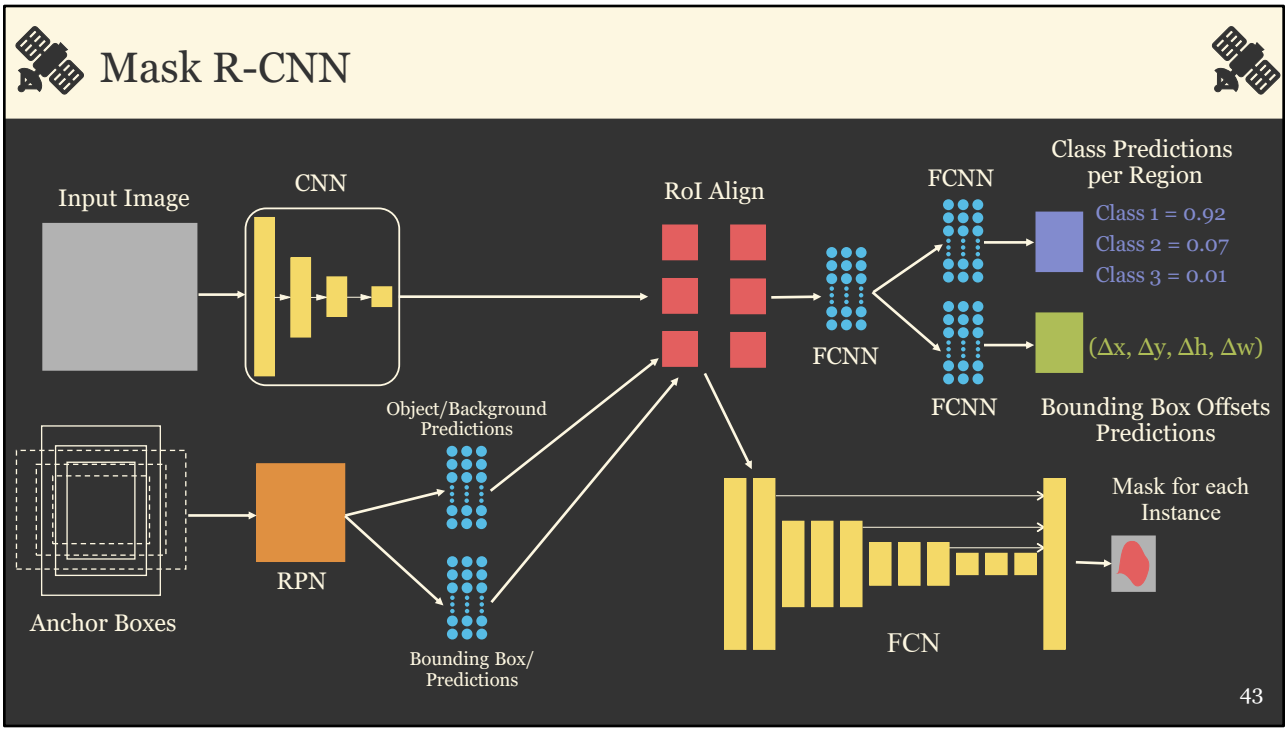
He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

42

Mask R-CNN extends faster R-CNN for instance segmentation.

The components of mask R-CNN are the same as those for faster R-CNN with the addition of the fully convolutional branch that performs the pixel-level mask predictions within the RoIs. The RoI pooling module is also replaced with an RoI align module.



Comparing this conceptualization of mask R-CNN provided on this slide with the conceptualization of faster R-CNN provided on a prior slide, you will notice that many of the same modules are used. One difference is that the RoI pooling module is replaced with a more refined RoI align module, which is necessary to obtain well aligned, pixel-level masks.

A branch is added to perform pixel-level segmentation using a fully convolutional neural network architecture.

All other components of the architecture, including the region proposal network (RPN), CNN backbone, object classification, and bounding box refinement, are identical.

In short, mask R-CNN is just an extensions of faster R-CNN that adds a branch for pixel-level classification within each bounding box. The RoI pooling module is also replaced with a refined RoI align module.

