Semi-supervised learning (SSL)

for remote sensing (SSL-for-RS)

About me

- Itzá Hernández
- PhD. Computer Science - Computer Vision
- University Jaume I, Castellón, Spain
- Internship Helmholtz Institute Freiberg for Resource Technology (HIF)
- @itzahs

Post-it

- **Name**: Itzá
- **First Job**: Institute for Tropical Agriculture
- **What I learned from it**: If you plant coffee and cacao together, your coffee beans will have a pinch of cacao flavor.

Learning outcomes of this workshop

1. Get data

Create a class dataset with a transformation pipeline

2. Clone repositories

Use software built by others and adapt it

3. Train model

Train SSL model from a Google Colab environment

4. Evaluate the model

Classification accuracy with confusion matrix.
Embeddings visualization with t-SNE.

Content of this workshop

01 Introduction to DSSL
16:00 - 16:15 (15 min)

02 Data & Code
16:15 - 16:30 (15 min)

03 Training model
16:30 - 17:00 (30 min)

04 Evaluate model
17:00 - 17:30 (30 min)

Questions welcome anytime

Total estimated time:
≈ 2 hours

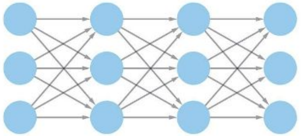
01

Introduction

Deep Semi Supervised Learning
16:00 - 16:15 (15 min)

Machine Learning & Deep Learning

ML

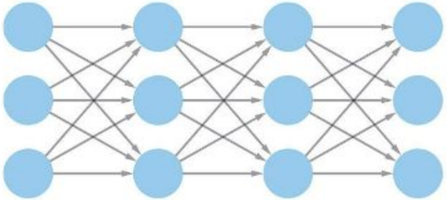


Feature extraction

Classification

Class probability

DL



Feature extraction + Classification

Class probability

Supervised

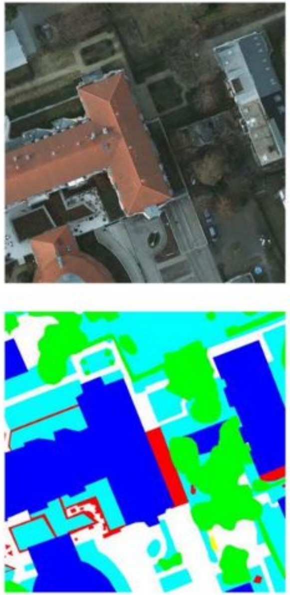
Semi-supervised

Unsupervised

[1] Bhavsar, K. (2018, September 14). Understanding your Convolution Network with Visualizations. Towards Data Science. https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b

