

Special Topics in Applications (AIL861)

Artificial Intelligence for Earth Observation

Lecture 4

Instructor: Sudipan Saha

Using models trained in supervised fashion on some other task

- ***Just use as feature extractor***

- Just use the features extracted from particular layer(s) without any tuning

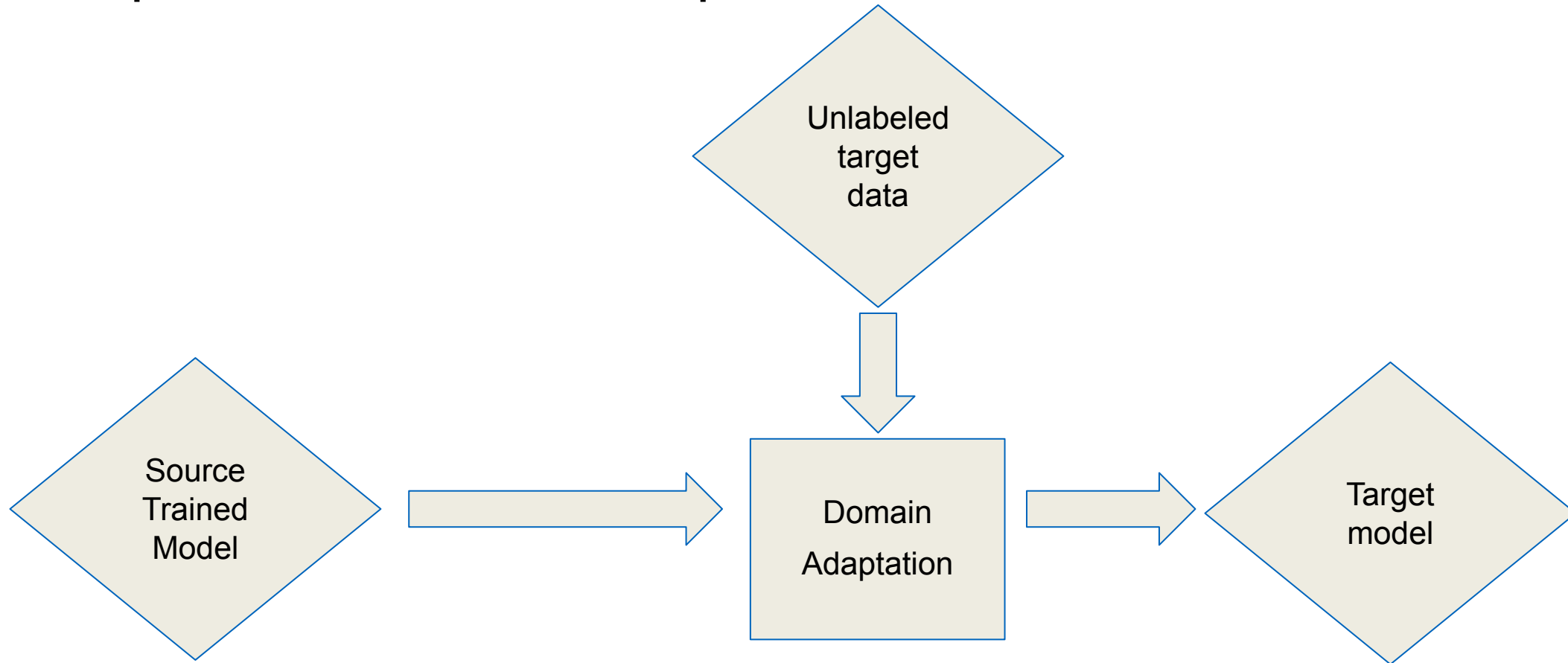
- ***Fine tuning on target data***

- Further train particular layers of the network

- ***Unsupervised domain adaptation***

- Adapt the network with unlabeled target data

Unsupervised Domain Adaptation



Unsupervised Domain Adaptation (UDA) is a type of machine learning technique used to adapt a model trained on one dataset (the source domain) to work well on a different dataset (the target domain), where the target domain data is not labeled. This is often used when the source domain data is abundant and well-labeled, but the target domain data is limited or not labeled at all. UDA can improve the performance of a model on the target domain by aligning the feature distributions of the source and target domains.

UDA typically works by learning a common feature representation for both the source and target domains, such that the features from the two domains are "aligned" in some sense. This alignment can be achieved in a number of ways, depending on the specific UDA method used. Some common techniques include:

Adversarial Training: This method involves training a model to discriminate between the source and target domains, while also training a feature extractor to produce features that are indistinguishable between the two domains.