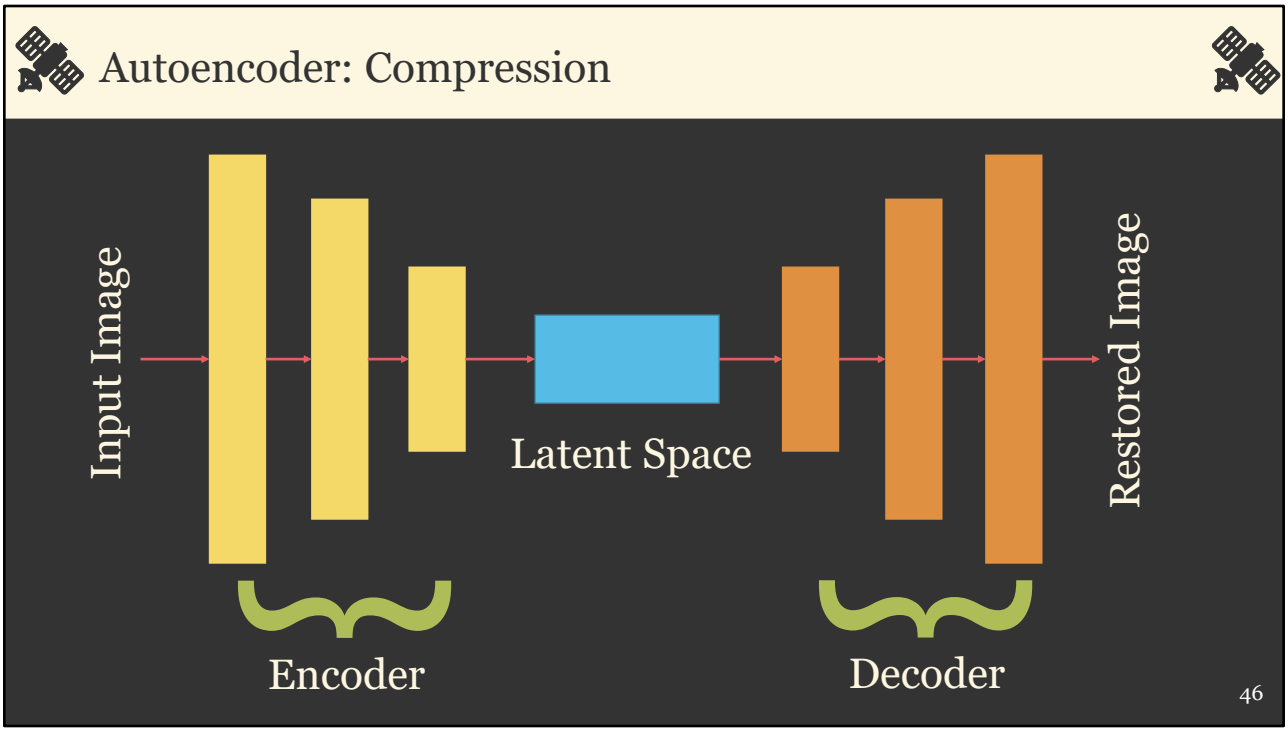
Other Use Cases

Use Cases

1. Compression
2. Pixel-level regression
3. Pixel-level change detection
4. Anomaly detection
5. Image translation/super-resolution
6. Anomaly detection
7. Gap filling
8. Cloud removal
9. Synthetic data generation

