AI4EO

Special Topics in Applications (AIL861)
Artificial Intelligence for Earth Observation
Lecture 5

Instructor: Sudipan Saha

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

Self-Supervised Learning

- Supervised learning tasks have pre-defined (and generally human-provided) labels.

- Unsupervised learning has just the data samples without any supervision, label or correct output.

- Self-supervised learning derives its labels from a co-occurring modality for the given data sample or from a co-occurring part of the data sample itself.

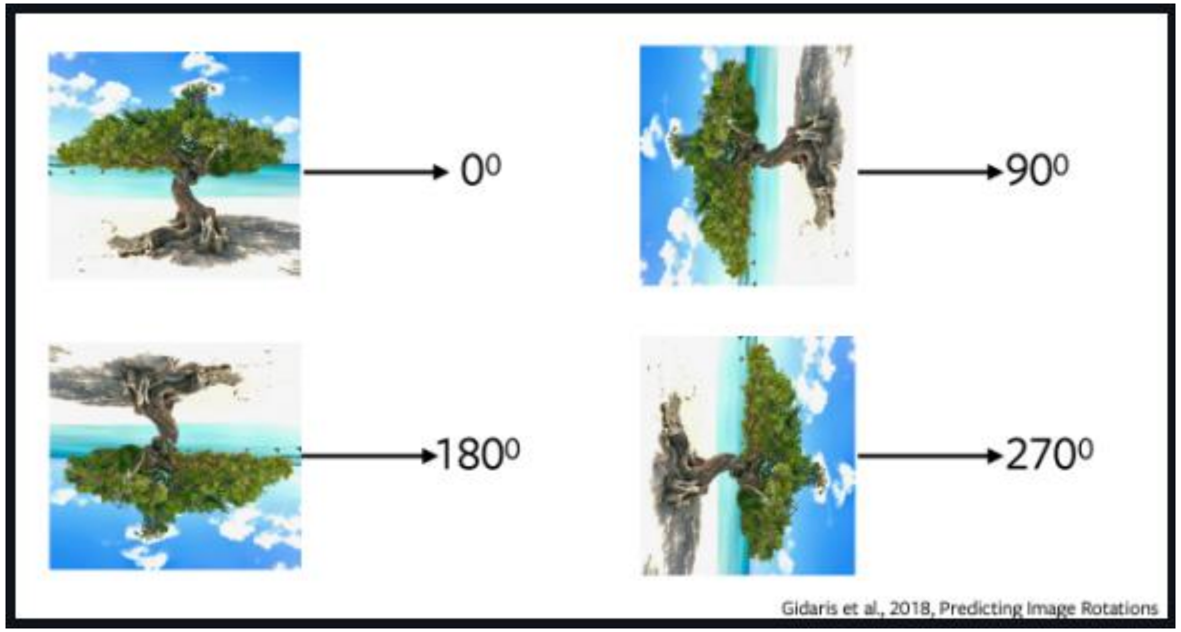
Self-Supervised Learning: Motivation

- Enables learning representations of data by just observations of how different parts of the data interact.
- Enables to leverage multiple modalities that might be associated with a single data sample.
- Drops the requirement of huge amount of annotated data.

Pre-Text Task

- The pretext task is the self-supervised learning task solved to learn visual representations.

- Rotation of images



In self-supervised learning, a pretext task is a task that is used to train a model on a dataset without any explicit supervision. The idea is that by solving the pretext task, the model will learn useful representations of the data that can be used for other downstream tasks.

For example, one common pretext task is to use the model to predict the color of a patch of an image, given the color of a surrounding patch. By training the model on this task, the model will learn to recognize features in the image that are useful for understanding the relationships between different parts of the image. These features can then be used for other tasks, such as object recognition or image classification.

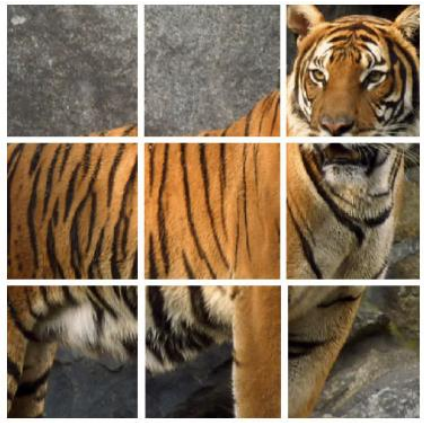
Another common pretext task is to use the model to predict the rotation of an image, given the original image. By training the model to predict the rotation, the model will learn to recognize features in the image that are invariant to rotation, such as object parts, edges etc.

Pretext tasks are useful for self-supervised learning because they allow the model to learn useful representations of the data without the need for manual annotation. This can be especially useful when labeled data is scarce or expensive to acquire.

It's important to note that the choice of pretext task is crucial for self-supervised learning. The pretext task should be designed to be sufficiently challenging and to encourage the model to learn useful representations of the data. It should also be chosen such that the learned representations are transferable to the downstream task.