Maximum Mean Discrepancy (MMD): This method seeks to minimize the difference in the means of the feature distributions of the source and target domains.

Correlation Alignment (CORAL): This method seeks to maximize the correlation between the source and target domain features.

Self-training: This method uses the model trained on the source domain to predict labels for the target domain, and then uses these predicted labels to fine-tune the model on the target domain.

Gradient Reversal Layer (GRL): This method introduces a gradient reversal layer in the neural network architecture which multiplies the gradient from the last layer by -1 during the back-propagation, thus reversing the direction of the gradients.

These are some of the common techniques used for UDA but depending on the problem and the data set, different methods can be used.

Unsupervised Domain Adaptation

- Problem setting: Given an **annotated** dataset from *source* domain and an **unlabelled** dataset from the *target* domain, the goal is to adapt the predictor to work well in the *target* domain.
- Assumption: the source and target *semantics* do not change.
- Formally:

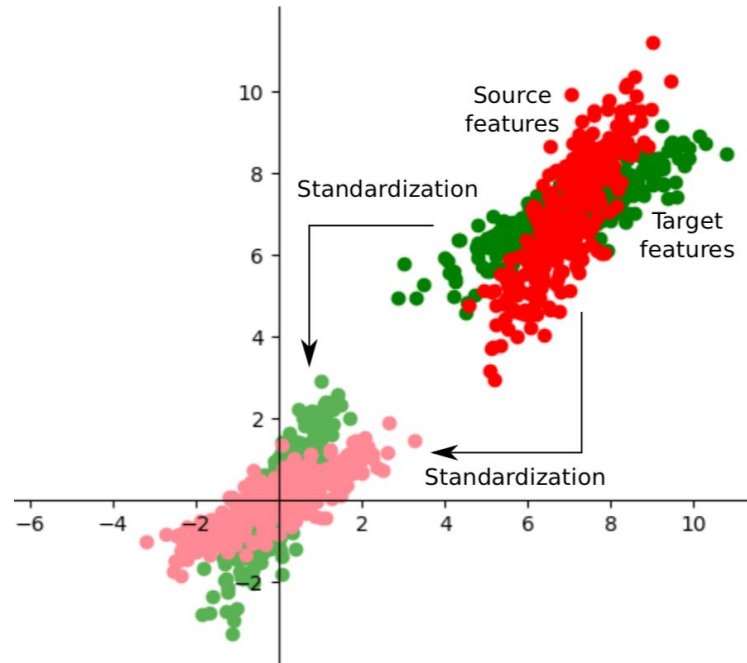
$$\mathcal{D}^s = \{(x_i^s, y_i^s)\}_{i=1}^N$$

$$\mathcal{D}^t = \{(x_j^t)\}_{j=1}^M$$

$$f_\theta : X \rightarrow Y$$

Unsupervised Domain Adaptation

- **Batch Normalization** based DA methods align feature distributions through feature standardization by setting mean of features to 0 and variance to 1, but they leave the feature correlations intact, leading to imperfect alignment.



Batch Normalization (BN) is a technique used to normalize the activations of a neural network across different layers and examples, by subtracting the mean and dividing by the standard deviation. Batch Normalization can also be used to align the feature distributions of the source and target domains in Unsupervised Domain Adaptation (UDA).

Ack: Subhankar Roy, UniTn for this slide

Unsupervised Domain Adaptation

- ✓ Domain adversarial training

- ✓ Domain translation

SEMI2I: Semantically Consistent Image-to-Image Translation for Domain Adaptation of Remote Sensing Data

- ✓ Let's learn more here:

https://cs330.stanford.edu/lecture_slides/cs330_domain_adaptation_2022.pdf

Domain Adversarial Training (DAT) is a method of using adversarial training in Unsupervised Domain Adaptation (UDA) to align the feature distributions of the source and target domains.

The basic idea behind DAT is to train a feature extractor and a domain classifier simultaneously, where the feature extractor is trained to extract features that are useful for the main task (e.g. image classification), and the domain classifier is trained to discriminate between the source and target domains. The goal is to learn a feature extractor that produces features that are indistinguishable between the source and target domains, so that the domain classifier cannot accurately predict the domain labels.