45

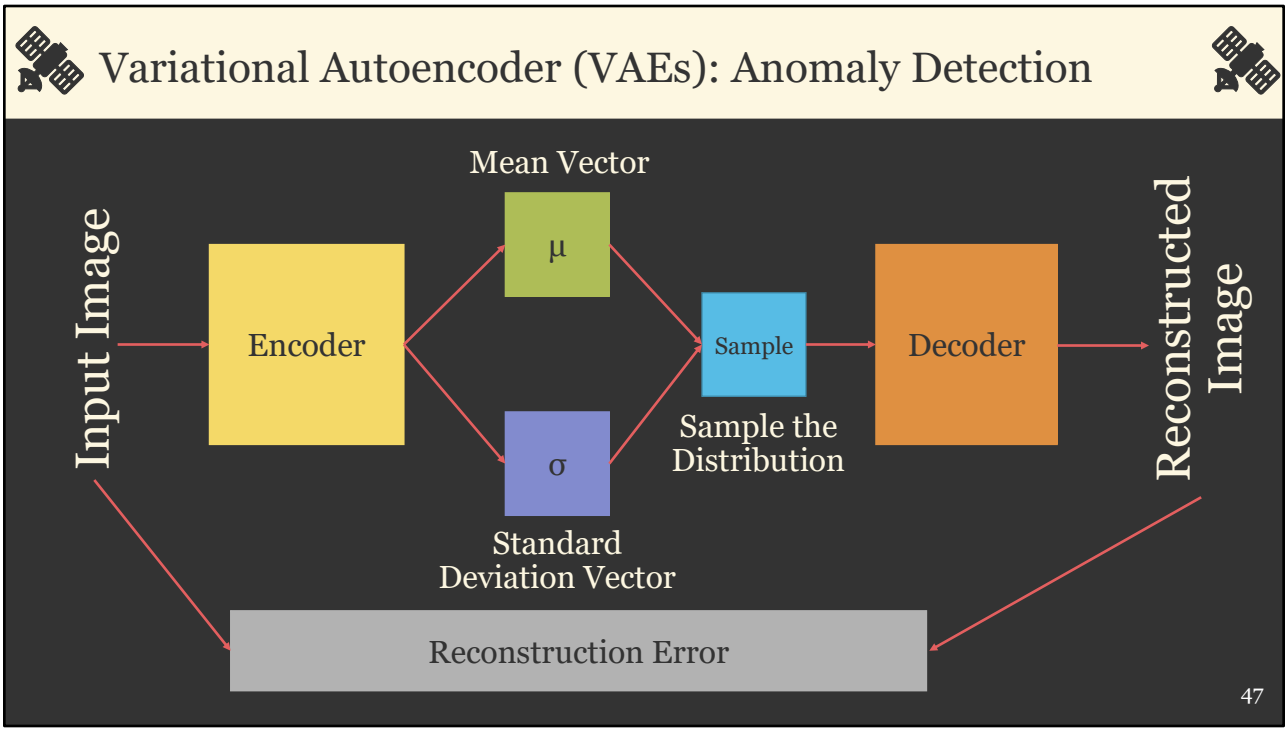
This slide lists other use cases of deep learning and GeoAI in the geospatial sciences and remote sensing. In the following slides, we provide a few examples.



The goal of an autoencoder is to be able to accept input data, transform it to a lower dimensional space, then rebuild or restore the data with limited errors. The lower dimensional space is generally termed the latent space. The encoder component converts the input data to the lower dimensional space while the decoder attempts to restore the data. The encoder and decoder components can be created using CNN or fully connected architectures. For example, A UNet-like architecture could be used. However, instead of attempting to predict pixel-level labels, the goal would be to reconstruct the original image. Also, skip connections would not be included since the goal of the decoder is to be able to reconstruct the input using only the latent space (i.e., the output from the bottleneck layer).

One common and very practical use of autoencoders is to compress files. The latent space would take up less space or memory on a system. It would also be easier to transfer these data over a network.

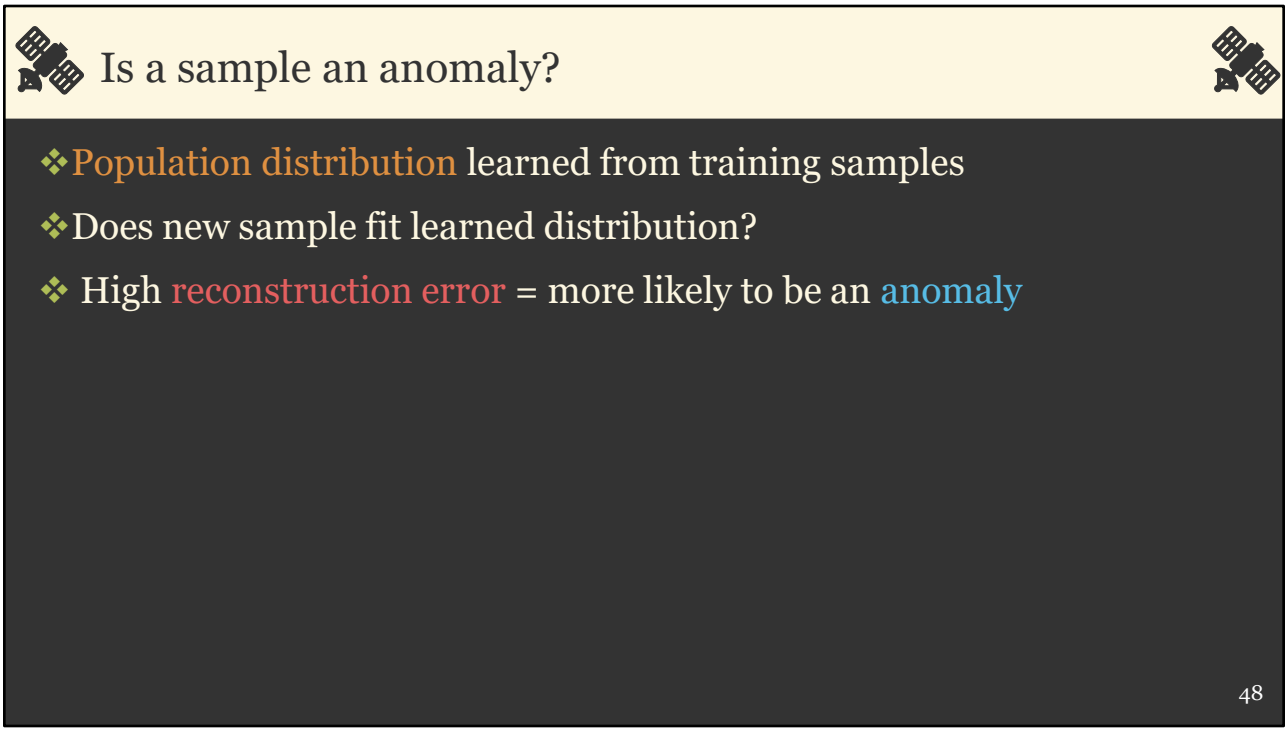
An autoencoder could be used to convert an image or audio file to a latent space for storage and/or transfer then subsequently restore the data with limited loss of data.



