Geospatial Deep Learning

Object Detection and Instance Segmentation

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

WV VIEW

AmericaView
Empowering Earth Observation Education
AmericaView.org - Est. 2003

This module will focus on object detection and instance segmentation using convolutional neural network (CNN)-based deep learning. We will not cover these topics in detail, as was the goal with the scene labeling and semantic segmentation materials. Instead, we will just provide an introduction to some common object detection and semantic segmentation architectures and discuss input data requirements.

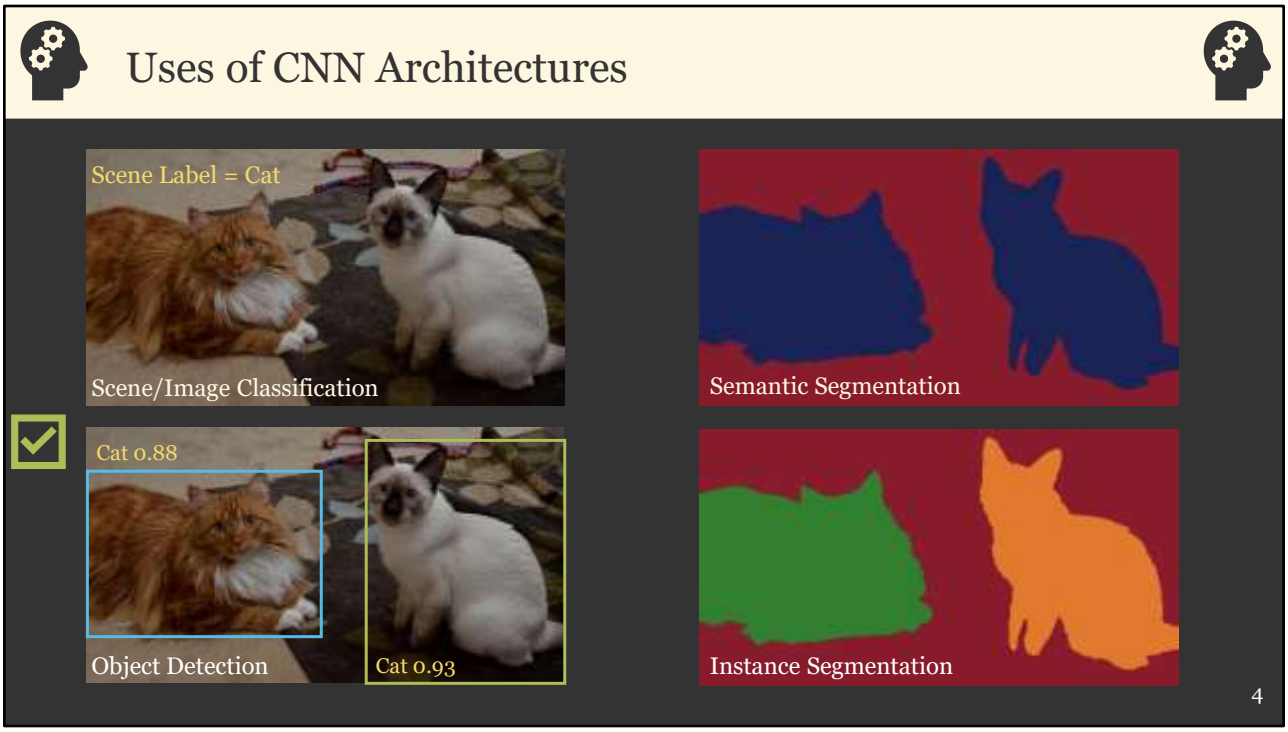
Module Topics

1. Object detection vs. instance segmentation
2. IoU and mAP
3. Non-maximum suppression
4. Selective search
5. R-CNN, fast R-CNN, and faster R-CNN
6. Single-Shot Multibox Detector
7. YOLO-family
8. UNet-ID
9. Mask R-CNN
10. Data requirements

2

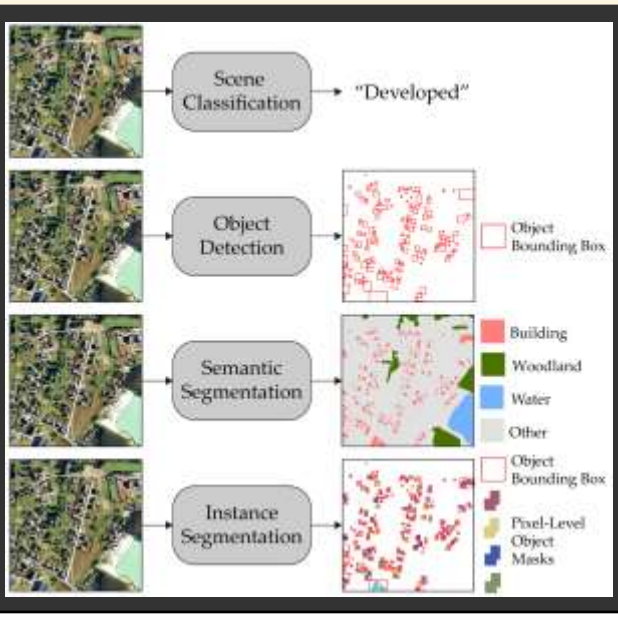
These are the topics covered in this module.

Object Detection



Object detection consists of predicting bounding boxes for objects within an image extent along with associated labels and class probabilities. Object detection is more localized than scene labeling but less localized than semantic segmentation.

Uses of CNN Architectures



5

The object detection problem presented on this slide is the extraction of bounding boxes for all building in an image extent.

Although this example demonstrates an object detection task where only one type of object is predicted (buildings), it is possible to detect multiple types of objects at once. For example, an algorithm could be trained to predict bounding boxes for buildings, swimming pools, ponds, and individual trees.

Object Detection

- Localization
- Predict bounding box for each instance of object of interest
- Predict label for each detected feature

6

Again, the output of an object detection model is predicted bounding boxes for each detected object of the class or classes of interest within the image extent along with class labels and predicted probabilities or logits. It is possible for bounding boxes to overlap.