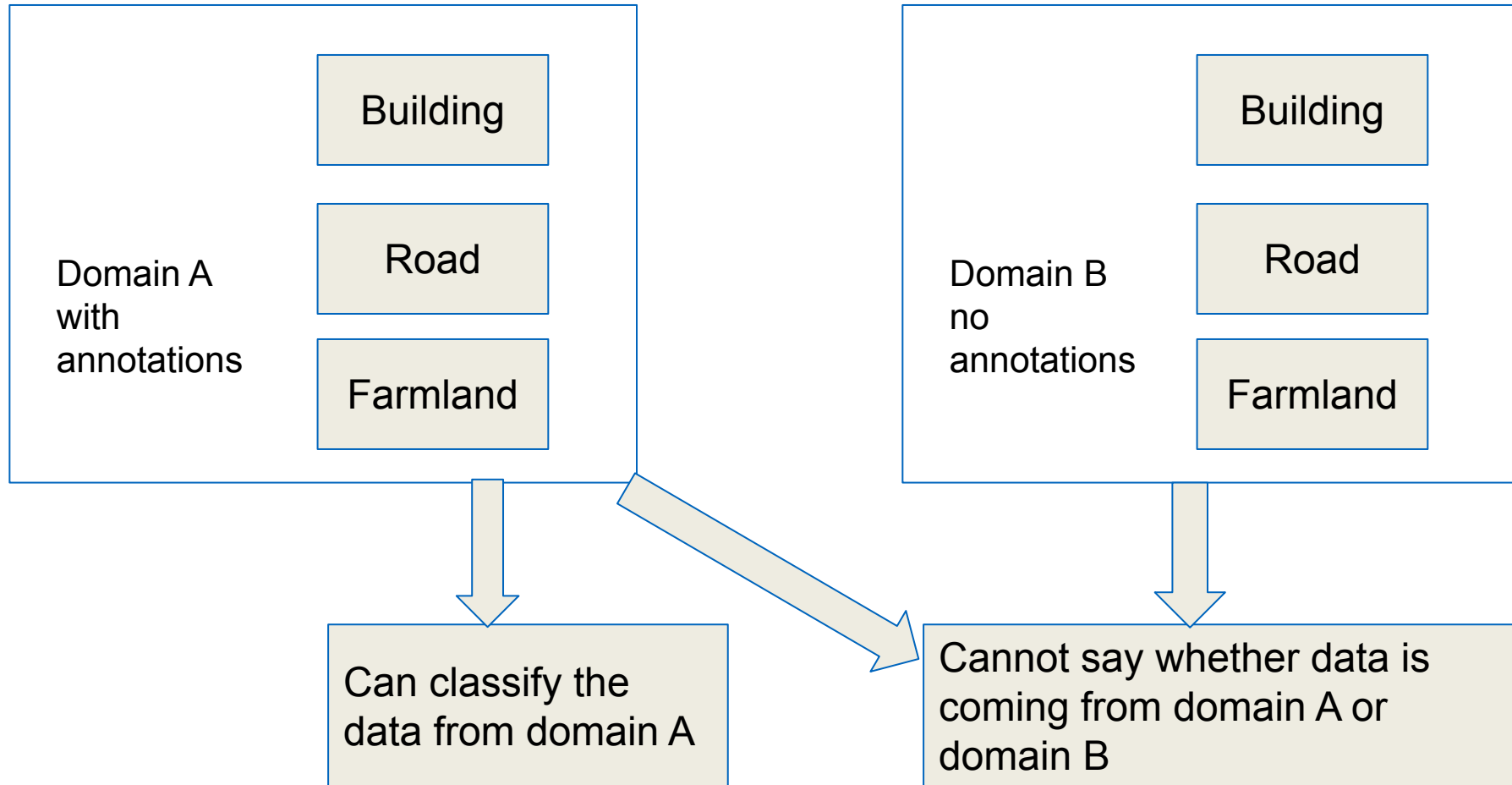
The process of DAT can be broken down into the following steps:

1. Train a feature extractor on the source domain data and use it to extract features from the source and target domains.
2. Train a domain classifier that takes the features as input and outputs a domain label (source or target).
3. Use the feature extractor and domain classifier to define a loss function that encourages the feature extractor to produce features that are difficult to classify by the domain classifier.
4. Minimize this loss function using backpropagation, this will update the feature extractor's weights.
5. Repeat steps 1-4 for multiple iterations.