One augmentation of autoencoders that have been shown to be especially useful are variational autoencoders, or VAEs. VAEs are different from autoencoders because they do not generate a single latent space. Instead, the goal is to predict mean and standard deviation vectors that represent the latent space as a distribution.

A sample can be drawn from the learned distribution and be fed through the decoder to generate a reconstruction.

These architectures can be used to create synthetic data.



One use of a VAE is to determine whether a new sample is an anomaly, or different from the samples that were used to train it. If the model is not able to reconstruct an input accurately, this could indicate that it is outside of the learned distribution.

For example, if an architecture is trained using only images of forests without any animals in the scene, it would be expected that reconstructing an image that does have an animal in a scene could not be done accurately. As a result, this model could be used to find images that contain animals.

One complexity is that the user must determine what level of reconstruction error should be flagged as indicating an anomaly. This generally is done by looking at the range of reconstruction errors for anomalous and non-anomalous samples.

Foundation Models

What is a Foundation Model?

1. General-purpose model that can be fine-tuned for specific tasks
2. Pre-trained on large and diverse datasets
3. Learn representations that can support a wide variety of tasks
4. Adaptable to downstream tasks
5. Generally large with a large number of trainable parameters
6. Trained without labels using self-supervised or weakly-supervised methods
7. Emergent capabilities

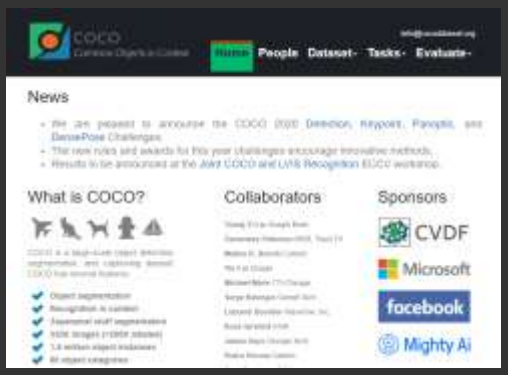
50

A foundation model is a large-scale AI system trained on vast amounts of diverse data that serves as a general-purpose base which can be adapted to a wide range of tasks. Rather than being built for a single specific purpose, foundation models learn broad patterns and knowledge during training, allowing them to be fine-tuned or prompted to perform tasks like answering questions, summarizing documents, generating images, writing code, and much more. The term "foundation" reflects their role as a versatile starting point upon which more specialized applications can be built, making them a cornerstone of modern AI development.

Transfer Learning

- Start with weights learned from a larger training dataset
- Refine weights using new data

http://www.image-net.org/



https://cocodataset.org/#home

51

Foundation models build upon transfer learning.

The idea here is that images contain some common, basic features and spatial patterns, so starting from these pre-trained parameters will generally be better than starting from random parameters even if your classification problem is different from the original problem. This can even be true if the defined classes are different.

Using a very large set of training data can allow a foundation model to develop emergent capabilities and be a good starting point for a variety of tasks.

