AI4EO

Special Topics in Applications (AIL861)
Artificial Intelligence for Earth Observation
Lecture 6,7

Instructor: Sudipan Saha

Image Segmentation

- ***Split image into different regions***
  - Different regions usually cover the whole image
  - Different regions usually do not overlap

- ***Similarity predicate***
  - Satisfied by each region
  - Not satisfied by union of different regions

- ***Can be subjective***
  - Depending on how we define the notion of similarity

If the similarity predicate is satisfied by the union of different regions, then it will not be possible to separate the regions and create a valid segmentation mask. A segmentation mask is a binary or multi-class image that is used to represent the result of an image segmentation process. The mask assigns a unique label or color to each distinct region or segment in the original image

Image segmentation is the process of dividing an image into multiple segments or regions, each of which corresponds to a different object or part of the image. The goal of image segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

Images are an important aspect of deep learning and computer vision, as they provide vast amounts of data for training and testing machine learning algorithms. Some of the ways images are useful in these fields include:

Object recognition: Using deep learning, algorithms can learn to identify objects within images, such as people, animals, cars, etc.

Image classification: Image classification involves assigning an image to a specific category, such as "dog", "cat", "car", etc.

Object detection: Object detection is the task of locating objects within an image and drawing a bounding box around them.

Image segmentation: As discussed earlier, image segmentation involves dividing an image into multiple segments, which can be useful for a variety of tasks, such as object recognition and image analysis.

Image generation: Using deep learning, algorithms can generate new images based on existing ones, such as generating a photorealistic image of a non-existent object.

These are just a few examples of the ways images play a crucial role in deep learning and computer vision.

Before Deep Learning

- ***Region growing***
  - Start with one pixel of a potential region and try to grow until pixels being compared are too dissimilar
- ***Clustering***
  - Name some?
- ***Split and merge***