By training the feature extractor and domain classifier together in an adversarial way, the feature extractor is forced to produce features that are indistinguishable between the source and target domains, thus aligning the feature distributions of the two domains. This method can improve the performance of the UDA model on the target domain by reducing the domain shift between the source and target domains.

It's worth noting that, DAT is a type of Generative Adversarial Networks (GAN) where the generator is the feature extractor and the discriminator is the domain classifier.

Domain Adversarial Training



Domain Translation

✓ Conditional GAN

✓ Example

SEMI2I: Semantically Consistent Image-to-Image Translation for Domain Adaptation of Remote Sensing Data

Different Learning Paradigms

- **Supervised learning** – learning with **labeled data**
 - Approach: collect a large dataset, manually label the data, train a model, deploy
 - Learned **feature representations** on large datasets can be transferred via pre-trained models to smaller domain-specific datasets
- **Unsupervised learning** – learning with **unlabeled data**
 - Approach: discover patterns in data either via clustering similar instances, or density estimation, or dimensionality reduction ...
- **Self-supervised learning** – representation learning with **unlabeled data**
 - Learn useful **feature representations** from unlabeled data through **pretext tasks**
 - The term “self-supervised” refers to creating **its own supervision** (i.e., without supervision, without labels)
 - Self-supervised learning is one category of unsupervised learning