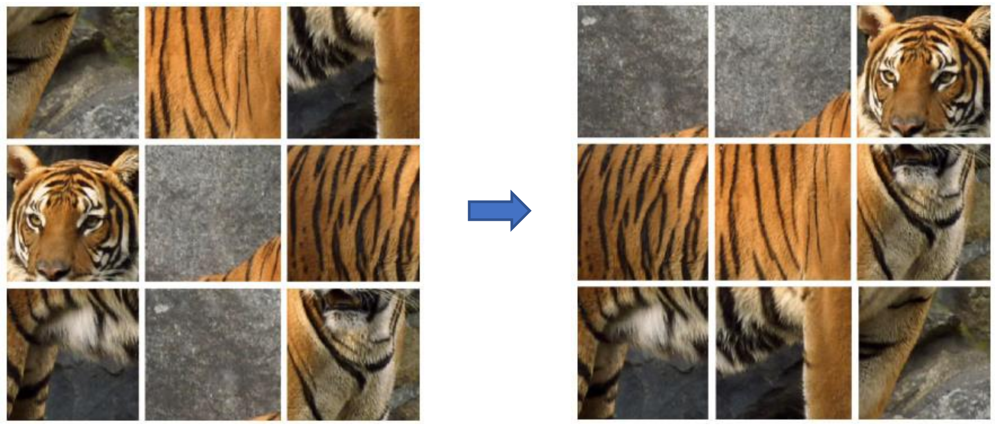
Pre-Text Task

- Relative position of image patch



Pre-Text Task

Image jigsaw puzzle



Pre-Text Task: Rotation / Jigsaw / …

- Rarely used in Earth observation

- Such spatial correlation is less dominant in EO images
  .

Pretext tasks are commonly used in various computer vision tasks such as image classification and object detection. However, it's not used in Earth observation as frequently as in other domains.

Earth observation data is often high-dimensional, multi-modal, and multi-temporal, which makes it challenging to design pretext tasks that are suitable for the data. Additionally, many earth observation tasks require the use of additional information, such as topography, land cover, or meteorological data, which can make it difficult to design pretext tasks that are independent of these additional data sources.

Pre-Text Task: Geolocation Classification

- Geolocation metadata is often available

- Cluster the dataset according to lat/long

  .

- Train a model to predict these clusters

Geography-Aware Self-Supervised Learning, 2021

Pre-Text Task: Predict Bands

- Drop a band from the image (or alternatively create a synthetic band like NDVI)

- Use rest of bands to predict the missing band

  .

Pre-Text Task: Image Inpainting

- Mask and restore local salient regions

- Originally proposed in Computer Vision, however also applicable to Earth observation

Semantic Segmentation of Remote Sensing Images With Self-Supervised Semantic-Aware Inpainting

.

Image inpainting is a computer vision task in which the goal is to fill in missing or corrupted parts of an image with plausible content. This can be done by analyzing the surrounding pixels and using that information to generate new pixels that are consistent with the rest of the image.

There are several techniques that can be used for image inpainting, such as exemplar-based inpainting, texture synthesis, and deep learning-based inpainting.

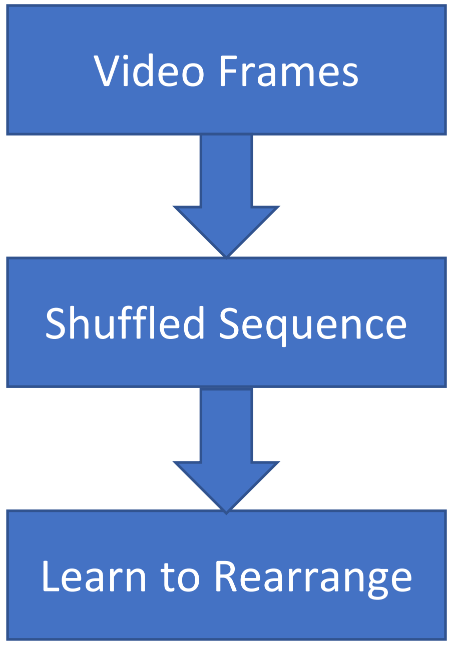
Exemplar-based inpainting is a technique in which the algorithm searches for similar patches in the surrounding area of the image and copies them to fill in the missing pixels.

Texture synthesis is a technique that involves synthesizing new texture to fill in the missing area, by analyzing the patterns and textures in the surrounding pixels.

Deep learning-based inpainting is a technique that uses deep neural networks to generate new pixels. The neural network is trained on a dataset of images with missing regions, and it learns to predict the missing pixels based on the surrounding context.

Image inpainting has many practical applications, such as image editing, image restoration, and image completion. It can also be used as a pretext task for self-supervised learning, where the goal is to learn features that are useful for other