Object Detection

Architecture	Release Year	Reference
Region-based CNN (R-CNN)	2014	(Girshick et al., 2014)
Spatial Pyramid Pooling Net (SPPNet)	2014	(He et al., 2015a)
Fast R-CNN	2015	(Girshick, 2015)
Faster R-CNN	2015	(Sun, 2015)
Single Shot Multibox Detector (SSD)	2016	(Liu et al., 2016)
YOLO-v1	2016	(Redmon et al., 2016)
Feature Pyramid Network (FPN)	2016	(Dollar et al., 2017)
YOLO-v2	2017	(Redmon and Farhadi, 2017)
Mask R-CNN	2017	(He et al., 2017)
RetinaNet	2017	(Lin et al., 2017)
YOLO-v3	2018	(Redmon and Farhadi, 2018)
Cascade R-CNN	2018	(Cai and Vasconcelos, 2018)
Path Aggregating Network (PANet)	2018	(Liu et al., 2018)
TridentNet	2019	(Li et al., 2019)
EfficientDet	2019	(Tan et al., 2020)
Composite Backbone Network (CBNet)	2019	(Liu et al., 2020)

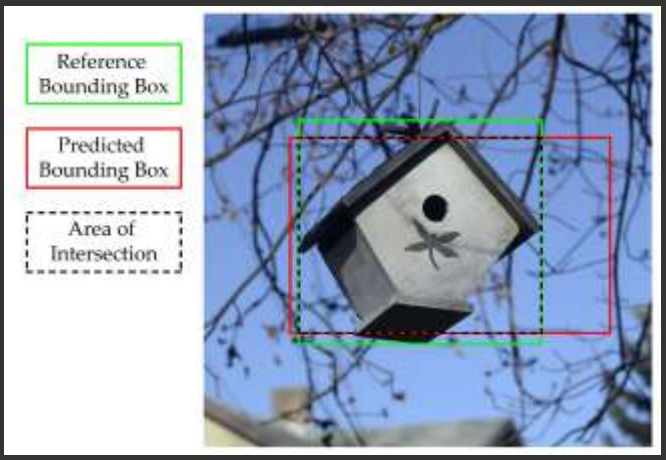
This slide outlines some object detection methods/architectures that have been proposed since 2014, not long after the introduction of AlexNet for scene labeling tasks.

We will focus on region-based convolutional neural networks (R-CNNs) and their derivatives: fast R-CNN and faster R-CNN. We will more generally discuss single shot multibox detectors (SSDs) and the YOLO family of architectures.

Intersection over Union (IoU)

$$\frac{Area\ of\ Intersection}{Area\ of\ Union}$$

- Use bounding box or pixels
- Compare predicted extent or bounding box to reference extent or bounding box
- Used for object detection
- Used in computer vision
- Average for multiple class = mean IoU (mIoU)



8

Before we begin, I wanted to provide a brief review of a few assessment metrics that are commonly used for object detection and/or instance segmentation tasks.

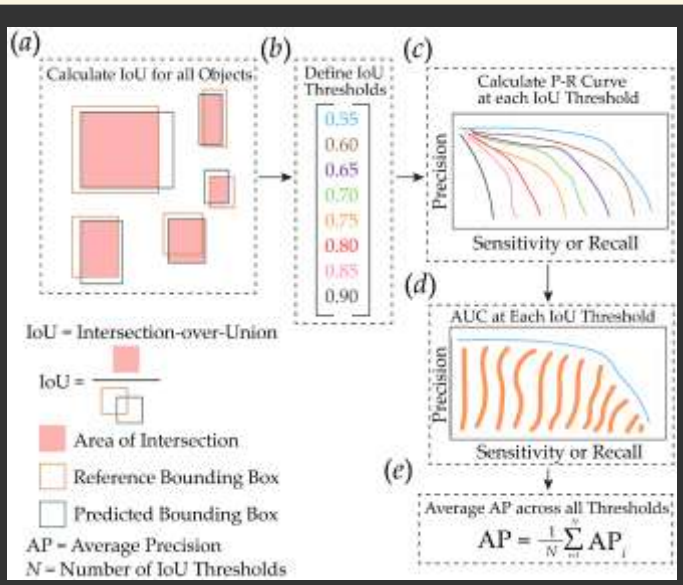
Intersection-over-union (IoU), also known as the Jaccard Index, represents the proportion or area that is shared of the combined area of the reference bounding box and predicted bounding box, or the area of intersection divided by the area of union of the two objects. A larger proportion of shared area or overlap suggests better performance.

In the provided example, the green box represents the correct or reference bounding box for the bird house object. The red rectangle represents a prediction, and the dashed, black line represents the area of overlap.

Mean IoU (mIoU) is the mean of IoU for all mapped classes. It is also possible to calculate a weighted mIOU that takes into account differences in class proportions or abundance.

Mean Average Precision (mAP)

- Define IoU thresholds
- Calculate PR Curves
- AP = area under curve averaged at different thresholds
- mAP = average AP for multiple classes



9

The figure on this slide conceptualizes the average precision (AP) or mean average precision (mAP) metrics. For each individual feature or detected object, correct or incorrect predictions can be defined based on a minimum IoU measure that is deemed acceptable to generate a classification and associated metrics. For example, features correctly classified and with an areal intersection of more than 60% with the associated reference feature could be labelled as TPs while features with less than 60% overlap, or those incorrectly classified, could be labelled as FNs. Precision-recall (PR) curves can then be generated based on multiple IoU thresholds by comparing the predicted probabilities of samples labelled as TP, FN, and FP. The area under each of these PR curves can then be calculated and averaged to obtain a combined average precision (AP). It is also possible to calculate AP for multiple classes and average them to a single AP value, which is commonly termed mean average precision (mAP); however, the distinction between AP and mAP is often confused in the literature.

Traditionally, IoU, mIoU, and AP, have been calculated using bounding boxes; however, it is possible to calculate them using pixel-level reference and predicted masks. In the context of object detection, bounding boxes are used.

Object Detection Losses

- Classification loss for class predictions
- Regression-based loss for bounding box prediction

$$Loss_{\text{bbox}} = \sum_{i \in \text{positive}} \left(\text{Loss}(\widehat{d_{x_i}} - d_{x_i}) + Loss(\widehat{d_{y_i}} - d_{y_i}) + Loss\left(\sqrt{\widehat{d_{w_i}}} - \sqrt{d_{w_i}}\right) + Loss\left(\sqrt{\widehat{d_{h_i}}} - \sqrt{d_{h_i}}\right)\right)$$

10

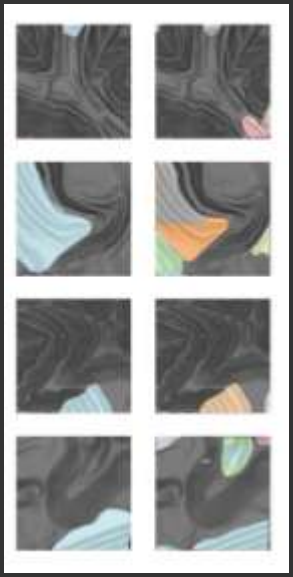
We will not discuss loss metrics for object detection and instance segmentation in detail. However, I did want to mention a few key criteria.

Since this problem involves both predicting a class label and a bounding box, it is necessary to use multiple loss metrics. For the classification component, a standard classification loss metric can be used, such as binary cross entropy (BCE) or cross entropy (CE) loss.