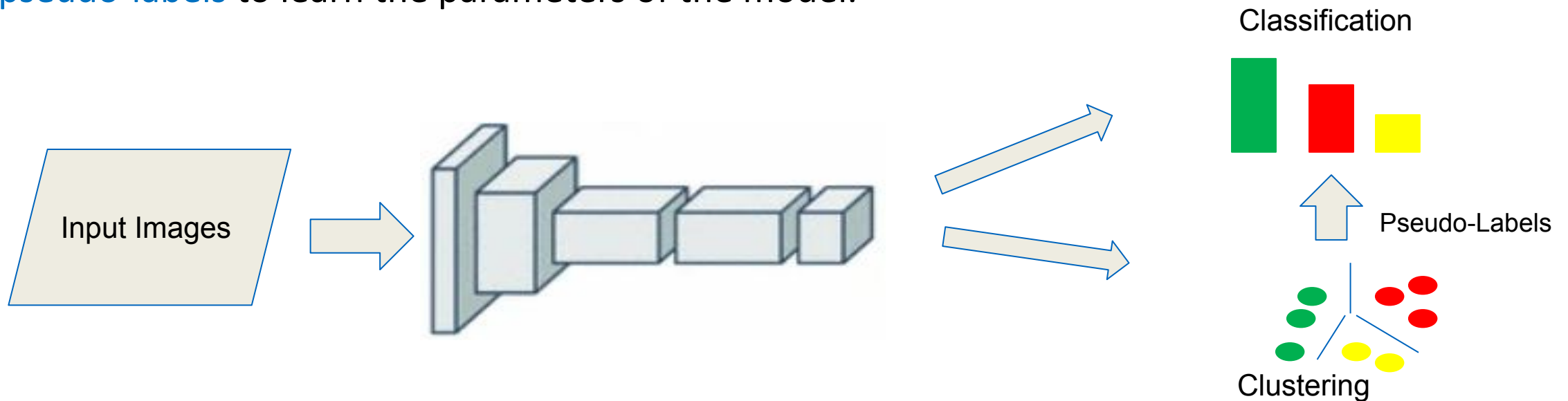
Unsupervised Learning

Unsupervised Learning

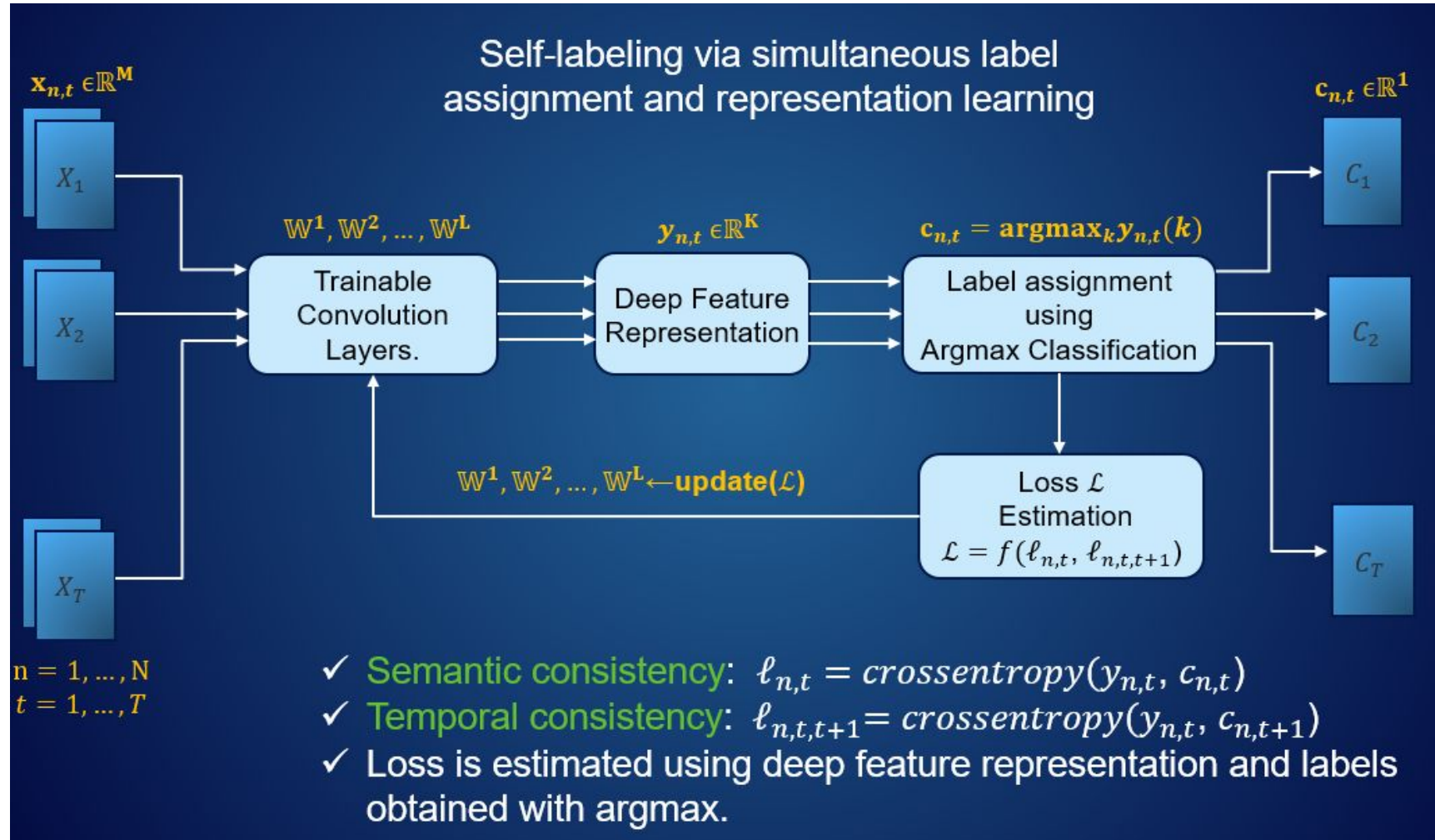
- *Unsupervised learning* – learning with **unlabeled data**
 - Deep Clustering
 - Approximating distribution (Generative Adversarial Network)
 - Learning based on dimensionality reduction (autoencoder)

Deep Clustering

- “**Deep Clustering** for Unsupervised Learning of Visual Features” – Caron et. al.
- Key Idea: **Iterative clustering** of deep features and using the cluster assignments as **pseudo-labels** to learn the parameters of the model.



Deep Multi-Temporal Clustering



Deep Multi-Temporal Clustering (DMTC) is a method of unsupervised learning that combines the use of deep neural networks with clustering algorithms to discover the underlying structure of multi-temporal data. Multi-temporal data refers to data collected at different times, such as satellite images or weather data.

The basic idea behind DMTC is to train a neural network to extract features from the multi-temporal input data, and then use a clustering algorithm to group the data points based on their feature representations. The neural network is trained to maximize the performance of the clustering algorithm, typically by minimizing the reconstruction error between the input data and the feature representations.

The process of DMTC can be broken down into the following steps:

- Train a neural network to extract features from the multi-temporal input data, typically using an autoencoder or a convolutional neural network (CNN) architecture.
- Use the feature representations produced by the neural network as input to a clustering algorithm, such as k-means, to group the data points into clusters.
- Use the clusters to train a classifier
- Use the classifier to predict on new data

DMTC has been used for a variety of tasks such as image classification, weather forecasting, and anomaly detection. It has the advantage of being able to discover the underlying structure of the data without the need for labeled data, which is particularly useful in cases where labeled data is scarce or expensive to obtain.