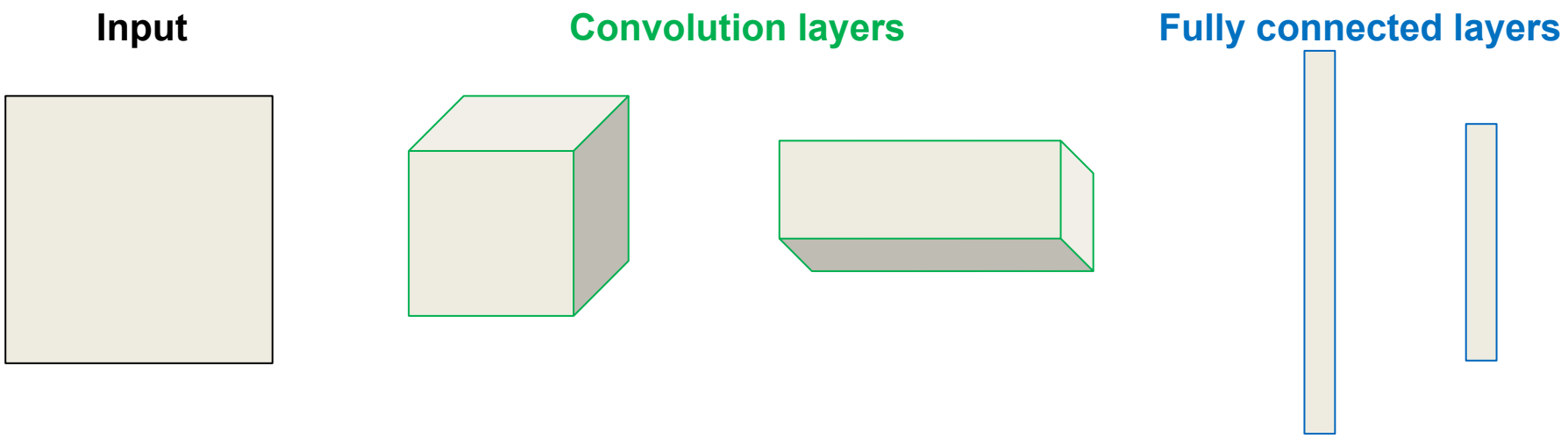
Input Space for Clustering

- ***Color***

- ***Texture feature***

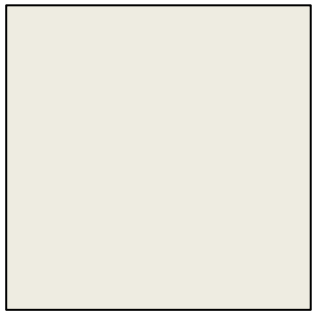
Deep Learning for Semantic Segmentation

Typical Classification Network

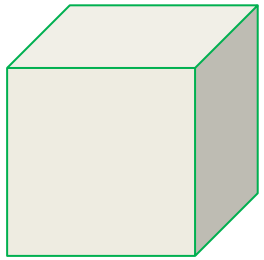


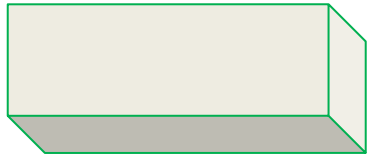
Fully Convolutional

Input



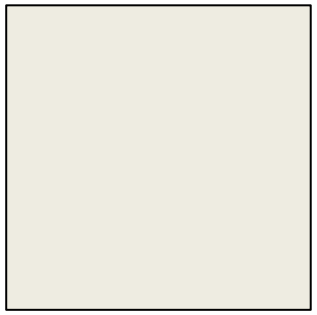
Convolution layers



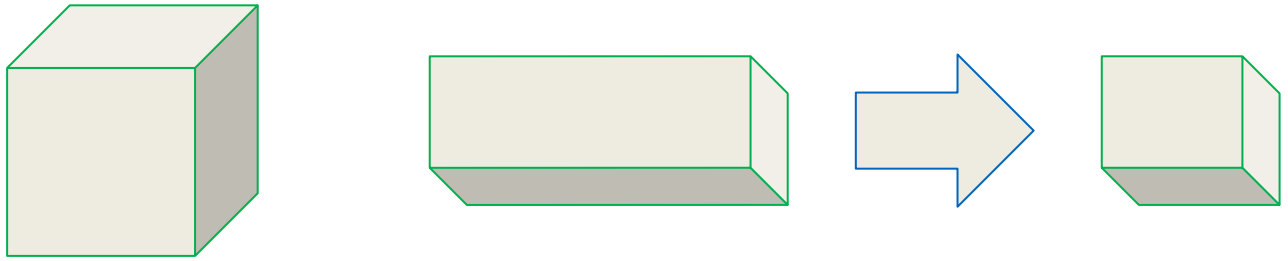


Fully Convolutional

Input

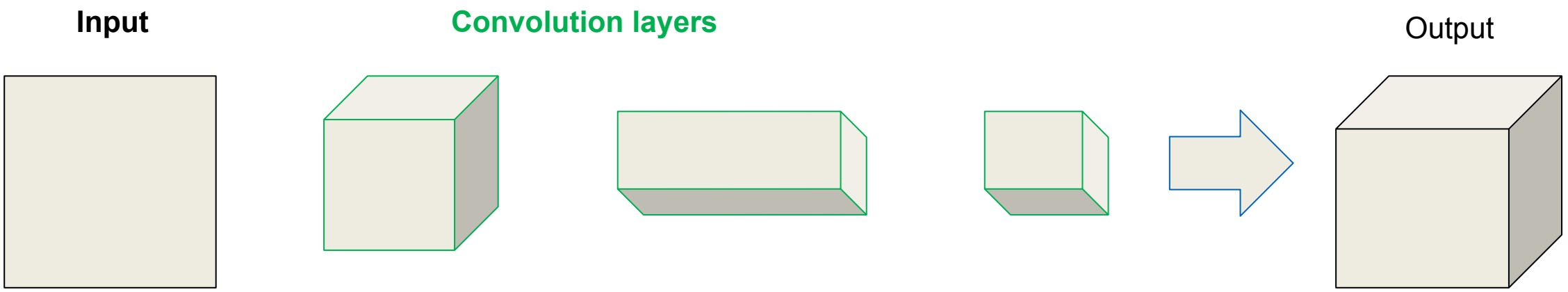


Convolution layers



Same number of channels as output class

Fully Convolutional



Classifier to Semantic Segmentation

- Convolutionalize the classification architectures: AlexNet, VGGNet
- Remove classification layer
- Append 1x1 convolution with required number of channel dimensions and upsample
- Or replace the last step with transposed convolution.

Semantic segmentation is a form of image segmentation where each pixel of an image is labeled with a class label that represents the semantic meaning of the objects present in that region. The goal of semantic segmentation is to label each pixel in an image with a class label that describes the type of object or objects that the pixel belongs to, such as "sky", "building", "person", etc. This results in a per-pixel classification of the image, providing a dense labeling of the objects present in the image. Semantic segmentation is useful in a variety of computer vision tasks, including object recognition and scene understanding

1x1 convolution is often used in semantic segmentation models to reduce the number of channel dimensions in the feature maps. This helps reduce the computational cost and memory usage of the model, as well as potentially improving its performance by reducing overfitting.

Upsampling is used to increase the spatial resolution of the feature maps, which is typically necessary in semantic segmentation to obtain a dense labeling of the objects in the image. Upsampling can be performed using various techniques, such as transposed convolutions, nearest-neighbor interpolation, or bilinear interpolation.

Appending the 1x1 convolution with the upsampling step allows the model to both reduce the number of channel dimensions and increase the spatial resolution of the feature maps. This can help the model achieve a good balance between computational efficiency and performance, as well as improving the quality of the predicted segmentation masks.

Q. explain more about why reduce the number of channel dimensions by 1x1 convolution improve its performance by reducing overfitting?

In deep learning, overfitting occurs when a model learns the training data too well and is unable to generalize to new, unseen data. This is often caused by having too many parameters in the model, which allows it to fit the training data too closely.
A 1x1 convolution can be used to reduce the number of channel dimensions in the feature maps, which in turn reduces the number of parameters in the model. By reducing the number of parameters, the risk of overfitting is also reduced, as there are fewer parameters that can be learned specifically for the training data.