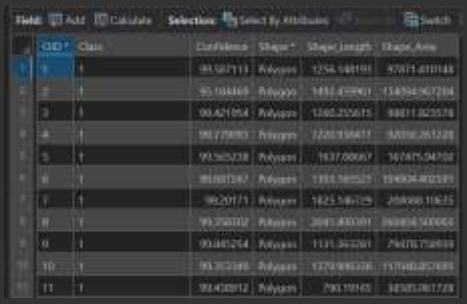
Bounding box prediction is framed as a regression problem. For example, a bounding box can be defined based on a coordinate of one of its corners or its center and an associated width and height. The bounding box loss presented on this slide considers the error or residual for predicting the center coordinates of the bounding box and also the residual or difference between the square root of the reference width and predicted width and reference height and predicted height. A square root is applied to the width and height components to scale the residuals so that they don't take on more weight in the loss than the residuals for the center x and y coordinates.

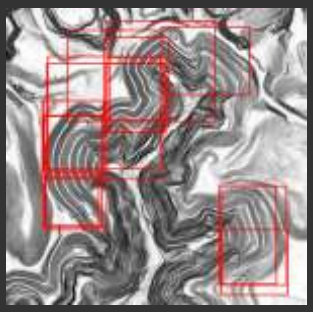
The equation provided on this page is just one example of a bounding box loss. There are different implementations.

Non-Maximum Suppression



- For overlapping predictions, only maintain bounding box or feature mask with highest probability
- Help deal with issues associated with processing data chip-by-chip





11

Another complexity is that multiple bounding boxes can be predicted for each feature. When this occurs, the user must determine which bounding box to select as the final prediction, which can then be compared to the reference bounding box to calculate assessment metrics.

Non-maximum suppression (NMS) is generally implemented separately for each predicted class. First, bounding boxes with a prediction confidence lower than a user-defined threshold for the current class of interest are removed. Next, bounding boxes are sorted based on their predicted logits or estimated class probability for the class label of interest to find the bounding box with the highest confidence. The bounding boxes that overlap with the selected, high confidence bounding box are removed if the area of overlap with the selected bounding box is greater than a defined threshold, as measured using IoU. This process is repeated for each class and until only a single bounding box prediction remains relative to the confidence and overlap thresholds specified.

R-CNN Components

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

1. Selective Search Algorithm = generate region proposals and associated bounding boxes
2. Warp Regions = make all regions a consistent shape, size, and number of rows/columns of cells
3. CNN = Generate feature maps for each region
4. Fully Connected Layers = further augmented flattened feature maps
5. SVM Classification = predict bounding box/object labels
6. Regression = refine bounding box

12

We will begin our discussion of object detection methods with the region-based convolutional neural network (R-CNN) architecture.

This architecture is not solely based on fully connected neural network and convolutional neural network components. It incorporates other routines; as a result, it is not possible to train the algorithm end-to-end with backpropagation.

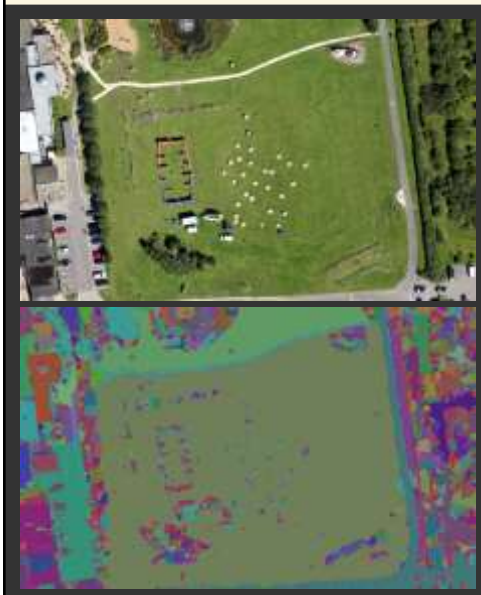
First, the image is broken into regions using a selective search algorithm. This is an algorithm that groups clusters of “similar” contiguous pixels. Different methods are available to implement this clustering. These clusters are used to define region proposals, or areas within the image that may represent a specific object.

The data within these regions is then warped so that they all have a consistent size and shape/aspect ratio. Each separate region is this passed through a convolutional neural network (CNN) to generate feature maps. It is common for this CNN to be pre-trained on a large dataset. In other words, the CNN acts as a feature extractor. If it is not fine-tuned for the current tasks, it is assumed that representations learned for the original tasks using the original dataset are also meaningful or useful for the current task.

Once the feature maps are produced, they are flattened to a 1D vector then passed through a series of fully connected layers. The output values or features from the final fully connected layer are then used as a feature space to perform the classification and bounding box regression tasks.

Classification is generally performed using a support vector machine (SVM) machine learning classifier while regression is performed using linear regression.