It's worth noting that, Deep Multi-Temporal Clustering is a challenging task, and it's difficult to find the optimal architecture and settings for the neural network and the clustering algorithm. Also, the performance of deep clustering depends on the quality of the feature representations produced by the neural network, which can be affected by factors such as the size and architecture of the network, the training data, and the optimization algorithms used.

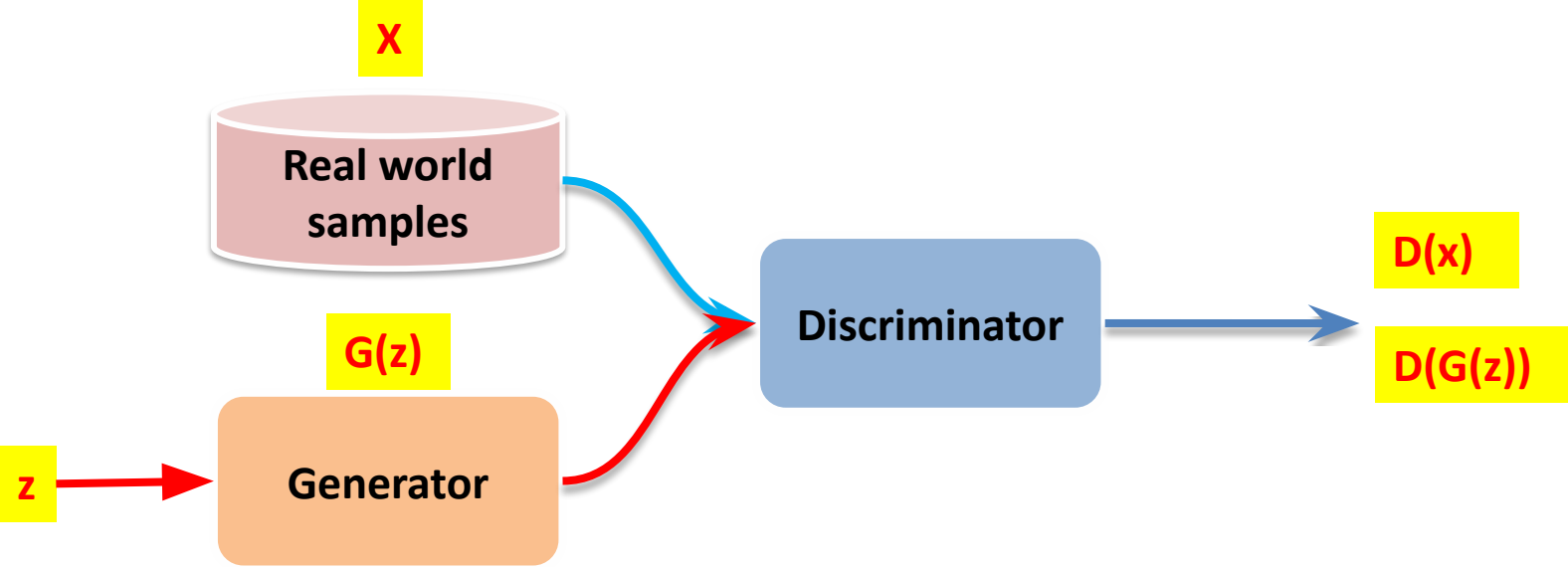
Unsupervised Learning

- *Unsupervised learning* – learning with unlabeled data
 - Deep Clustering
 - Approximating distribution (Generative Adversarial Network)
 - Learning based on dimensionality reduction (autoencoder)

Generative Adversarial Network

- Implicit density estimation approach
- In basic setup: one generator and one discriminator
- Generator tries to mimic a data distribution
- Discriminator tries to distinguish between real data and fake data (i.e., generated by the generator)
- Inspiration from game theory
- Objective is to reach Nash equilibrium
- Many design choices, e.g., maybe weaken the discriminator intentionally?

GAN



Generative Adversarial Networks (GANs) are a type of unsupervised generative model that consist of two main components: a generator and a discriminator. The generator is trained to produce new samples that are similar to the training data, while the discriminator is trained to determine whether a given sample is real (from the training data) or fake (generated by the generator). The two components are trained in an adversarial way, where the generator is trying to produce samples that can fool the discriminator, and the discriminator is trying to correctly identify whether a sample is real or fake.