DL in Aerial Image Classification

Pixel-level



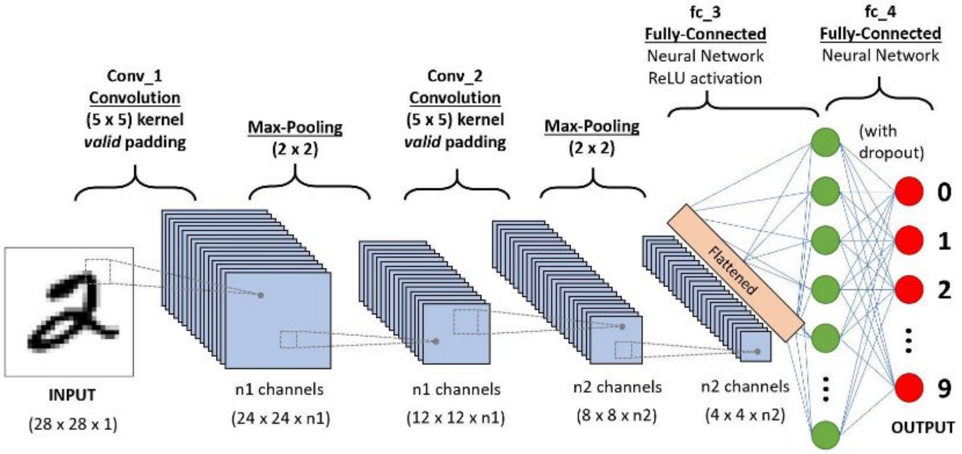
Object-level

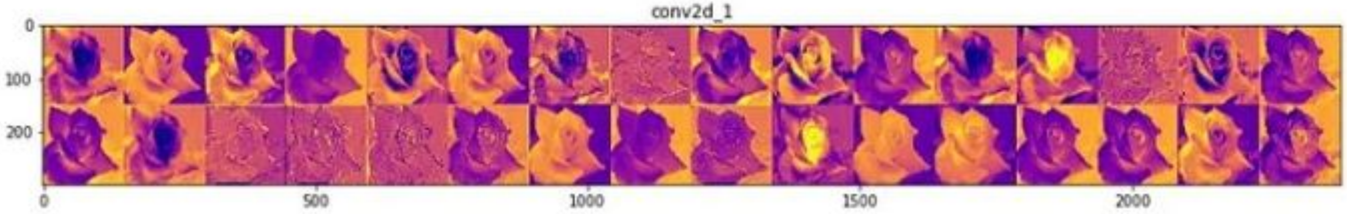
Scene-level

[2] G. Cheng, X. Xie, J. Han, L. Guo and G. -S. Xia, "Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 3735-3756, 2020, doi: 10.1109/JSTARS.2020.3005403 .

Convolutional Neural Network (CNN)