Selective Search

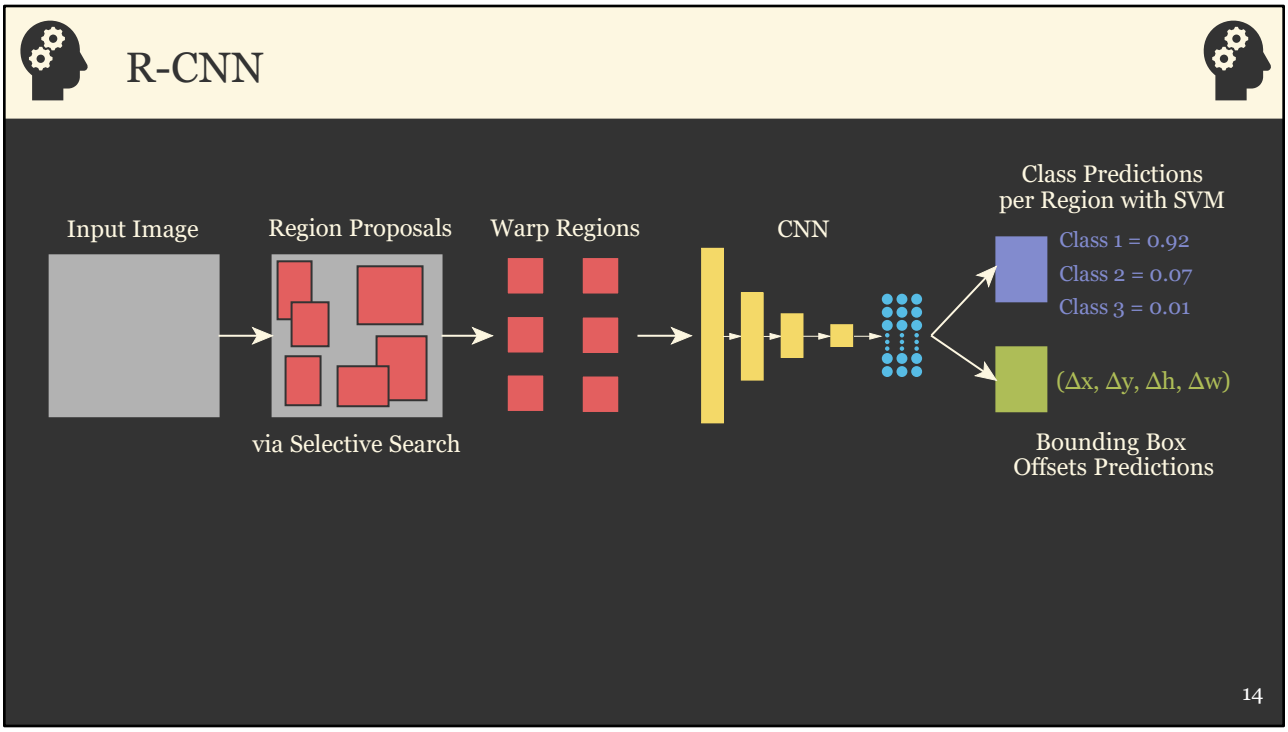


- Cluster pixels based on similarity
- Not based on deep learning

13

This slide conceptualizes the selective search component of the architecture. Again, the goal is to cluster the image pixels into regions or individual objects.

Selective search algorithms are not based on artificial neural networks; as a result, backpropagation cannot be used in this component of the architecture.



This slide diagrams the R-CNN architecture.

First, a selective search algorithm is used to define region proposals. These regions are then warped to a consistent size and shape/aspect ratio.

Feature maps are generated for each warped region proposal separately using a CNN.

The output from the final CNN layer is flattened then passed through a fully connected neural network.

The output features from the fully connected neural network are used as a feature space for SVM-based classification to predict bounding box labels and linear regression to refine the bounding box.

Fast R-CNN Components


Fast R-CNN

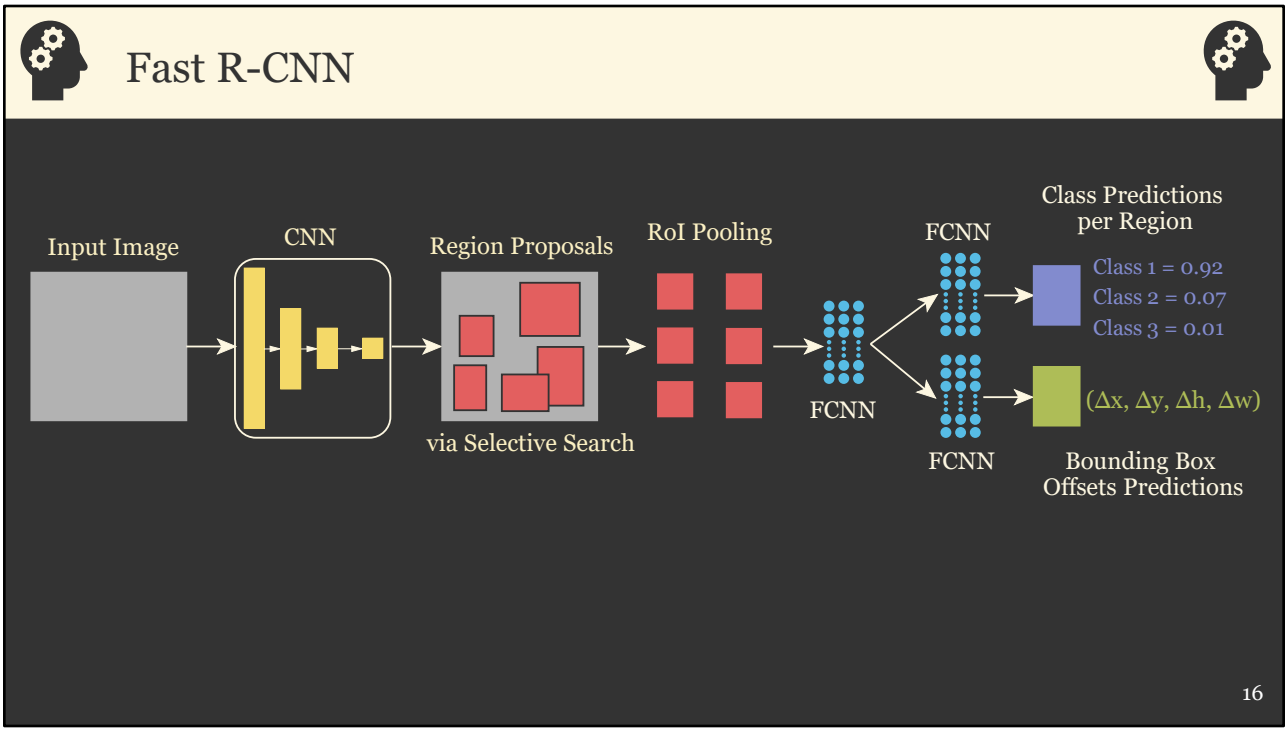
1. Convolutional Neural Network = learn feature maps
2. Selective Search Algorithm = generate region proposals and associated bounding boxes
3. RoI Pooling = make all regions a consistent shape and size
4. Fully Connected Layers = for bounding box regression and class prediction

Girshick, R., 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

15

One of the main issues with R-CNN is that it can be very slow and computationally expensive. This is because each warped region proposal must be passed through the CNN architecture separately. For example, if 100 region proposals are generated, processing a single image requires that 100 regions be passed through the CNN.

Fast R-CNN offers three main improvements over R-CNN. First, instead of creating region proposals from the input data, the input data are first passed through a CNN architecture to generate feature maps. The region proposals are generated using these output feature maps and a selective search algorithm. Second, the region proposals are standardized to a consistent size and shape/aspect ratio using a region of interest (RoI) pooling module. Lastly, the SVM classification and linear regression are replaced with fully convolutional neural networks.



This slide conceptualizes the architecture of fast R-CNN.

First, the input data are passed through a CNN architecture. This CNN can be based on a prior training process or fine tuned for the current task.

The resulting feature maps and a selective search algorithm are then used to generate region proposals.

An RoI pooling module is used to warp the region proposals into a consistent size and shape/aspect ratio.

The warped region proposals are flattened then passed through a fully connected architecture.

The features generated by the first fully connected architecture are then passed to two subsequent and separate fully connected architectures. One is responsible for classification while the other is responsive for the bounding box refinement.

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*.