Pre-Text Task for Video



Time-Series Analysis by Learning to Reorder

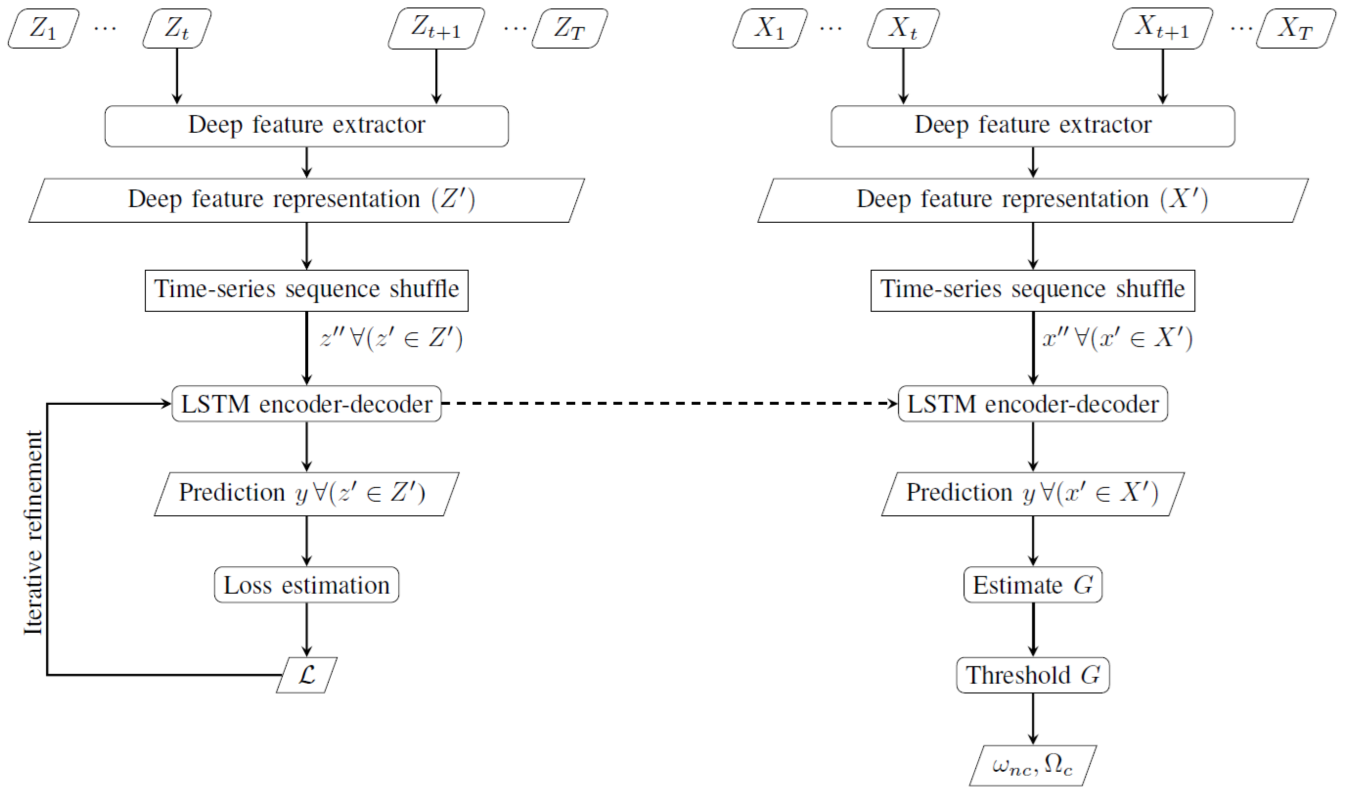
✓ Learn in unsupervised way from time-series

✓ Jumble the temporal data in random order and then learn to rearrange

Unchanged

Sequence

(training)



Z_1 ⋯ Z_t
Z_{t+1} ⋯ Z_T
X_1 ⋯ X_t
X_{t+1} ⋯ X_T
Deep feature extractor
Deep feature extractor
Deep feature representation (Z')
Deep feature representation (X')
Time-series sequence shuffle
Time-series sequence shuffle
$z'' \forall (z' \in Z')$
$x'' \forall (x' \in X')$
LSTM encoder-decoder
LSTM encoder-decoder
Iterative refinement
Prediction $y \forall (z' \in Z')$
Prediction $y \forall (x' \in X')$
Loss estimation
Estimate G
$\mathcal{L}$
Threshold G
ω_{nc}, Ω_c

Time-series analysis by learning to reorder is a machine learning technique that aims to analyze time-series data by reordering the observations in a way that makes it easier to extract useful information.

The basic idea is to train a model to predict the order of a set of time-series observations, such that the reordered observations are more informative or easier to analyze. The model can be trained using a supervised or unsupervised learning algorithm, depending on the task.

One example of a time-series analysis task that can benefit from reordering is anomaly detection. If a time-series data has a temporal structure such as seasonality or trend, it can be hard to detect anomalies by simply comparing each observation to a threshold. By learning to reorder the observations, the model can learn to identify patterns or features that are indicative of normal behavior and make it easier to detect anomalies.

Another example is forecasting, where the goal is to predict future values of a time-series. In this case, the reordering can be used to align similar observations together and make it easier for the model to learn the underlying patterns or trends in the data.

In addition, reordering can be used as a pretext task for self-supervised learning, where the goal is to learn features that are useful for other time-series analysis tasks such as classification, or clustering.

It's important to note that, as with other pretext tasks, the choice of reordering will depend on the specific characteristics of the time-series data and the desired downstream task.

Contrastive Self-Supervised Learning

- The goal of contrastive representation learning is to learn such an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart.

- Contrastive learning can be applied in both supervised and unsupervised setting.

Contrastive self-supervised learning is a method of self-supervised learning that aims to learn representations of data by comparing different examples. The basic idea is to train a model to distinguish between different augmentations of the same data, rather than providing explicit labels.

In contrastive self-supervised learning, the model is trained to predict a similarity score between a pair of data points, where one is the original data point and the other is a transformed version of the original data point. The goal is to learn a representation of the data that is robust to different transformations, such as rotation, translation, or scaling.