Additionally, a 1x1 convolution can also help to alleviate overfitting by introducing a form of spatial bottleneck. This means that the model must learn to preserve the important information from the input feature maps while reducing the number of channel dimensions. This can help to prevent the model from memorizing the training data and improve its ability to generalize to new data.

In summary, reducing the number of channel dimensions through a 1x1 convolution can improve the performance of semantic segmentation models by reducing the risk of overfitting, and by introducing a form of spatial bottleneck that can improve the model's ability to generalize.

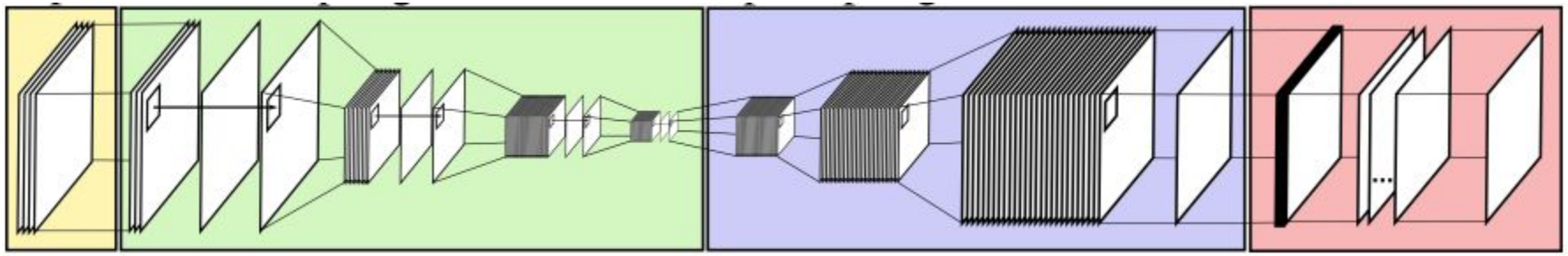
Q. How introducing a form of spatial bottleneck that can preserve the important information?

A spatial bottleneck is created by reducing the spatial resolution of the feature maps, typically through a combination of pooling layers and strided convolutions. By reducing the spatial resolution, the model is forced to learn a more compact representation of the input image, which can help to preserve the important information from the input feature maps.

When a 1x1 convolution is used to reduce the number of channel dimensions, it can be seen as a form of spatial bottleneck in that it forces the model to learn a more compact representation of the feature maps. This is because the 1x1 convolution acts as a form of channel-wise feature selection, where only the most important information from each channel is retained. The 1x1 convolution can help to preserve the important information from the feature maps by selecting the most important activations and discarding the less important ones.

By introducing a form of spatial bottleneck, the model is encouraged to learn a more compact and meaningful representation of the input image, which can help to preserve the important information and improve the performance of the model.

CNN for Semantic Segmentation of EO Images



Dense Semantic Labeling of Subdecimeter Resolution Images With Convolutional Neural Networks, 2017

Random Sampling

Semantic classes are unevenly distributed spatially and their frequency highly varies. The class “road” is ubiquitous, while the class “car” is localized and rare. Consequently, it is beneficial to sample the training patches randomly in space (among the training images) and with respect to class frequencies.

Challenges for Semantic Segmentation

- What and where
- Classes with similar spectral signature
- Inconspicuous classes
- *Solution for all above problems:* context, context at multiple scales, global context

Contraction Phase

- Reduces spatial size
- Increases understanding of content (“what”)
- Though loses some object information due to contraction (loses “where” to some extent)

The contraction phase, also known as the encoder phase, is a part of the architecture in Convolutional Neural Networks (CNNs) used for tasks such as image classification, segmentation, and generation. It refers to the layers of the network responsible for reducing the spatial resolution of the feature maps and increasing the number of channel dimensions.

In the contraction phase, the model learns a compact representation of the input image by using convolutional layers, pooling layers, and/or strided convolutions to reduce the spatial resolution of the feature maps. This allows the model to capture the high-level, abstract features of the image, such as the presence of certain objects, textures, and shapes.

The contraction phase is typically followed by the expansion phase, also known as the decoder phase, which increases the spatial resolution of the feature maps to produce the final output. The contraction and expansion phases together form the backbone of many successful CNN architectures, such as U-Net, SegNet, and ResNet, among others.

The main disadvantage of the contraction phase in Convolutional Neural Networks (CNNs) is that it reduces the spatial resolution of the feature maps, which can result in the loss of fine-grained details and information from the input image.

As the spatial resolution of the feature maps is reduced, the model becomes less sensitive to small and intricate structures in the input image. This can lead to inaccurate predictions or misclassifications, particularly in tasks such as semantic segmentation and object detection, where fine-grained details are crucial for obtaining accurate results.

Another disadvantage of the contraction phase is that it increases the number of channel dimensions, which can lead to increased computational cost, memory usage, and risk of overfitting. As the number of channel dimensions increases, the model becomes more complex, and the risk of overfitting also increases, particularly when the number of training samples is limited.