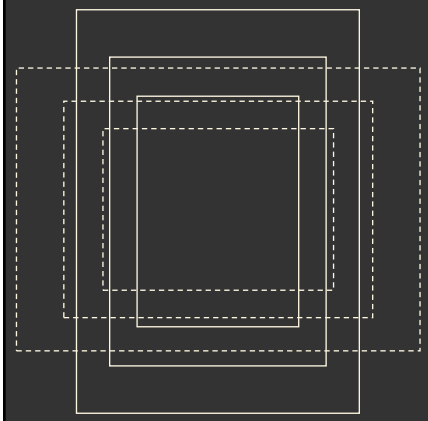
17

One issue with R-CNN and fast R-CNN is that they cannot be trained from beginning to end since the entire architecture is not based on fully connected or convolutional neural networks. R-CNN requires the selective search algorithm and uses SVM machine learning for classification and linear regression for bounding box refinement. Fast R-CNN still incorporates selective search.

Faster R-CNN further speeds up the process while being constructed such that backpropagation can be used to train the entire architecture.

The key refinement is the introduction of the region proposal network (RPN).

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

18

Before talking about the region proposal network, we first need to discuss the concept of anchor boxes.

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

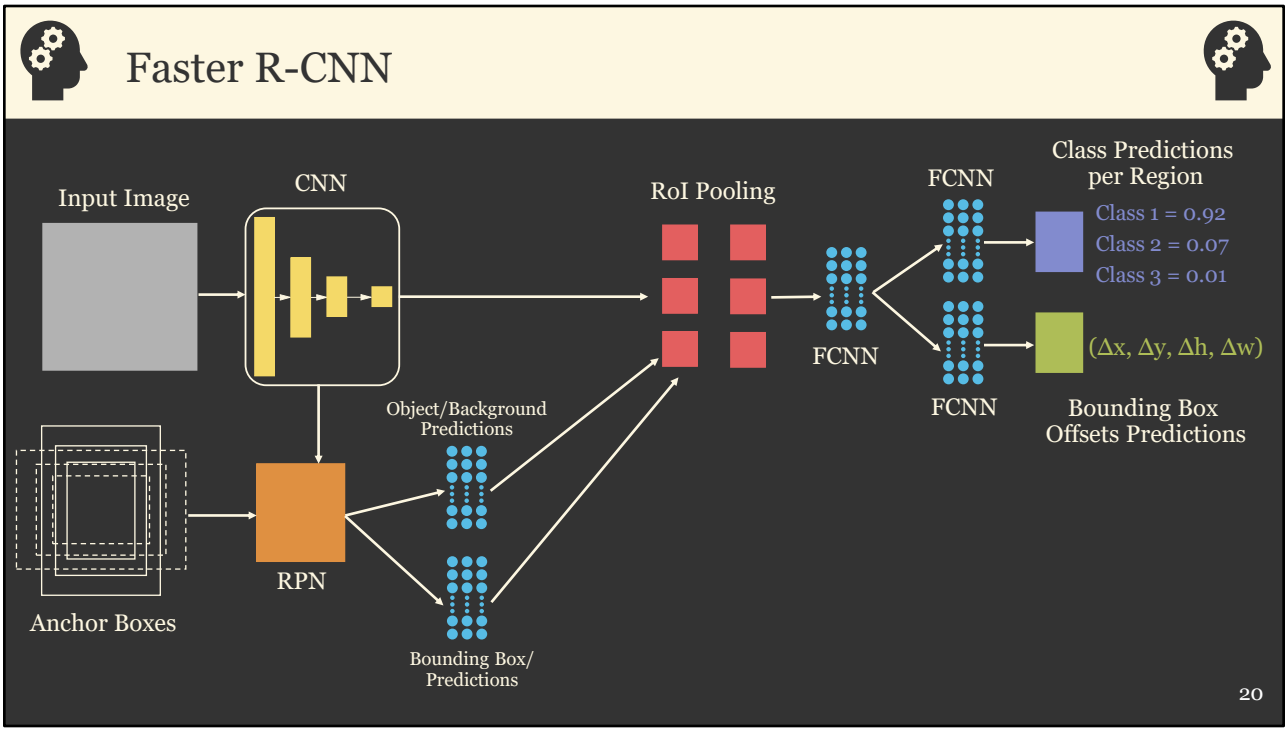
19

The large set of anchor boxes are passed to the RPN.

The anchor boxes are compared to the reference bounding boxes, and boxes that have an 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 RPN. Regions are cropped out using a large set of anchor boxes, and the RPN is used, as described in the prior slide, 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.

Single-Shot Multibox Detector

- For object detection
- Real-time detection
- Faster than region proposal network methods (R-CNN/Faster-RCNN)
- Uses CNNs for feature extraction
- Single-Shot Detector (SSD) head (additional convolutional layers) for generation of bounding boxes and class predictions

21

There are other techniques available for object detection. We will not discuss them in detail here. Instead, I will provide only a brief introduction.

Single-shot multibox detectors (SSDs) offer fast, often real-time detection. They rely on CNNs to generate feature maps followed by a single-shot detector head, which is used to generate the bounding boxes and class predictions.

Faster R-CNN is an example of a two-shot detector since it involves (1) the generation of region proposals and (2) the refinement and classification of the proposals using bounding box regression and classification.

When using an SSD, the single-shot detector head is used to select the best bounding boxes using a single pass.

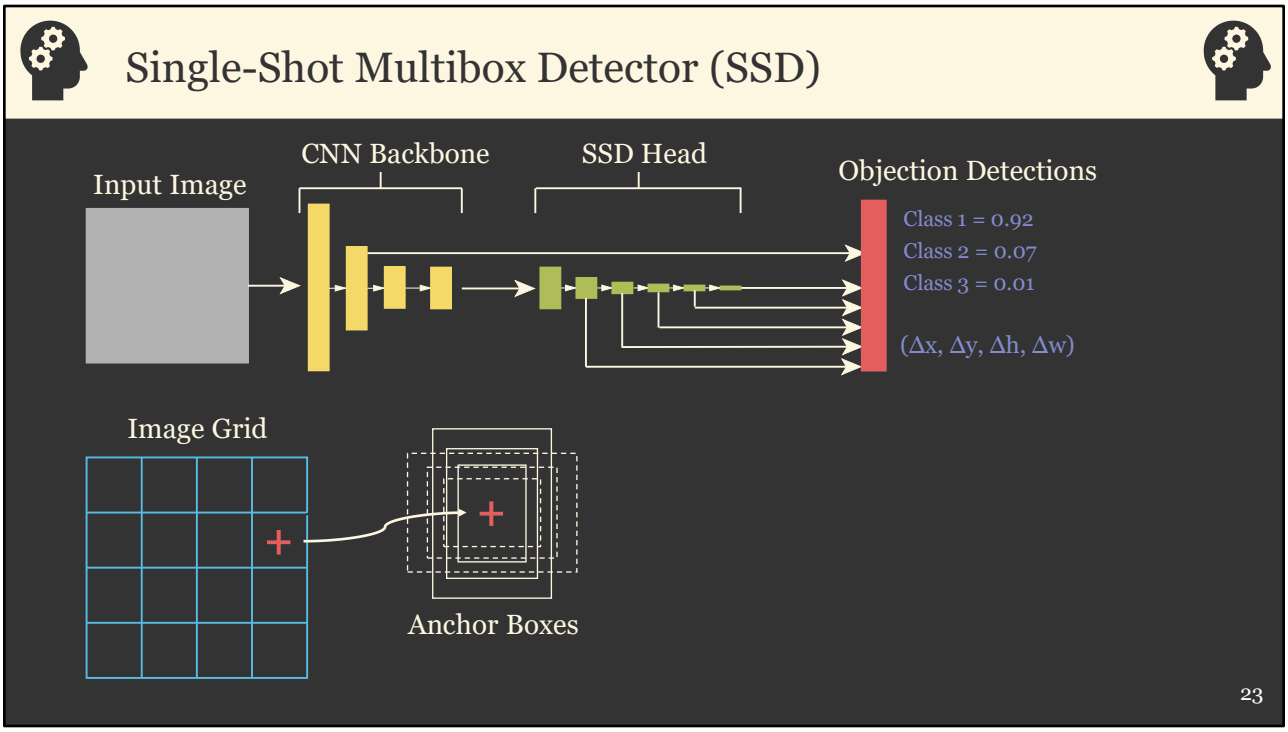
SSDs tend to be faster than region-based methods. However, they also tend to be less accurate.