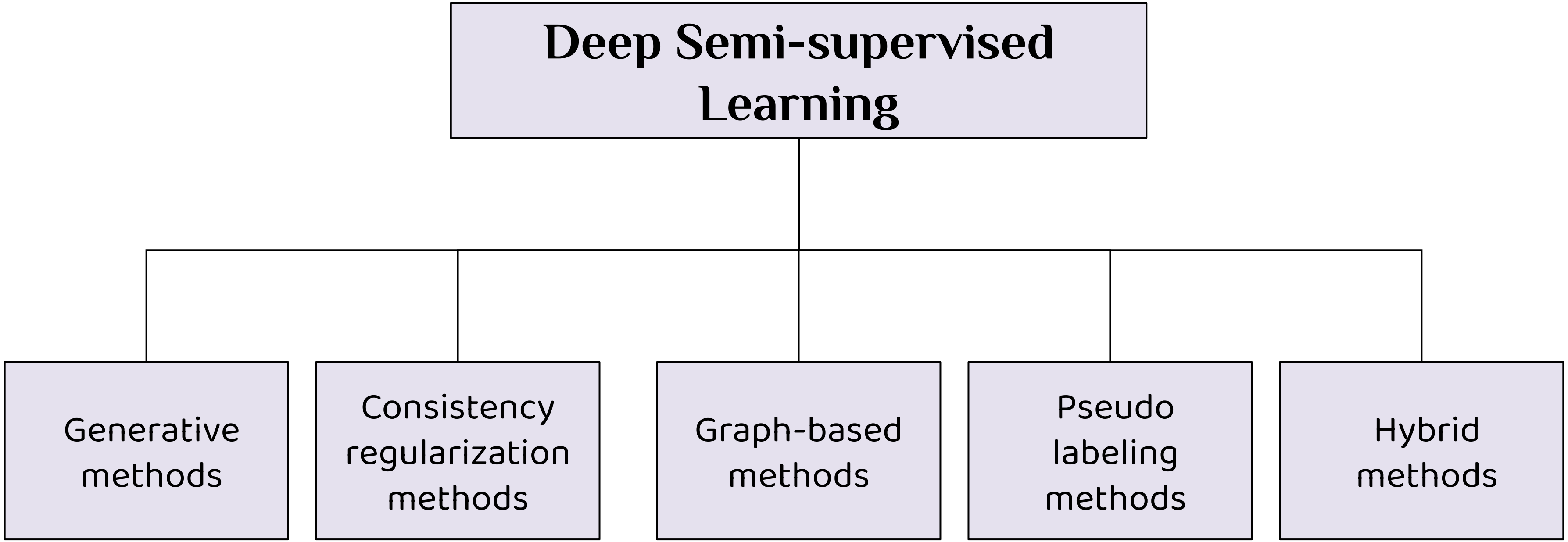
Original Image





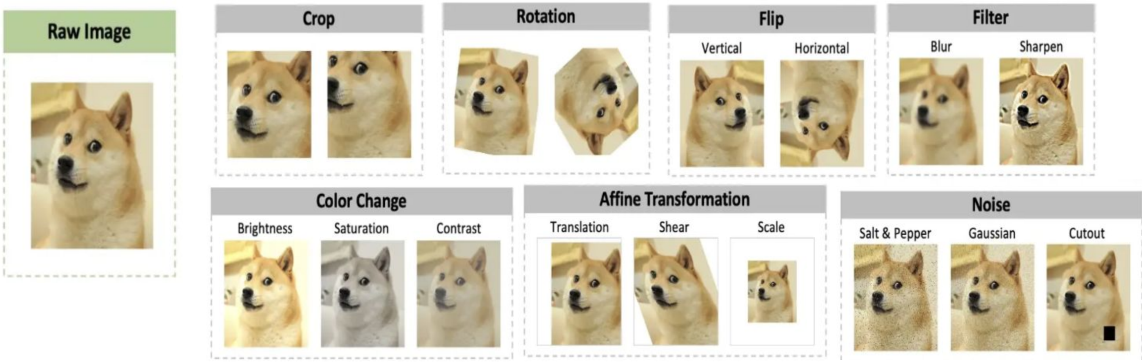
[1] Bhavsar, K. (2018, September 14). Understanding your Convolution Network with Visualizations. Towards Data Science. https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b

Taxonomy



[3] X. Yang, Z. Song, I. King and Z. Xu, "A Survey on Deep Semi-Supervised Learning," in IEEE Transactions on Knowledge and Data Engineering, 2022, doi: 10.1109/TKDE.2022.3220219.

Image data augmentation for DL



- **Data Warping**: transforms existing images such as the label is preserved.

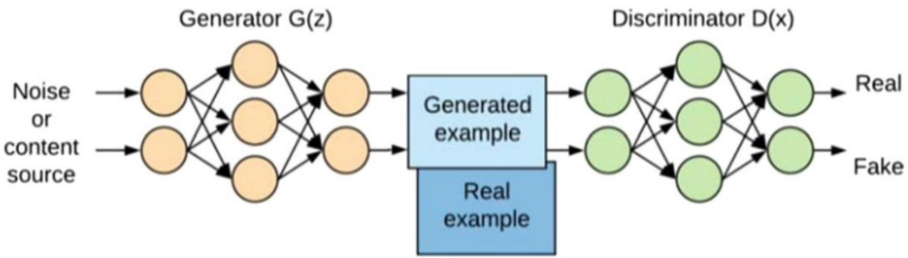
- **Oversampling**: creates synthetic instances and add them to the training set.

Synthetic data - CycleGANS - emotion classification

[4] Sharma, A. (2019, June 12). Complete Guide to Data Augmentation for Computer Vision. Towards Data Science. https://towardsdatascience.com/complete-guide-to-data-augmentation-for-computer-vision-1abe4063ad07.

[5] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J Big Data* 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

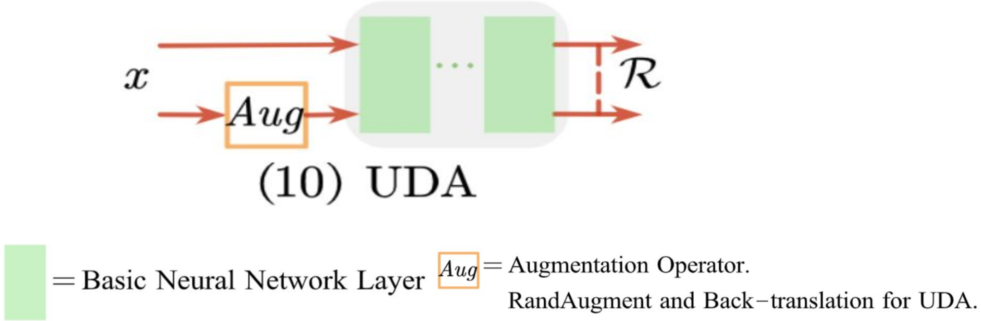
Generative methods



Generative modeling refers to the practice of **creating artificial instances** from a dataset such that they retain similar characteristics to the original dataset.