In conclusion, while the contraction phase is crucial for reducing the spatial resolution of the feature maps and capturing high-level features of the input image, it can also result in the loss of fine-grained details and information, and lead to increased computational cost and risk of overfitting.

Recover “where” in expansion phase

- Concatenate feature maps from contraction phase during expansion phase during recovery
- Helps in recovery of “where” information
- Principle behind U-Net

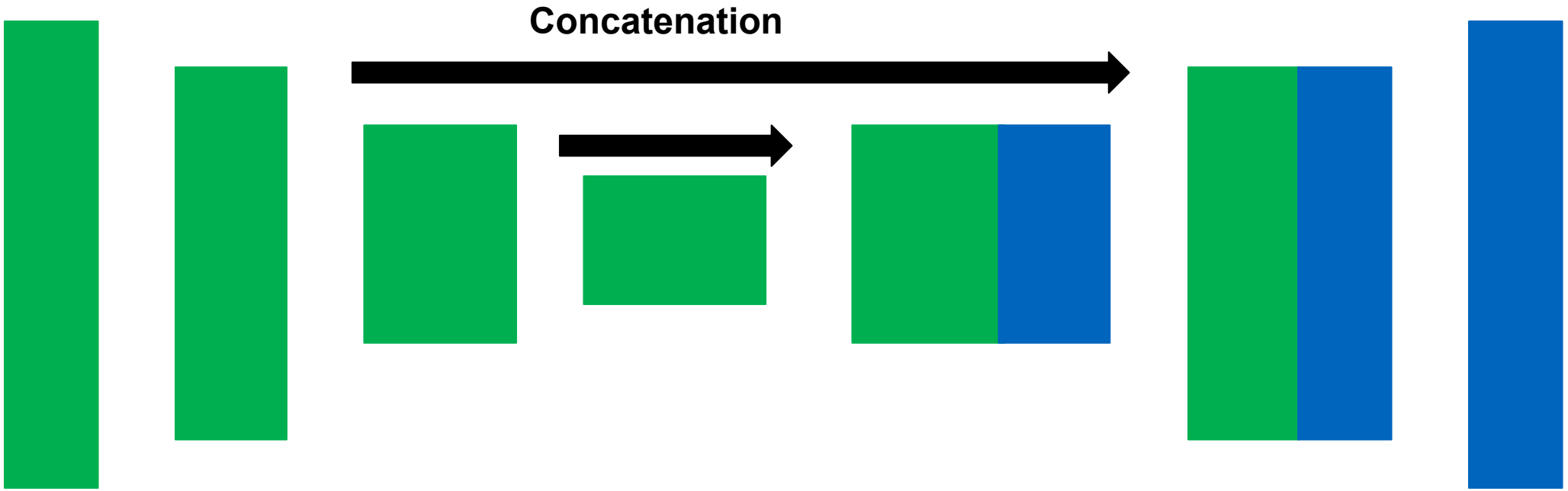
The expansion phase, also known as the decoder phase, is a part of the architecture in Convolutional Neural Networks (CNNs) used for tasks such as image classification, segmentation, and generation. It refers to the layers of the network responsible for increasing the spatial resolution of the feature maps and reducing the number of channel dimensions.

In the expansion phase, the model uses transposed convolutions, upsampling, and/or concatenation operations to increase the spatial resolution of the feature maps. This allows the model to recover the fine-grained details and spatial information that was lost in the contraction phase.

The expansion phase is typically preceded by the contraction phase, also known as the encoder phase, which reduces the spatial resolution of the feature maps and increases the number of channel dimensions. The contraction and expansion phases together form the backbone of many successful CNN architectures, such as U-Net, SegNet, and ResNet, among others.

The goal of the expansion phase is to produce a high-resolution output that accurately captures both the high-level, abstract features and the fine-grained details of the input image. The expansion phase plays a crucial role in tasks such as semantic segmentation, where accurate boundary detection and fine-grained details are critical for obtaining accurate predictions.

U-Net



U-Net is a type of Convolutional Neural Network (CNN) architecture that is widely used in image segmentation tasks. It was originally proposed for biomedical image segmentation, but it has since been applied to a wide range of other domains as well.