Components of Single-Shot Multibox Detector

- Image divided into grid cells
- SSD detects objects in each grid cell
- Anchor boxes are used to refine the prediction
- Can define different sizes and aspect ratios for anchor boxes
- Zoom level allows anchor boxes to vary in size
- CNNs allow for detecting features of different sizes
- Uses non-maximum suppression

22

This slide highlights the components of an SSD.



This slide diagrams a basic SSD.

SSDs consist of two general components. A CNN backbone is used to create feature maps. A pre-trained backbone can be used. It can also be further refined using the current data and task.

The second component of the architecture is the SSD head. It consists of convolutional layers, and its purpose is to predict bounding boxes and class predictions associated with each bounding box.

In order to make localized predictions, the entire image extent is broken into grid cells. Anchor boxes with varying sizes and aspect ratios are generated for each grid cell. To train the model, these anchor boxes are matched with training bounding boxes such that the anchor box with the largest IoU is matched with an associated training bounding box. The training process refines the bounding boxes.

As with the R-CNN methods, non-maximum suppression is used to remove multiple

predictions for the same object.

YOLO

- "You Only Look Once"
- Series of architectures with multiple versions
- Extremely fast
- Realtime object detection
- Split input into grid of cells
- Finds best bounding boxes
- Makes class predictions
- Uses non-maximum suppression
- Predicts X Coord, Y Coord, Height, Width, BBox Confidence/Class Probabilities

Input → 7x7 Conv → Max Pooling → 3x3 Conv → 1x1 Conv → 3x3 Conv → 1x1 Conv → 3x3 Conv → Max Pooling → 1x1 Conv → 3x3 Conv → 1x1 Conv → 3x3 Conv → 1x1 Conv → 3x3 Conv → 1x1 Conv → 3x3 Conv → 1x1 Conv → 3x3 Conv → Max Pooling → 1x1 Conv → 3x3 Conv → 1x1 Conv → 3x3 Conv → 3x3 Conv → 3x3 Conv → 3x3 Conv→ 3x3 Conv -> Fully Connected → Fully Connected → (X Coord, Y Coord, Height, Width, BBox Confidence Class Probabilities)

Example = YOLO v1

24

YOLO, or "You Only Look Once", is also an extremely fast object detector. It works similar to SSDs. The main difference is how the bounding box grids are defined and that fully connected or densely connected layers are used to select bounding boxes.

Note that there are different versions or iterations of YOLO that function differently and have evolved over time.

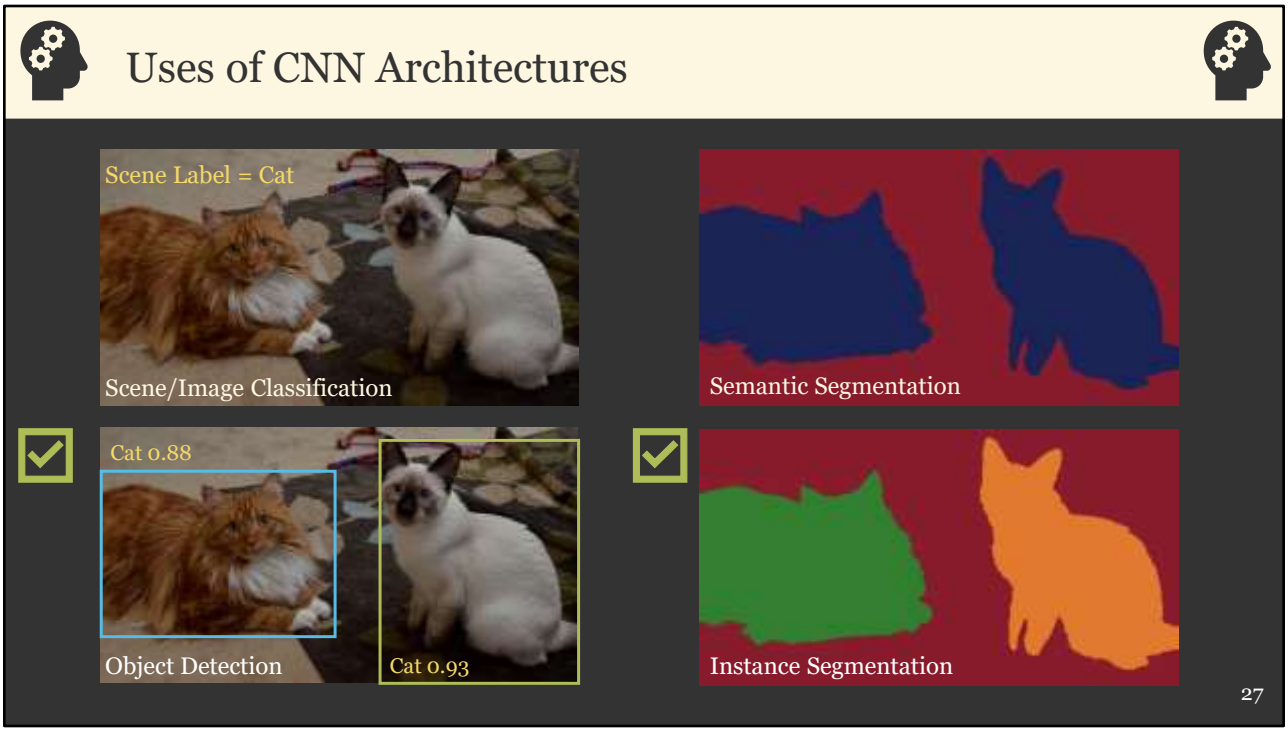
Object Detection Data Requirements

- Images
- Bounding boxes for each object instance
- Can generate bounding boxes from pixel-level masks