The generator is typically a neural network that takes a random noise vector as input and generates samples that should be similar to the training data. The generator is trained to minimize a loss function that compares the generated samples to the real samples. The discriminator is also a neural network that takes a sample as input and outputs a scalar that represents the probability that the sample is real. The discriminator is trained to maximize this probability for real samples and minimize it for fake samples.

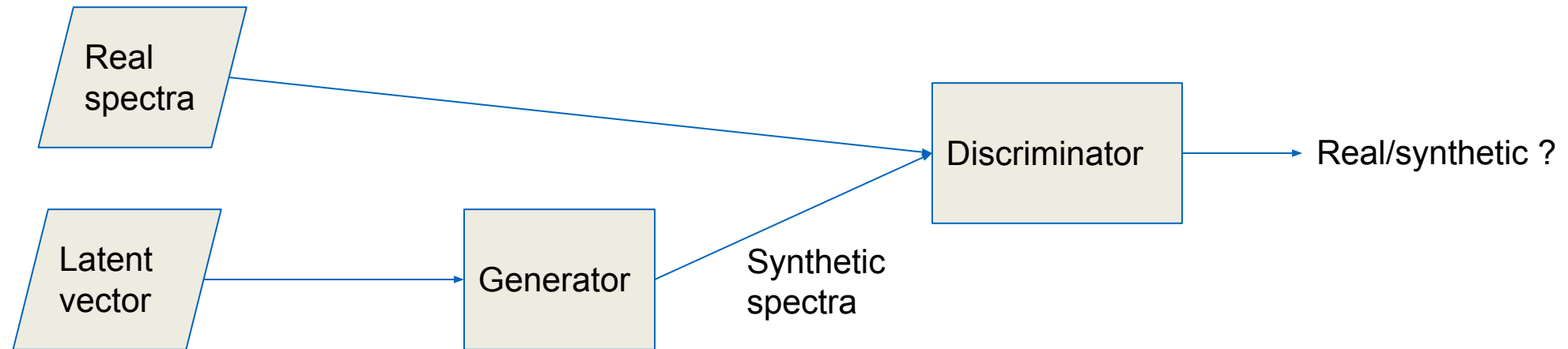
During training, the generator and discriminator are trained simultaneously. The generator is updated to produce better samples by using the gradients from the discriminator's loss, and the discriminator is updated to better distinguish real and fake samples by using the gradients from its own loss.

Once the GAN is trained, the generator can be used to produce new samples that are similar to the training data. These samples can be used for a variety of tasks such as image generation, data augmentation, and text generation.

In the context of Unsupervised Domain Adaptation (UDA), GANs can be used to align the feature distributions of the source and target domains. The generator is the feature extractor and the discriminator is the domain classifier, the generator is trained to produce features that are indistinguishable between the source and target domains, so that the domain classifier cannot accurately predict the domain labels. This method is known as Domain Adversarial Training (DAT)

It's worth noting that, GANs are difficult to train and require a lot of computational resources and it's easy to fall into mode collapse or other issues. Also the generator may not generate samples that are perfect copies of the real data, but instead samples that are similar in distribution to the real data.

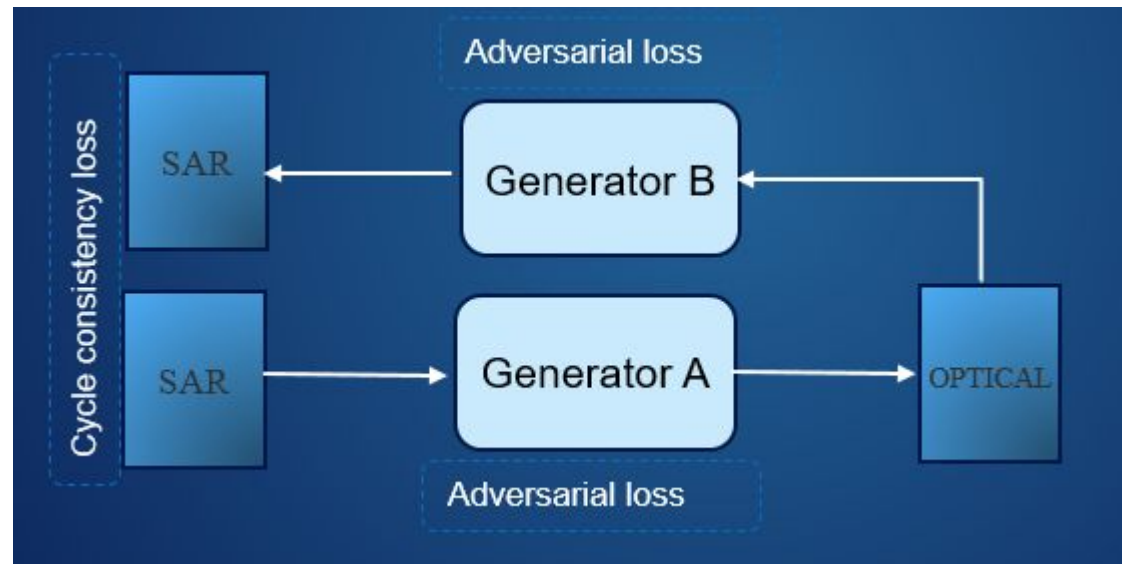
GAN-Based Synthesis of Hyperspectral Data