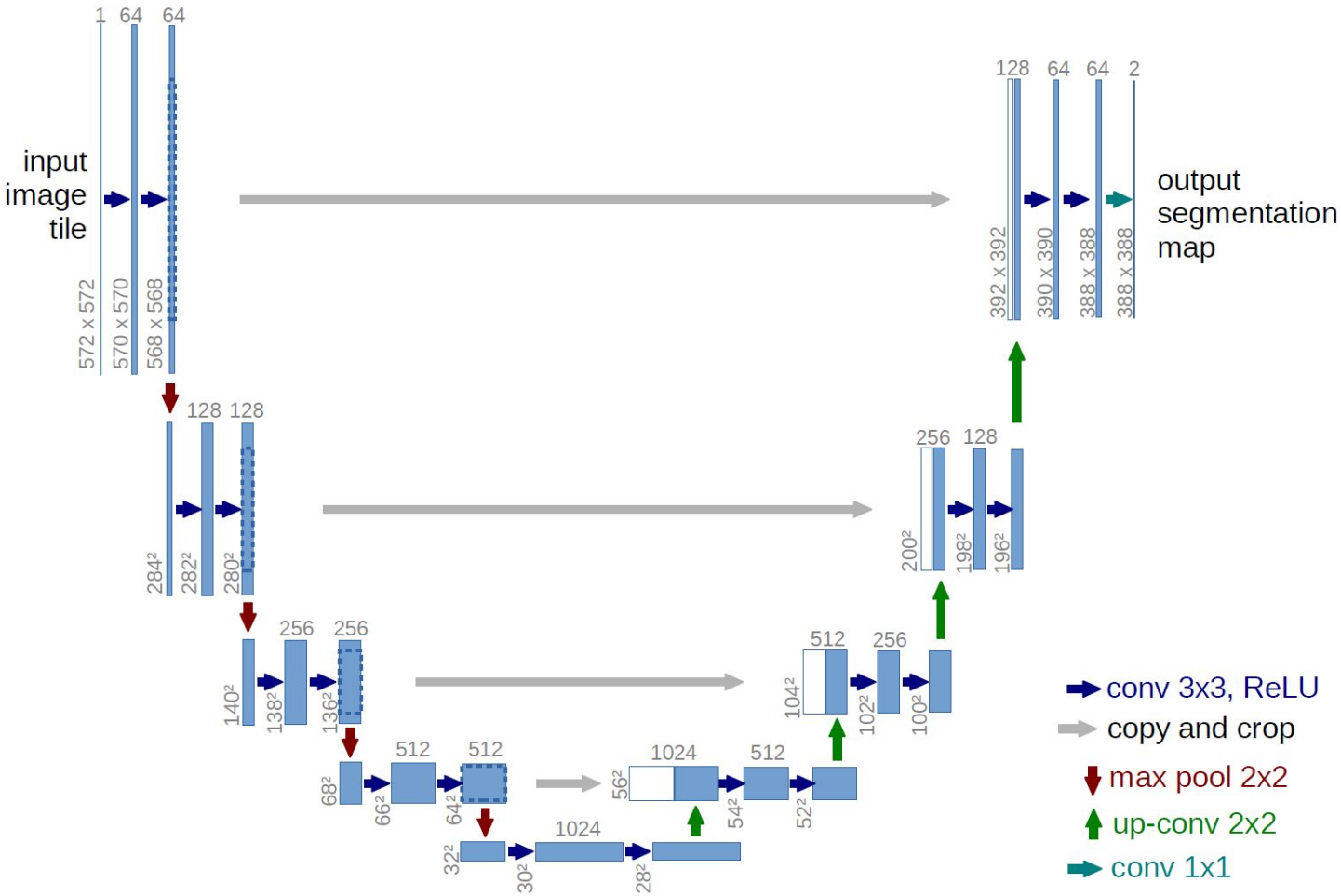
U-Net is characterized by its unique architecture, which consists of two main parts: the contraction phase, also known as the encoder, and the expansion phase, also known as the decoder. The encoder reduces the spatial resolution of the input image and increases the number of channel dimensions, while the decoder increases the spatial resolution and reduces the number of channel dimensions.

The encoder and decoder are connected by skip connections, which pass information from the encoder to the decoder, allowing the decoder to recover the fine-grained details and spatial information that was lost in the encoder. This allows U-Net to produce high-resolution, fine-grained predictions that accurately capture both the high-level, abstract features and the fine-grained details of the input image.

U-Net is widely used for image segmentation tasks due to its ability to effectively capture fine-grained details and spatial information, and its ability to handle large amounts of noise and uncertainty in the input data. Additionally, U-Net is relatively easy to implement and train, making it a popular choice for researchers and practitioners working in the field of image segmentation.

In summary, U-Net is a popular and effective CNN architecture for image segmentation tasks, characterized by its unique encoder-decoder architecture and skip connections, which allow it to effectively capture fine-grained details and spatial information in the input image.

U-Net



Ronneberger et al., 2015

U-Net

Original loss function: Gives weightage to edge pixels (“ small separation borders that we introduce between touching cells”)

U-Net Flood Segmentation

See “Flood segmentation on Sentinel-1 SAR imagery with semi-supervised learning”: an interesting case that combines U-Net based segmentation with softmax based confidence estimation.

"Flood segmentation on Sentinel-1 SAR imagery with semi-supervised learning" is a study that combines the strengths of two different machine learning techniques to address the challenge of flood segmentation in satellite imagery.