25

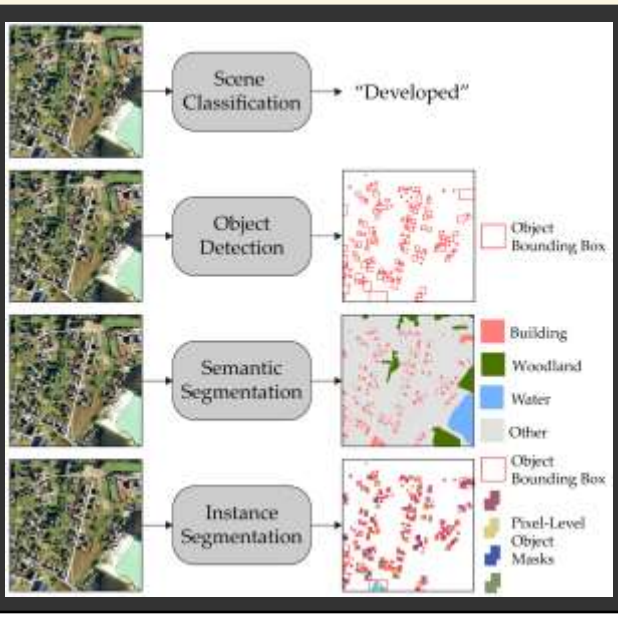
For object-detection, input training and validation data must consist of (1) images or input predictor variables, (2) coordinates that specify bounding boxes for each instance of the feature(s) of interest, and (3) class labels for each bounding box.

Instance Segmentation



Instance segmentation consists of predicting a bounding box for each detected object in an image, the associated class label, and a pixel-level mask for each predicted object. Instance segmentation combines object detection and instance segmentation.

Uses of CNN Architectures

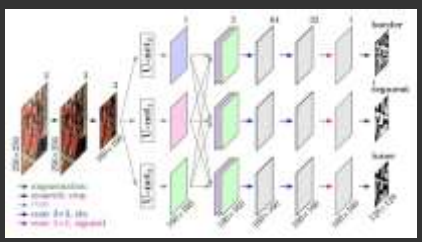


28

The geospatial instance segmentation example provided on this slide consists of identifying bounding boxes and pixel-level masks for each building in the image extent. Each building is detected as a unique instance of the building class.

UNet-ID

- Based on UNet
- Expands UNet for instance segmentation
- Three UNets: border, segment, and inner



Conceptualization from original paper (Wagner et al. 2020)

U-Net-Id, an Instance Segmentation Model for Building Extraction from Satellite Images—Case Study in the Joanópolis City, Brazil

https://zenodo.org/record/3926831#.XzhMbOhKiUk

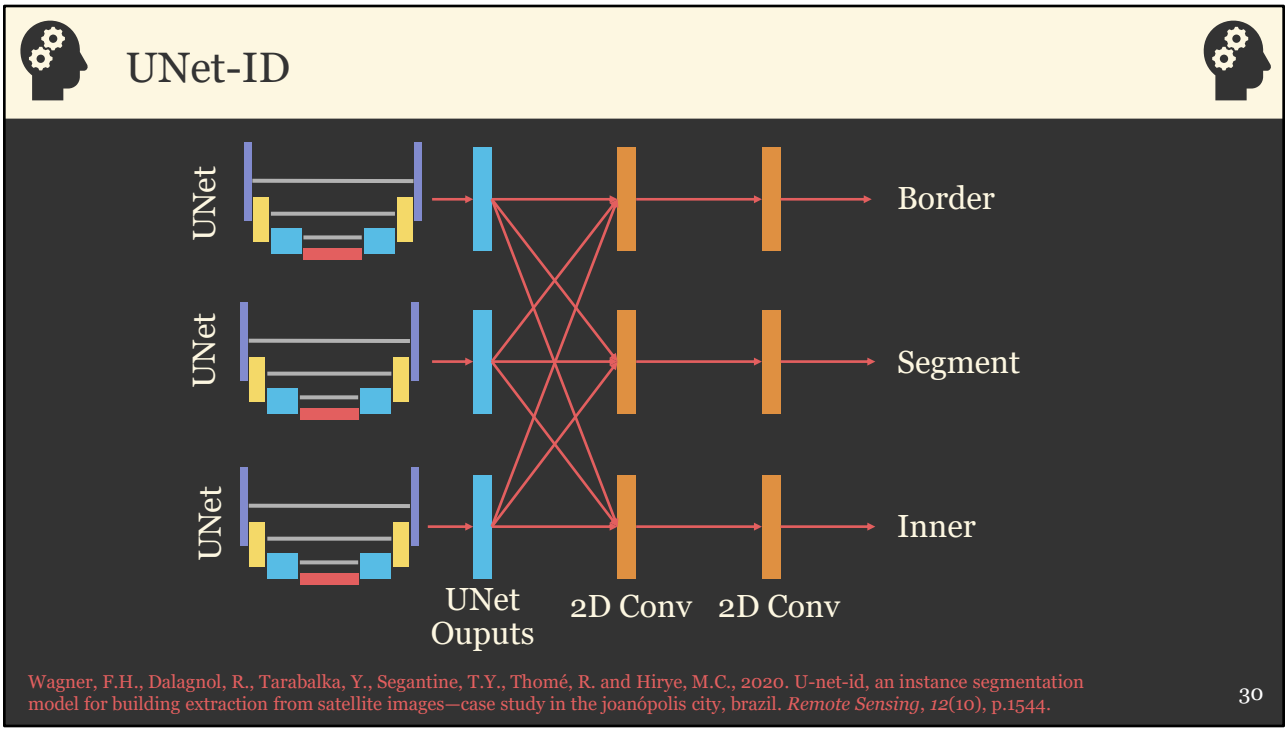
https://www.mdpi.com/2072-4292/12/10/1544

29

Some semantic segmentation architectures have been adapted for instance segmentation.

For example, UNet-Id uses multiple UNet architectures to detect individual instances of features by learning to identify features, feature boundaries, and feature interiors separately.

We will not discuss this method in detail here, but I have provided a links to the associated paper and code on this slide.



This slide highlights the architecture of UNet-ID. Three separate UNets are trained to segment feature borders, entire features, and feature interiors. The outputs from each of the three models are then combined and passed through additional 2D convolution operations so that the information from each of the three processes can be used in each prediction.