Learning Methods

Method	Required Pre-Defined Labels?	Description	Examples
Supervised	Yes	Learn from only labeled data	Methods discussed in this class (SVM, RF, *k*NN, ANNs)
Semi-Supervised	Yes	Learn from a small set of labeled data and a larger set of unlabeled data	Pseudo-labeling, self-training, consistency regularization, mean teacher, noisy student
Self-Supervised	No	Create "labels" from the input data	Autoencoders, Denoising autoencoders, Masked autoencoders, jigsaw puzzle prediction
Weakly-Supervised	Yes, but assumed imperfect	Learn from noisy, incomplete, coarse, or approximate labels	image-level labels for segmentation, bounding box instead of mask, point supervision, noisy label learning
Unsupervised	No	Clustering methods	*k*-Means Clustering, ISODATA

52

Machine learning and deep learning encompass several training paradigms that differ in how much labeled data they rely on. Supervised learning is the most straightforward: every training example is paired with a correct label, and the model learns to map inputs to outputs directly from that human-annotated data. Unsupervised learning sits at the opposite end of the spectrum, using no labels at all and instead discovering hidden structure, such as clusters or patterns, purely from raw data.

Semi-supervised learning bridges these two extremes by combining a small amount of labeled data with a much larger pool of unlabeled data, allowing models to achieve strong performance without the cost of labeling everything.

Weakly-supervised learning takes a different angle, training on labels that are noisy, incomplete, or derived from heuristics rather than careful human annotation, accepting some inaccuracy in the supervision signal in exchange for scalability.

Self-supervised learning, perhaps the most influential of these paradigms in recent years, is a special form of unsupervised learning where the model generates its own supervisory signal from the raw data itself (for example, predicting a masked word in a sentence or the next frame in a video, which can allow it to learn rich representations without any human labeling at all).

It is important to note here that foundation models often use training techniques that allow for the incorporation of unlabeled data.

Segment Anything

- Computer vision foundation model
- Generated by Meta
- For general-purpose image segmentation
- Currently, versions 1, 2, and 3 available

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y. and Dollár, P., 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4015-4026).

53

We now provide a few examples of foundation models that have been found to be useful in remote sensing and geospatial science.

Meta's Segment Anything Model (SAM) series, developed by Meta AI, represents a progression of increasingly capable segmentation foundation models, with each generation expanding what kinds of inputs the model can understand and what kinds of data it can operate on.

SAM 1 (released April 2023) established the foundation for promptable, zero-shot image segmentation. Trained on the large-scale SA-1B dataset, SAM 1 supports spatial prompts including points, bounding boxes, and masks, enabling strong zero-shot interactive segmentation without task-specific fine-tuning. Its architecture consists of an image encoder, a prompt encoder, and a mask decoder, and it can automatically segment all objects in an image or isolate specific regions based on a user-provided prompt. Crucially, SAM 1 operates only on static images and has no understanding of natural language or temporal context.

SAM 2 (released mid-2024) adds support for video. SAM 2 introduced a unified image and video segmentation capability, with a streaming memory module that tracks objects across frames. This temporal key-value memory supports long-sequence video reasoning and maintains object identity across

frames, combining ViT-based visual tokens with a memory retrieval pathway to propagate masks over time.

SAM 3 (released November 2025) adds Promptable Concept Segmentation (PCS), where a short text phrase or example image tells the model to segment every instance of a concept in an image or video.

Segment Anything

Version	Release Year	Main idea	Inputs/prompts	Main output	Biggest advance
SAM 1	April 2023	Promptable segmentation for images	Points, boxes, masks	Masks for prompted regions or objects	General-purpose, class-agnostic image segmentation
SAM 2	July 2024	Promptable segmentation for images and video	Points, boxes, masks across frames	Masks that can persist through video	Adds temporal memory and video object segmentation
SAM 3	November 2025	Promptable concept segmentation for images and video	Text, exemplars, points, boxes, masks	Detects, segments, and tracks all matching concept instances	Adds open-vocabulary text/exemplar-based segmentation

Assumption	Description
Image Encoder	Convert input images into dense image embeddings
Prompt Encoder	Convert text, points, boxes, or mask into prompts
Mask Decoder	Combines images and prompt information to generate segmentation masks

54

The tables on this slide summarize the different versions of SAM and the different encoder components.

Fine-Tuning SAM

- Zero-Shot = no new labels (just provide new data and obtain the segmentation)
- Few-Shot = Given small set of labels

Method	What is trained?
LoRA	Low-rank updates inside attention/projection layers
Adapters	Small trainable layers inserted into the encoder or decoder
Prompt Tuning	Learned prompt embeddings or prompt generator
Decoder-Only Fine-Tuning	Mask Decoder
Input Adapter	Small network that maps non-standard input bands into SAM-compatible inputs

Low-Rank Adaption (LoRA) = Keep the original model weights frozen, then add small trainable low-rank matrices that learn task-specific adjustments

55

Zero-shot learning and few-shot learning both address the same fundamental challenge: how can a model perform a task it was never explicitly trained on? They differ, however, in how much task-specific information is available at inference time.