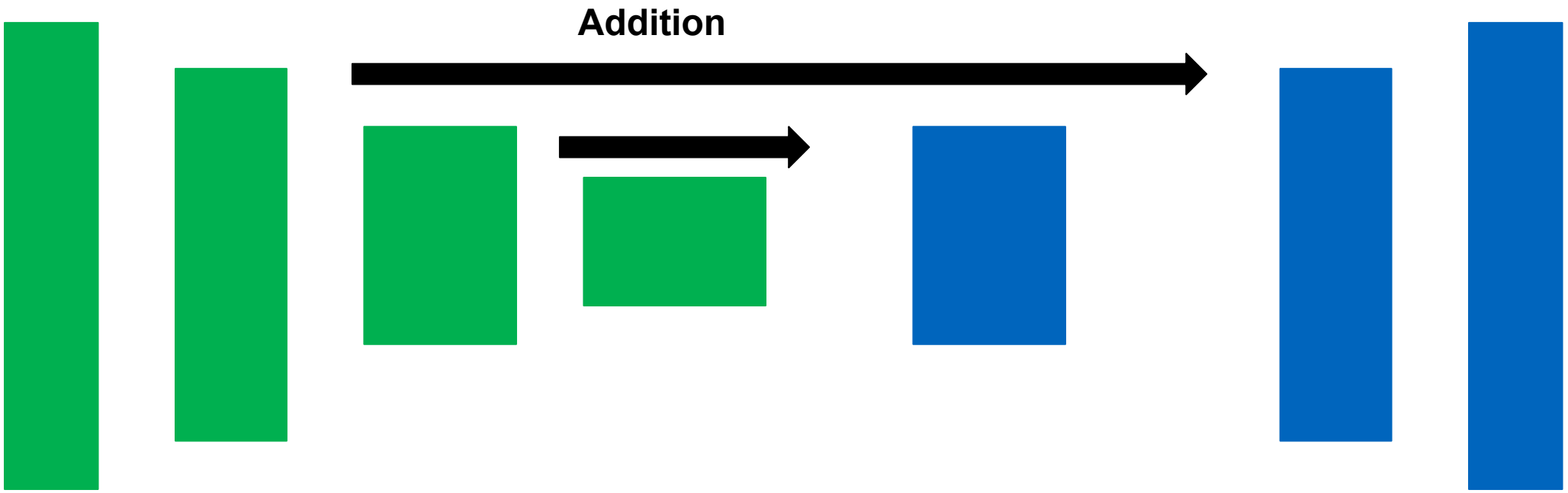
The study focuses on using Sentinel-1 Synthetic Aperture Radar (SAR) imagery, which provides valuable information about flood events, as well as other environmental and geophysical phenomena. However, the task of segmenting flooded areas from SAR imagery is challenging due to the presence of noise, artifacts, and other confounding factors in the data.

To address this challenge, the study combines U-Net based image segmentation with softmax based confidence estimation. U-Net is a popular and effective Convolutional Neural Network (CNN) architecture that is widely used in image segmentation tasks. The study employs U-Net to segment flooded areas from the Sentinel-1 SAR imagery, using a semi-supervised learning approach that incorporates both labeled and unlabeled data.

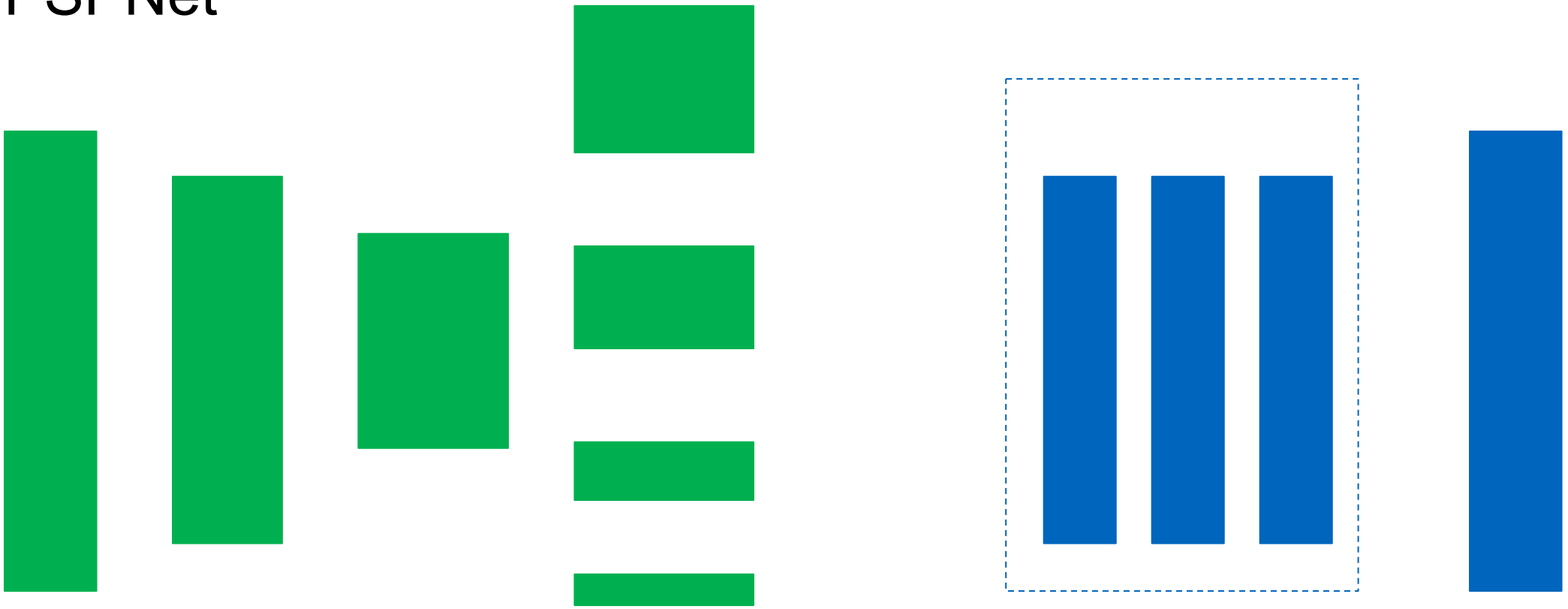
Softmax based confidence estimation is a statistical method that is used to estimate the confidence or uncertainty of a prediction. In the study, softmax is used to estimate the confidence of the U-Net predictions, allowing the researchers to identify and exclude predictions with low confidence, which are more likely to be incorrect.