One popular approach to contrastive self-supervised learning is the use of a contrastive loss function, where the model is trained to maximize the similarity score for positive pairs (original and transformed versions of the same data) and minimize the similarity score for negative pairs (original and transformed versions of different data).

Contrastive self-supervised learning has been shown to be effective in learning representations of images, videos, and text data, and has been used to improve the performance of various computer vision, natural language processing and speech recognition tasks.

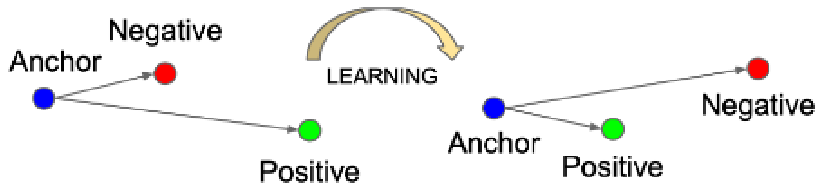
It's worth noting that Contrastive Learning is a kind of self-supervised learning, but it's not the only one, other methods exist like Auto-encoders, Generative models, etc.

In addition, there are also different variants of contrastive learning, such as SimCLR, MoCo, BYOL, etc. Each of these variants has its own strengths and weaknesses and is better suited for certain types of data and tasks. Therefore, it's important to choose the right variant based on the characteristics of the data and the desired downstream task.

Contrastive Losses

- Contrastive loss first introduced in 2005, in context of metric learning (Learning a Similarity Metric Discriminatively, with Application to Face Verification)

- Triplet loss, proposed in 2015 (FaceNet: A Unified Embedding for Face Recognition and Clustering)



Some sources are from https://lilianweng.github.io/posts/2021-05-31-contrastive/

Contrastive Losses

- InfoNCE loss proposed in the context of contrastive predictive coding

As defined in the paper of MoCo (not the original definition)

For a query q, we have a set of keys $\{k_1, k_2, \ldots, k_N\}$, where among those N keys, there is one positive key k_+ , InfoNCE is defined as:

$$L_q = -\log\left(\frac{\exp\left(\frac{qk_+}{\tau}\right)}{\sum_i \exp(qk_i/\tau)}\right)$$

How to Generate Positive and Negative Pairs

- How to generate positive and negative pairs in unsupervised manner?

“Geographic” concept

- Geographically closer objects are likely to share the label.

- Geographically farther objects are likely to be different.

Tile2Vec

Tile2Vec: Unsupervised representation learning for spatially distributed data (a 2018 paper) observes that “*This is akin to the distributional hypothesis used to construct word vector representations in natural language: words that appear in similar contexts should have similar meanings.*”

Tile2Vec is a self-supervised learning method for geospatial data, specifically for satellite imagery. The method is based on the idea of using convolutional neural networks (CNNs) to extract features from image tiles, and then training a model to predict the context of the image tiles.

The process starts by dividing the satellite image into small non-overlapping tiles, and then feeding these tiles through a pre-trained CNN to extract features. The extracted features are then used as input to a second model, which is trained to predict the context of the tile, such as the land cover or land use class.

Tile2Vec can be used to train models for a wide range of tasks, such as land cover classification, object detection, and image retrieval. Since the Tile2Vec is self-supervised, it does not require any labeled data, which makes it an efficient and cost-effective solution for training models for geospatial data.

Tile2Vec is also related to the pretext task in self-supervised learning, where a model is first trained on a task that is easy to solve but still requires the model to learn useful features, and then the model is fine-tuned on a more complex task, such as classification. In the case of Tile2Vec, the pretext task is predicting the context of an image tile, which is easy to solve and still requires the model to learn useful features for other geospatial tasks.

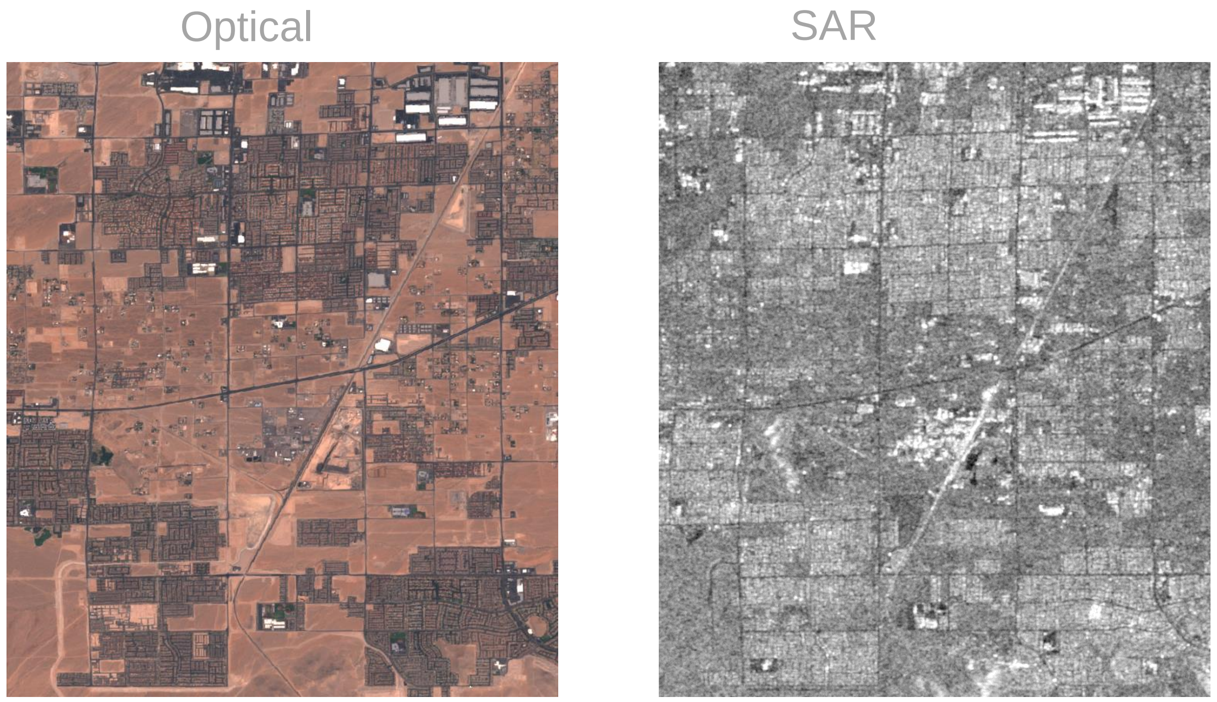
Multi-Sensor Optical-SAR Change Detection