Generative Adversarial Network Synthesis of Hyperspectral Vegetation Data

GAN: Learning from Unlabeled Paired Dataset

- Cycle GAN: using unlabeled SAR and optical dataset



<https://ieeexplore.ieee.org/abstract/document/9120230>

CycleGAN is a type of Generative Adversarial Network (GAN) designed for image-to-image translation tasks. It is a deep learning model that can transform images from one domain to another, without the need for paired data.

CycleGAN consists of two generators and two discriminators. The generators are responsible for transforming the images from one domain to another, while the discriminators are responsible for determining whether the transformed images are realistic or fake. The generators and discriminators compete with each other in a zero-sum game, where the generators try to fool the discriminators, and the discriminators try to correctly identify the fake images.

One of the unique features of CycleGAN is its use of cycle consistency loss, which ensures that the transformation is bijective, meaning that the transformed image can be transformed back to its original domain. This helps to ensure that the transformed images are not just randomly generated, but are instead meaningful and preserve the structure and content of the original images.


CycleGAN has been used for a variety of image-to-image translation tasks, such as style transfer, converting photographs to paintings, and transforming satellite images to maps, among others. It is a powerful tool for bridging the gap between different domains and transforming images in a semantically meaningful way.

In summary, CycleGAN is a type of Generative Adversarial Network designed for image-to-image translation tasks, that consists of two generators and two discriminators, and uses cycle consistency loss to ensure that the transformed images are bijective and preserve the structure and content of the original images.

Unsupervised Learning

- *Unsupervised learning* – learning with **unlabeled data**

- Deep Clustering
- Approximating distribution (Generative Adversarial Network)
- Learning based on dimensionality reduction (autoencoder)



Let's
skip this

An autoencoder is a type of neural network architecture that is used for unsupervised learning. The main goal of an autoencoder is to learn a compact representation, or bottleneck, of the input data, known as the encoded representation, and then use this representation to reconstruct the original data, with minimal loss.

An autoencoder typically consists of two main components: an encoder and a decoder. The encoder is a neural network that takes the input data and maps it to a lower-dimensional encoded representation. The decoder is another neural network that takes the encoded representation and maps it back to the original data, or a reconstruction of the original data.

The encoder and decoder are trained together in an unsupervised way, using a loss function that compares the original data to the reconstructed data. The idea is that the encoder learns to extract the most important features of the data and discard the noise, and the decoder learns to use those features to reconstruct the original data.

Autoencoder can be used for a variety of tasks such as image compression, anomaly detection, and feature extraction. Autoencoder can be used as pre-training for supervised tasks, by initializing the encoder's weights with the trained weights from the autoencoder.

It's worth noting that autoencoder can have different architectures such as Convolutional Autoencoder, Variational Autoencoder, and Denoising Autoencoder, depending on the problem and data set, different architectures can be used.

Different Learning Paradigms

- **Supervised learning** – learning with **labeled data**
 - Approach: collect a large dataset, manually label the data, train a model, deploy
 - Learned **feature representations** on large datasets can be transferred via pre-trained models to smaller domain-specific datasets
- **Unsupervised learning** – learning with **unlabeled data**
 - Approach: discover patterns in data either via clustering similar instances, or density estimation, or dimensionality reduction ...
- **Self-supervised learning** – representation learning with **unlabeled data**
 - Learn useful **feature representations** from unlabeled data through **pretext tasks**
 - The term “self-supervised” refers to creating **its own supervision** (i.e., without supervision, without labels)
 - Self-supervised learning is one category of unsupervised learning