By combining U-Net based image segmentation with softmax based confidence estimation, the study is able to achieve high accuracy in flood segmentation, while also effectively reducing the impact of noise, artifacts, and other confounding factors in the data. This is an interesting case that demonstrates the potential of combining different machine learning techniques to address challenging real-world problems, such as flood segmentation in satellite imagery.

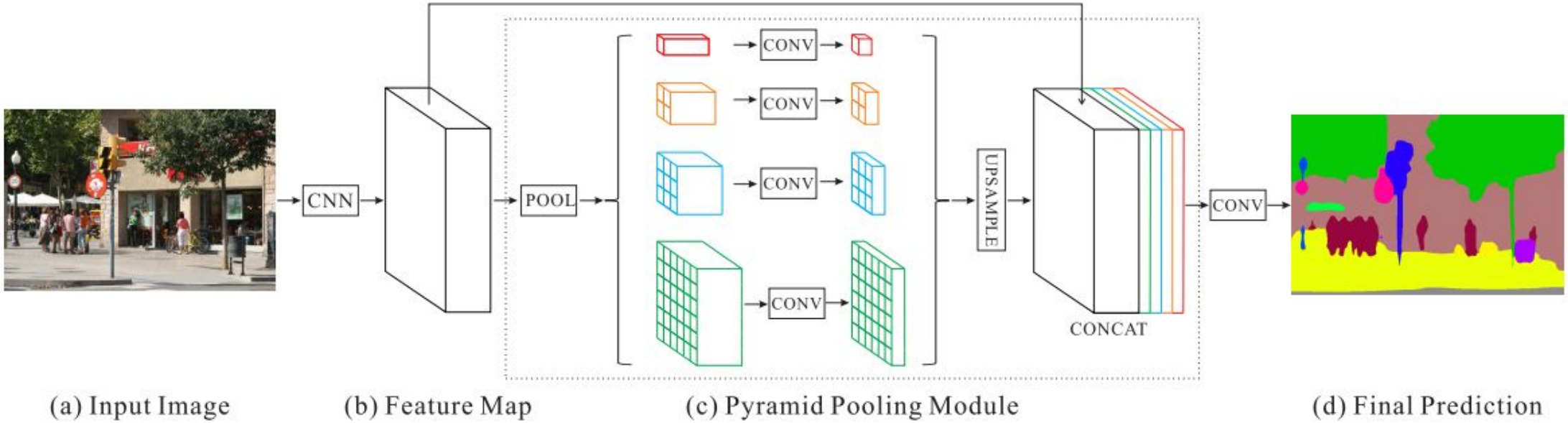
Linknet



PSPNet

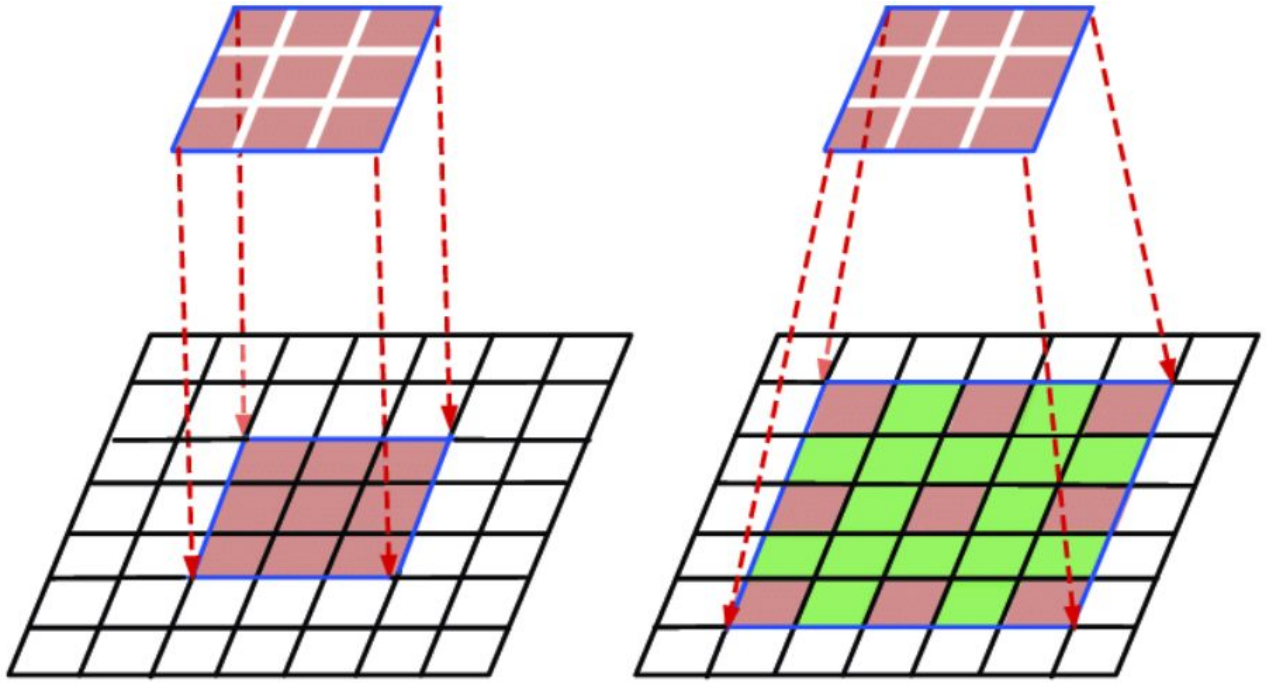


PSPNet


CNN
POOL
CONV
CONV
CONV
CONV
UPSAMPLE
CONCAT
CONV
(a) Input Image
(b) Feature Map
(c) Pyramid Pooling Module
(d) Final Prediction

DeepLabV3+

Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, Chen et al., 2018



Inference Time

DeepLab V3 > UNet > FCN

Evaluation Indices

Pixelwise accuracy

Precision, recall, F1-score are computed over each class

Mean-IoU: IoU is computed over each class and mean is computed.

Semantic Segmentation with Weak (Image Level) Annotations

- Train a semantic segmentation model with image-level labels (same label for each pixel)
- Use the above model to pseudo label the images and train again
- Keep repeating

Semantic Segmentation with Sparse Annotations

- Use sparse annotations for supervised training
- For each pixel, find the closest pixel in feature space. Further, minimize distance between two.
- For each pixel, find the farthest pixel in feature space. Further, maximize distance between two.
- For each pixel, find the most similar pixel among 8 neighbors. Further, minimize distance between two.

Semantic segmentation with sparse annotations refers to the task of assigning semantic labels to each pixel in an image, but with only a limited number of labeled pixels available for training the model. Sparse annotations typically consist of only a small fraction of the total number of pixels in the image, and the goal is to train a model that can accurately segment the image despite the limited amount of labeled data.

This is a challenging problem because semantic segmentation models typically require large amounts of labeled data to learn to accurately segment an image. However, annotating large datasets can be time-consuming and expensive, making sparse annotations an attractive alternative.