In zero-shot learning, the model receives no examples of the target task whatsoever; it must rely entirely on knowledge and representations built up during pretraining, along with a natural language description or instruction explaining what is expected.

Few-shot learning provides the model with a small number of input-output examples before asking it to perform the task on a new input. These examples are included directly in the prompt and serve as an implicit guide to the format, style, and logic of the expected response.

The practical tradeoff between the two is one of convenience versus performance: zero-shot is simpler and requires no curated examples, but few-shot typically yields better results by anchoring the model's behavior more precisely to what is actually needed. Both approaches stand in contrast to traditional fine-tuning, where a model's weights are updated using a larger labeled dataset. Zero- and few-shot learning work entirely within the inference step, with no modification to the model itself.

The table on this slide lists some different methods for fine-tuning SAM.

SAM for Geospatial Data

- In Python SamGeo — https://samgeo.gishub.org/
- InGIS (qgis-sam-geo-plugin) — https://github.com/opengeos/qgis-samgeo-plugin
- In ArcGIS — https://www.arcgis.com/home/item.html?id=9b67b441f29f4ce6810979f5f0667ebe

56

SAM has been implemented for application to geospatial data using the tools referenced on this slide.

AlphaEarth

- Developed by Goole DeepMind
- General-purpose Earth observation embeddings
- 64 embeddings
- 10 m spatial resolution
- Derived from
- Global extent
- Annual update (2017 to 2014)
- Input data = Sentinel-2 L1C, Sentinel-1 GRD, Landsat 8/9 Tier 1 ToA, and others (GEDI, GRACE, GLO-30 DEM, NLCD)

Brown, C.F., Kazmierski, M.R., Pasquarella, V.J., Rucklidge, W.J., Samsikova, M., Zhang, C., Shelhamer, E., Lahera, E., Wiles, O., Ilyushchenko, S. and Gorelick, N., 2025. Alphaearth foundations: An embedding field model for accurate and efficient global mapping from sparse label data. *arXiv preprint arXiv:2507.22291*.

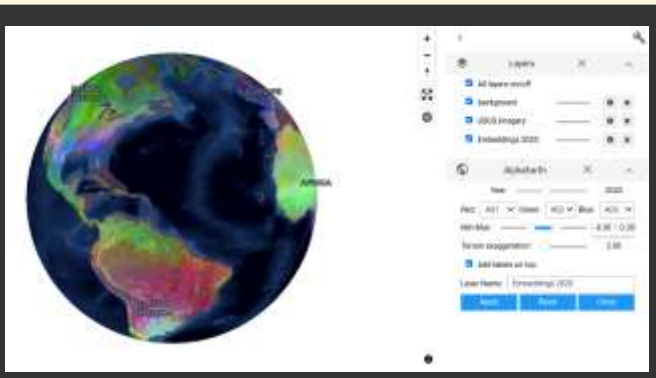
57

AlphaEarth Foundations is a geospatial foundation model developed by Google DeepMind, announced in mid-2025. It is designed to function like a virtual satellite, characterizing the planet's terrestrial land and coastal waters by integrating large amounts of Earth observation data into a unified digital representation, or embedding, that computer systems can process.

The model's core approach centers on creating compact, information-rich summaries of the Earth's surface. It divides the planet into 10-by-10 m cells, each containing a compact summary of what's there, updated annually, automatically fusing and standardizing data to create a consistent view of the planet. The data sources ingested include optical and thermal imagery from Sentinel-2 and Landsat satellites, radar data that can see through clouds, 3D measurements of surface properties, global elevation models, climate information, gravity fields, and descriptive text. Each location is represented as a 64-dimensional embedding that distills complex optical, radar, and climate inputs into an analysis-ready format.

AlphaEarth Via Google Earth Engine

- Made available via Google Earth Engine



https://developers.google.com/earth-engine/tutorials/community/satellite-embedding-01-introduction

https://developers.google.com/earth-engine/datasets/catalog/GOOGLE_SATELLITE_EMBEDDING_V1_ANNUAL

https://opengeoai.org/examples/AlphaEarth/

58

The AlphaEarth embedding are available via Google Earth Engine.

This is the end of this lecture module.

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

59

Thanks! Hope you found this useful.