e.g. Generative Adversarial Networks (**GAN**) and Variational AutoEncoder (**VAE**).

Consistency regularization



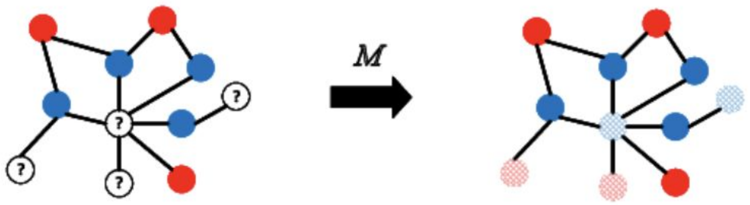
A consistency regularization term is applied to the **final loss function**. e.g. **Cross Entropy for example H(F(x), Tx).**

A realistic perturbation in the training data should not change the output of the model.

e.g. Unsupervised Data Augmentation (**UDA**).

[3] X. Yang, Z. Song, I. King and Z. Xu, "A Survey on Deep Semi-Supervised Learning," in IEEE Transactions on Knowledge and Data Engineering, 2022, doi: 10.1109/TKDE.2022.3220219.

Graph-based

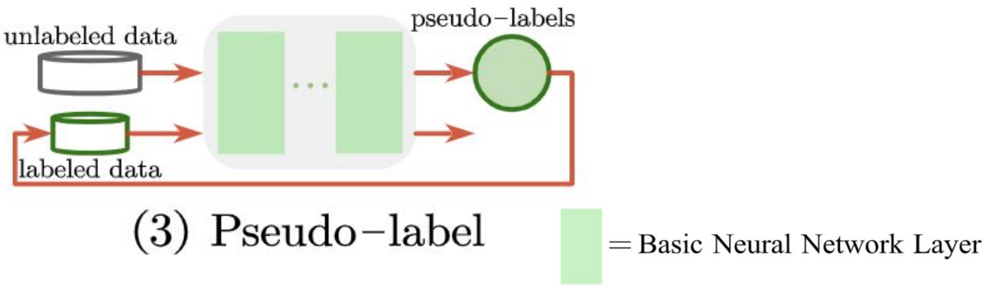


The nodes/vertices are **representations of the training samples** and the edges encode the relationships between the nodes.

The goal is to encode the nodes as small-scale vectors at first and then how each node belongs within the context in the graph.

Deep embedding methods are **AutoEncoders** and Graph Neural Networks (GNN).