To tackle this challenge, various techniques have been proposed, including transfer learning from pre-trained models, data augmentation, and weakly supervised learning. In weakly supervised learning, the model is trained on image-level labels, which are much easier to obtain than pixel-level labels. The model then learns to propagate the information from the image-level labels to the pixel-level predictions.

Semantic segmentation with sparse annotations is a relevant problem in many real-world applications, such as medical imaging, remote sensing, and object recognition, where obtaining large amounts of labeled data can be difficult or impractical. The goal of this research area is to develop methods that can effectively segment images using limited labeled data, making it possible to apply semantic segmentation to a wider range of applications.

SAR segmentation

Heterogeneous Feature Distillation Network for SAR Image Semantic Segmentation, Gao and Dong, 2022

Uses optical image as an additional support for segmenting SAR image.

SAR Segmentation

What if you do not have paired optical - SAR images for training?

If you do not have paired optical-SAR images for training, then you have several options:

Pretrained models: You can use pre-trained models that have been trained on large datasets and then fine-tune them on your SAR data. This approach can be effective when the pre-trained models are trained on similar data and can provide a good starting point for your SAR data.

Unpaired data: You can use unpaired optical and SAR data for training. This approach involves training two separate models, one for optical images and one for SAR images, and then combining the results. The challenge in this approach is finding a way to effectively align the results from the two models, as the data is not paired.

Weakly supervised learning: You can use weakly supervised learning, where you have only image-level labels for your data and not pixel-level labels. The model then learns to propagate the information from the image-level labels to the pixel-level predictions.

Synthetic data: You can generate synthetic data that pairs optical and SAR images. This approach involves using computer graphics to simulate SAR images that correspond to optical images. This can be useful when paired data is not available, but it requires a significant amount of work to generate the synthetic data.

Each of these options has its advantages and disadvantages, and the best approach depends on the specific requirements of the problem and the available resources.