✓ Propose an unsupervised multi-sensor change detection method:

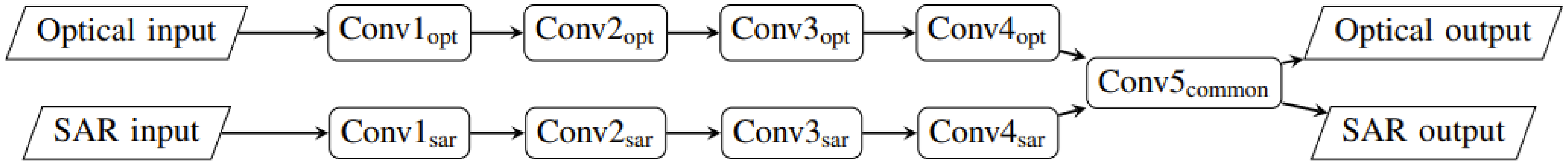
❑ Pre-change image: Optical

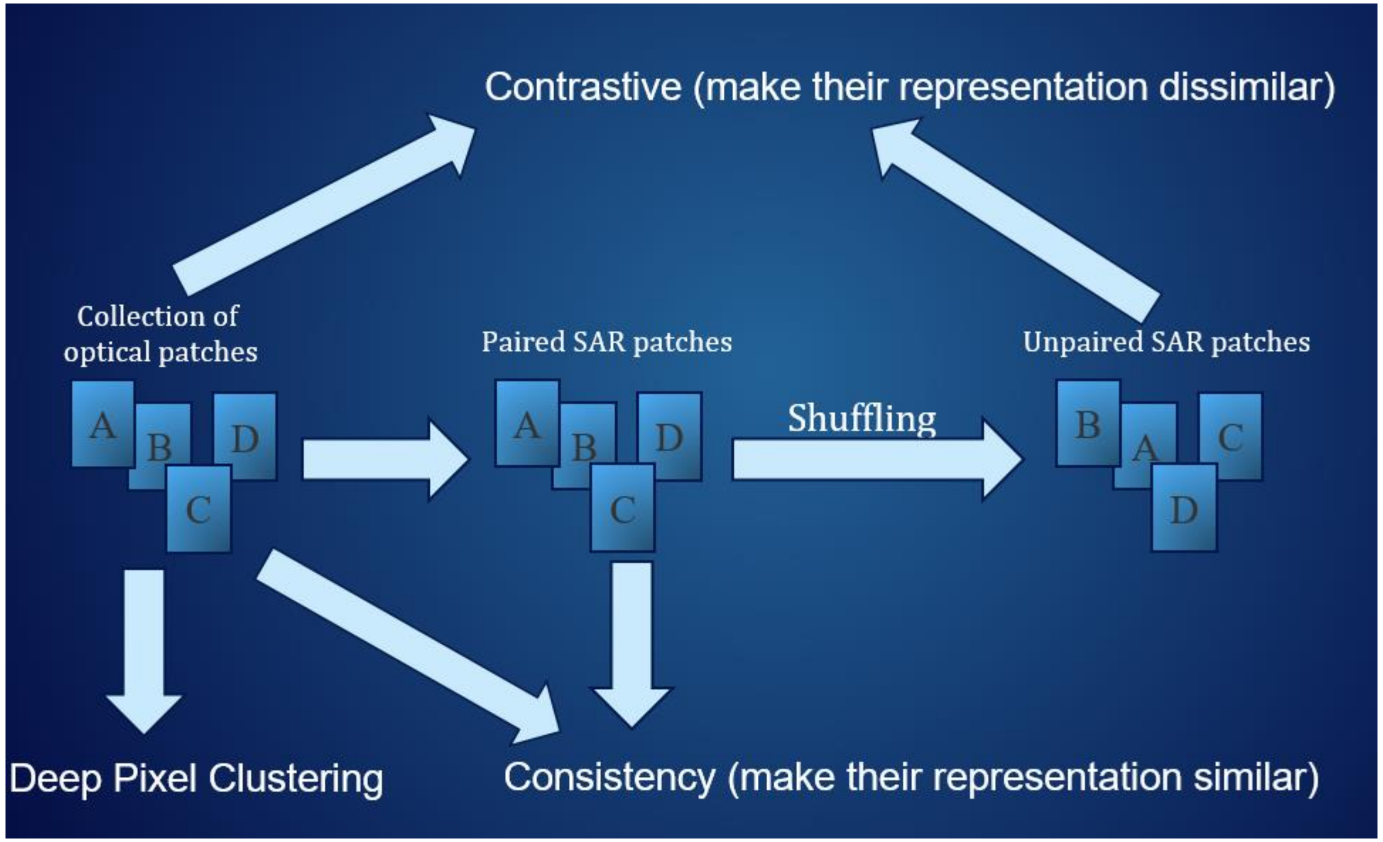
❑ Post-change image: Synthetic Aperture Radar (SAR)