Pseudo-labelling

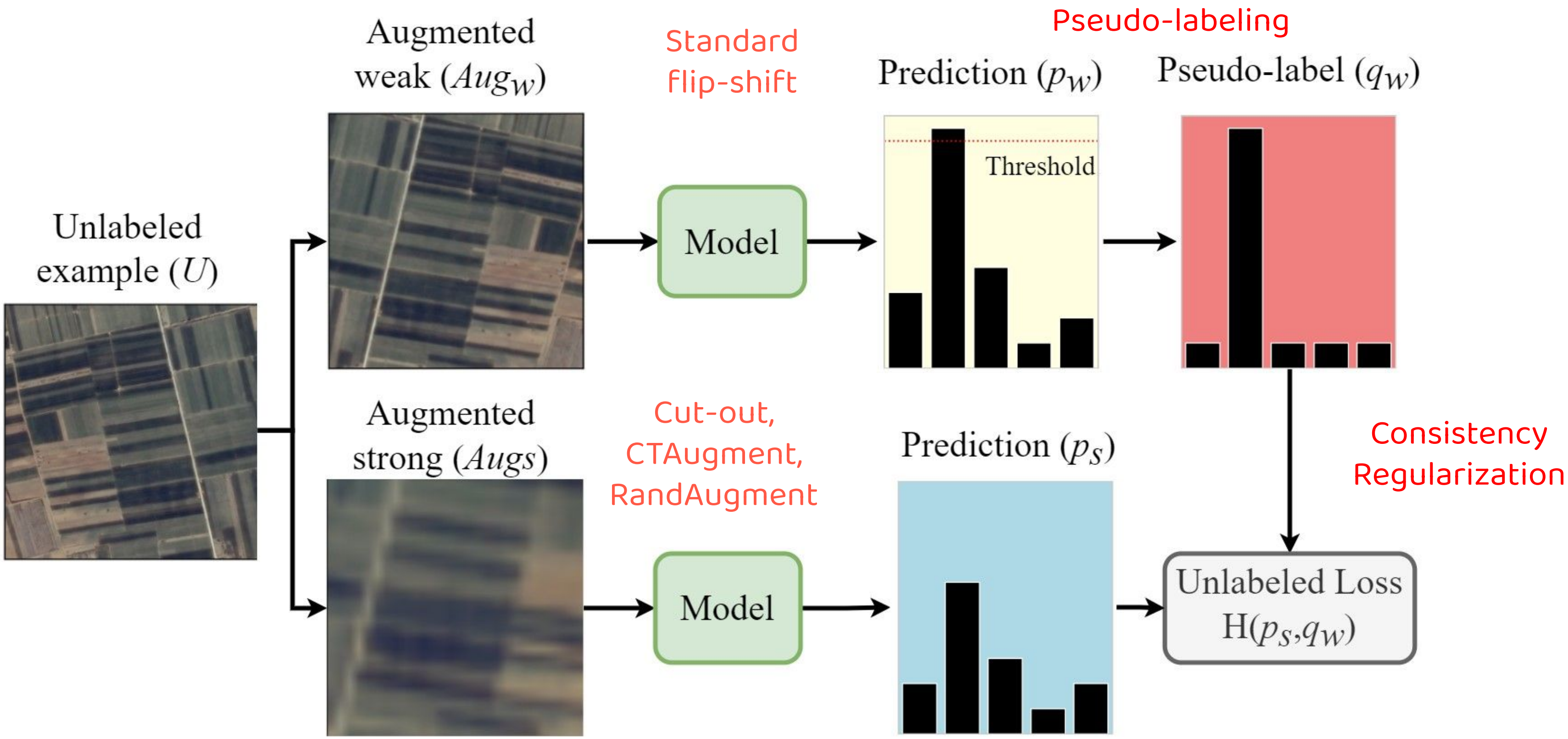


Pseudo-labeling methods rely on the **high-confidence of pseudo-labels**, which can be added to the training data set as labeled data.

Self-training: leverage model's own confident predictions to produce the pseudo-labels for **unlabeled data**.

[3] X. Yang, Z. Song, I. King and Z. Xu, "A Survey on Deep Semi-Supervised Learning," in IEEE Transactions on Knowledge and Data Engineering, 2022, doi: 10.1109/TKDE.2022.3220219.