Once the three predictions are made, post-processing can be used to differentiate each unique instance.

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

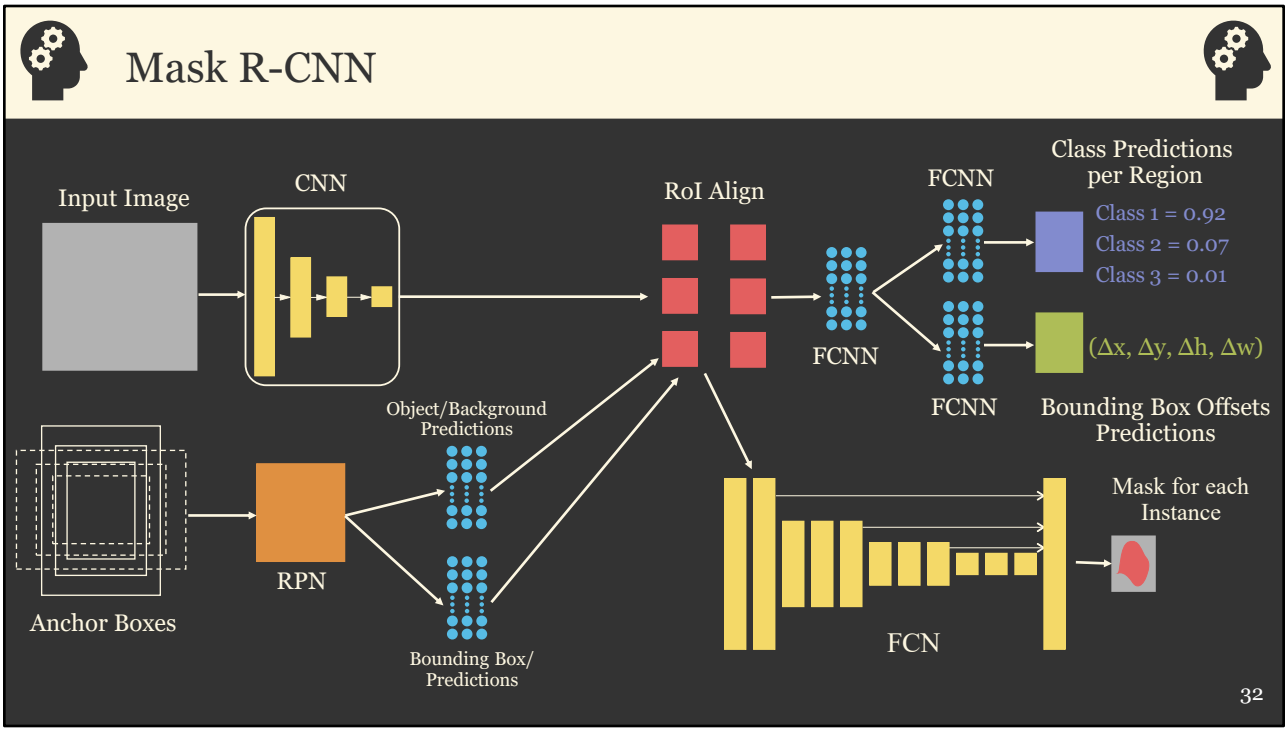
He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

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

31

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.

Mask R-CNN Output

- Class prediction
- Class probabilities
- Instance bounding box
- Instance segmentation (pixel-level)

33

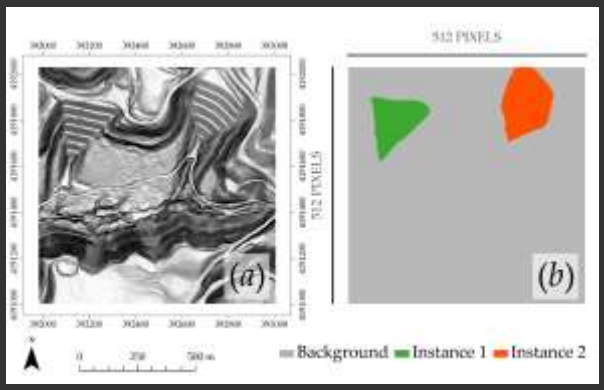
Similar to faster R-CNN, mask R-CNN generates bounding boxes and class predictions and associated probabilities.

The pixel-level segmentation is an added output not generated by faster R-CNN.

Note that it is a simple task to convert instance segmentation pixel-level masks to a semantic segmentation-style output where each pixel is labeled to a class using unique class indices.

Instance Segmentation Data Requirements

- Image chips of defined shape: (3, 512, 512), (3, 256, 256), (3, 128, 128), etc.
- Label/classified image chips:
  - Different folder for each class
  - Unique ID for each instance of each class per image chip
- Common formats: JPEG, PNG, TIFF



34

The input requirements for mask R-CNN are similar to those for semantic segmentation. Generally, you do not need to provide bounding boxes, as these can be derived from the masks.

One key difference is that each instance will need to be given a unique ID. In the provided example, the two topographic features belong to the same class but are assigned different unique numeric codes.

If multiclass classification is to be performed, then different masks for each class are commonly stored in different folders. Within each folder all masks will have unique IDs assigned to each instance of the class in the chip.

Similar to semantic segmentation methods, larger chips will require more computational power and time to process.

ArcGIS Pro/ArcPy and the ArcGIS API for Python have tools available for generating training data appropriate for instance segmentation and mask R-CNN specifically.

Mask R-CNN Implementations

https://github.com/matterport/Mask_RCNN

- PyTorch
  - https://www.learnopencv.com/mask-r-cnn-instance-segmentation-with-pytorch/
- Keras/Tensorflow
  - https://github.com/matterport/Mask_RCNN
- ArcGIS Pro
  - https://developers.arcgis.com/python/api-reference/arcgis.learn.toc.html

35

There are mask R-CNN implementations available for Keras/Tensorflow and PyTorch. The GitHub repository provided by Matterport is a common starting point.

Mask R-CNN is available in ArcGIS Pro and in the arcgis.learn module of the ArcGIS API for Python.

Panoptic Segmentation

- Instance + Semantic Segmentation
- Assign a class label and object ID to each pixel in the image
- Active area of research

36

As a final comment, at the time of this writing there is a lot of active research in panoptic segmentation methods.

Panoptic segmentation consists of labeling each pixel in the image to one of a set of pre-defined classes, similar to semantic segmentation. However, each unique instance of the class are also differentiated.

In the example, image, each pixel could be labeled to one of two classes: “background” and “cat”. The cat pixels could then be differentiated into those belonging to the “Freddy” and “Peri” instances.

We will not discuss panoptic segmentation methods in this course. However, I wanted to make you aware that this is an important active area of research.

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.