✓ Project optical and SAR scenes to a common feature domain, so that they become comparable.



Contrastive Multi-Sensor Change Detection





Contrastive (make their representation dissimilar)
Collection of optical patches
A
B
D
C
Paired SAR patches
A
B
D
C
Shuffling
Unpaired SAR patches
B
A
C
D
Deep Pixel Clustering
Consistency (make their representation similar)

How to Generate Positive and Negative Pairs

- How to generate positive and negative pairs in unsupervised manner?

“Temporal” concept

- For a given location, successive images can be treated as positive pairs.

Geography-Aware Self-Supervised Learning, 2021

Negative Samples

- Challenging to obtain
- Geographically “farther-closer” does not always ensure that we indeed got negative samples
- Quality of negative samples may hinder contrastive learning

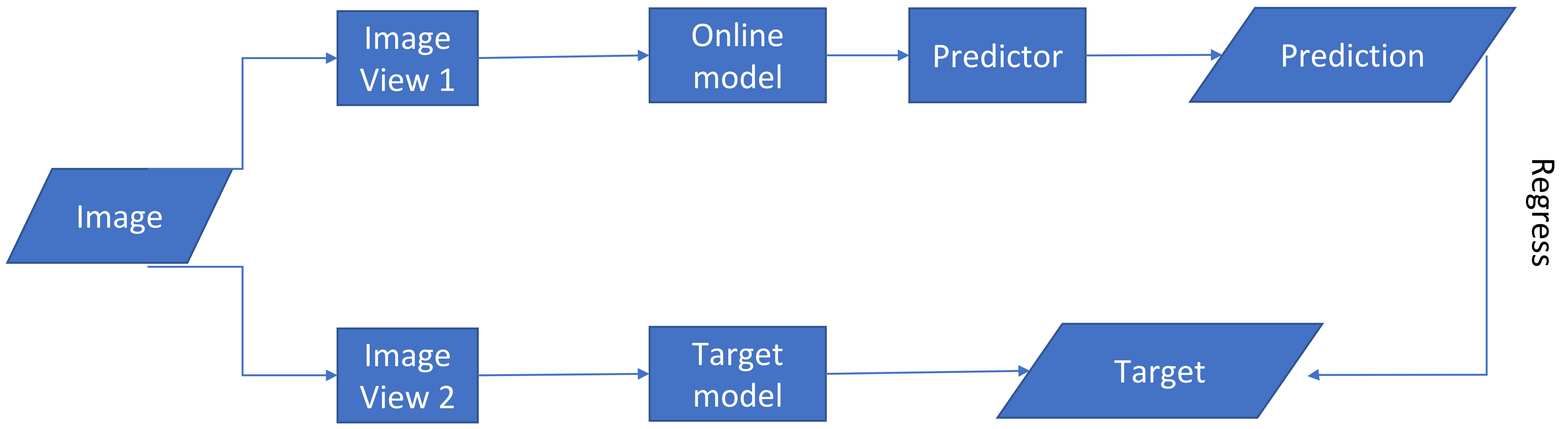
Bootstrap Your Own Latent

- No pseudo-label
- Unlike previous contrastive methods, no negative sample
    - ❑ Negative samples are expensive
- Training procedure is simple

BYOL

- Image: generate two views
- Two different networks: online network and target network
- One input is processed through the online network and the other through the target network

BYOL: Mechanism



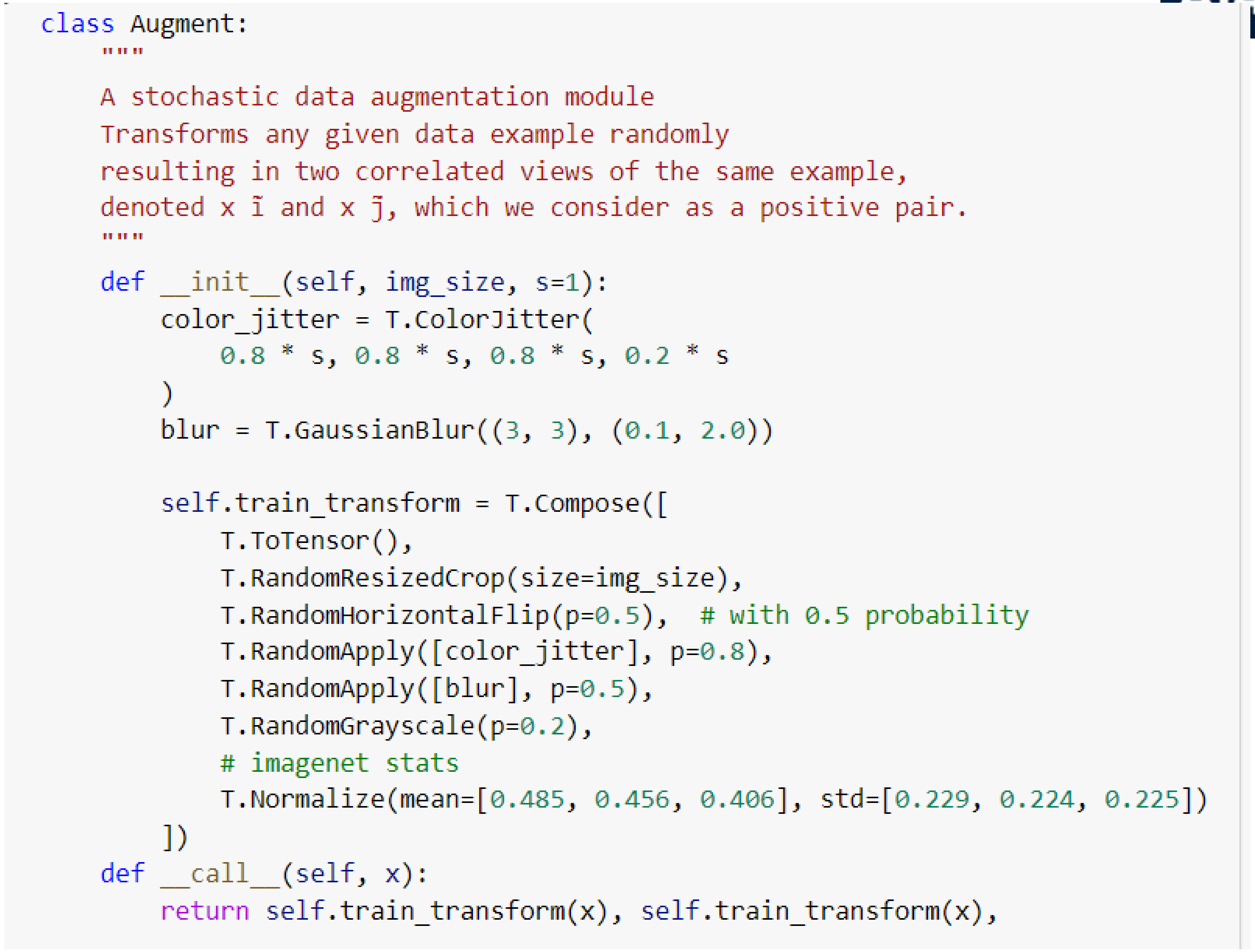
in the BYOL method there are two neural networks: the online network and the target network. The online network is the one that is trained during the training process, while the target network is kept fixed.

The online network is used to generate the internal representations of the input data, and the target network is used to predict those representations. The goal is for the online network to learn to produce representations that the target network can easily predict, which will lead to the online network learning useful features for the input data.

The online network and the target network are usually implemented as identical neural networks, with the same architecture and initial weights. The online network is updated during the training process, while the target network is kept fixed. This approach is called "mean teacher"

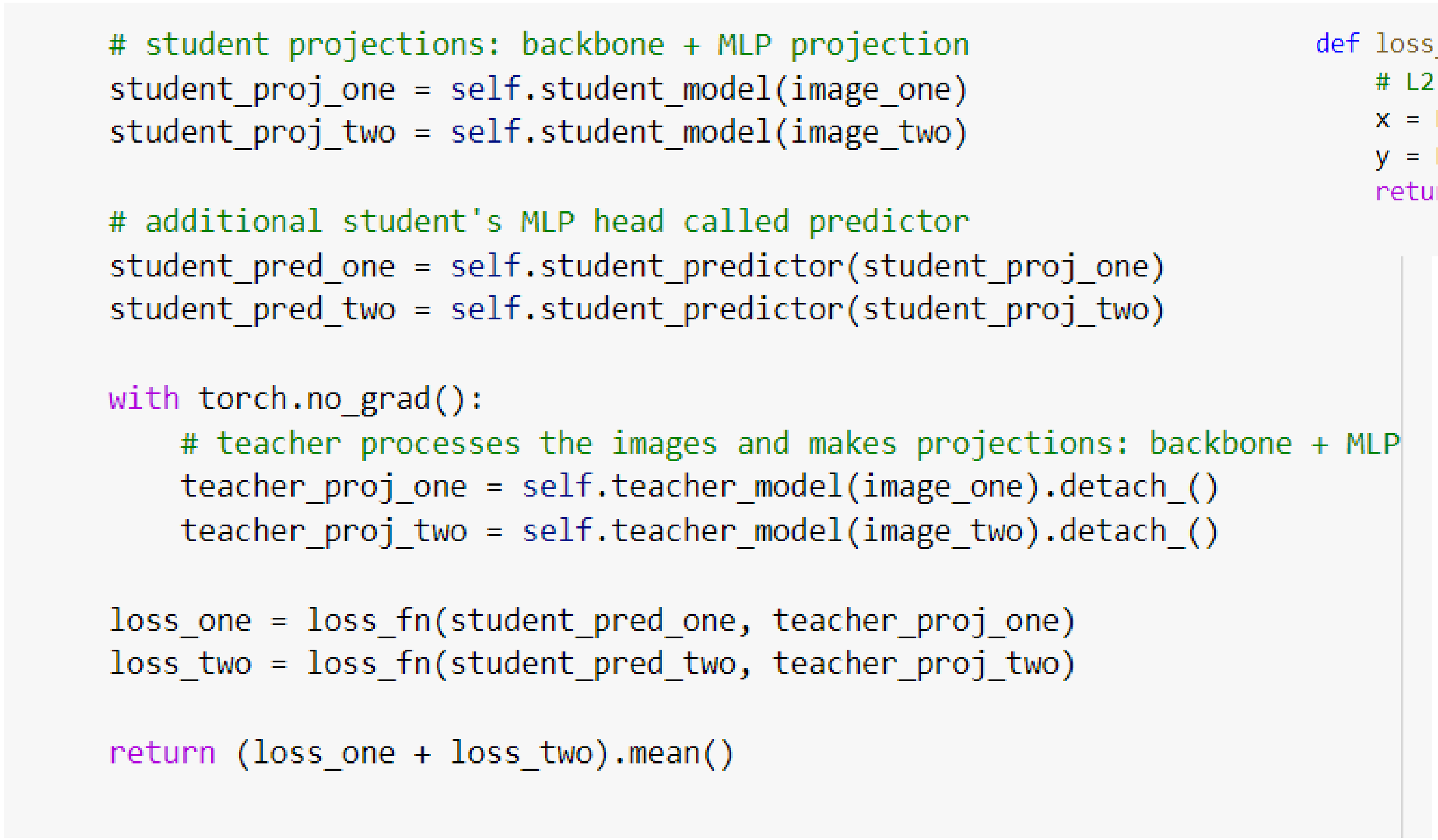
The online and target network are used in an iterative process, where the online network generates representations of the input data, and the target network is trained to predict those representations. This process is repeated for multiple iterations, and the goal is for the online network to learn to produce representations that the target network can easily predict, which will lead to the online network learning useful features for the input data.

Ack:
https://theaisummer.com/byol/



```
class Augment:
    """
    A stochastic data augmentation module
    Transforms any given data example randomly
    resulting in two correlated views of the same example,
    denoted x ĩ and x j̃, which we consider as a positive pair.
    """
    def __init__(self, img_size, s=1):
        color_jitter = T.ColorJitter(
            0.8 * s, 0.8 * s, 0.8 * s, 0.2 * s
        )
        blur = T.GaussianBlur((3, 3), (0.1, 2.0))

        self.train_transform = T.Compose([
            T.ToTensor(),
            T.RandomResizedCrop(size=img_size),
            T.RandomHorizontalFlip(p=0.5),  # with 0.5 probability
            T.RandomApply([color_jitter], p=0.8),
            T.RandomApply([blur], p=0.5),
            T.RandomGrayscale(p=0.2),
            # imagenet stats
            T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])
    def __call__(self, x):
        return self.train_transform(x), self.train_transform(x),
```



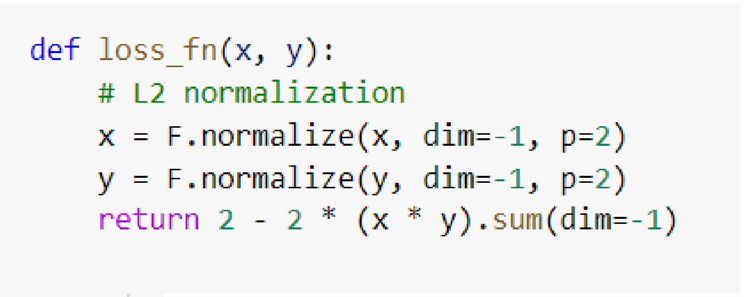
```
# student projections: backbone + MLP projection
student_proj_one = self.student_model(image_one)
student_proj_two = self.student_model(image_two)

# additional student's MLP head called predictor
student_pred_one = self.student_predictor(student_proj_one)
student_pred_two = self.student_predictor(student_proj_two)

with torch.no_grad():
    # teacher processes the images and makes projections: backbone + MLP
    teacher_proj_one = self.teacher_model(image_one).detach_()
    teacher_proj_two = self.teacher_model(image_two).detach_()

loss_one = loss_fn(student_pred_one, teacher_proj_one)
loss_two = loss_fn(student_pred_two, teacher_proj_two)

return (loss_one + loss_two).mean()
```



```
def loss_fn(x, y):
    # L2 normalization
    x = F.normalize(x, dim=-1, p=2)
    y = F.normalize(y, dim=-1, p=2)
    return 2 - 2 * (x * y).sum(dim=-1)
```

```
class EMA():
    def __init__(self, alpha):
        super().__init__()
        self.alpha = alpha

    def update_average(self, old, new):
        if old is None:
            return new
        return old * self.alpha + (1 - self.alpha) * new
```

Self-Supervised Learning Evaluation Strategies

- Linear classification
- Data efficiency (e.g., fine-tune with 1% of actual training dataset)
- Transfer learning

AI4EO

MTP/MSR Topic

Topic 1: Self-supervised learning for Earth observation images

Topic 2: Few shot filtering

Topic 3: Combining satellite images and meteorological data to monitor and model extreme weather events