Hybrid: Fixmatch - Consistency Regularization and Pseudo Labeling



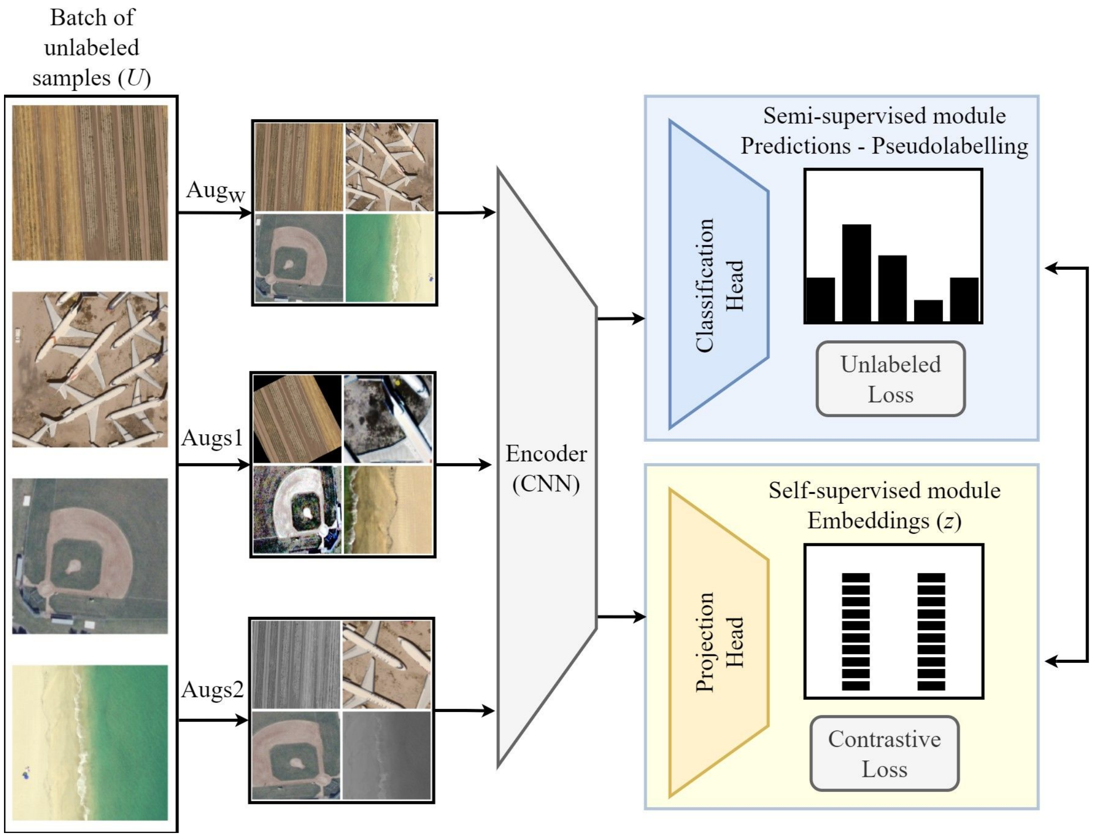
[6] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. NeurIPS, 33, 2020.

Hybrid: CoMatch - Consistency Regularization and Pseudo Labeling

Standard flip-shift

Cut-out, CTAugment, RandAugment

Color Jittering Grayscaling



Semi-supervised

Self-supervised

[6] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. NeurIPS, 33, 2020.

02
Data & Code

Cloning the GitHub repository and downloading the dataset
16:15 - 16:30 (15 min)

1 or 2

- Do you work 1- with satellite imagery or 2- other types of data.
- When using Deep Learning do you prefer working with 1-Pytorch or 2-Tensorflow.
- For working you use 1-Cloud services or 2-Local servers/Personal laptop.

Codes for this workshop

@itzahs SSL-for-RS

SSL-for-RS Public

Configuration files for the ICIAP2023 paper "Semi-supervised classification for remote sensing datasets" & step-by-step implementation for the Geomundus2023 Workshop.

Jupyter Notebook

Step-by-step implementation in 4 notebooks on Colab

Visual Studio Code

SCAN ME

Semi-supervised learning toolbox

Class-Aware Contrastive Semi-Supervised Learning

Publisher: IEEE Cite This PDF

Fan Yang ; Kai Wu ; Shuyi Zhang ; Guannan Jiang ; Yong Liu ; Feng Zheng ; Wei Zhang ; Chengjie Wang ; Long Zeng **All Authors**

Abstract:
Pseudo-label-based semi-supervised learning (SSL) has achieved great success on raw data utilization. However, its training procedure suffers from confirmation bias due to the noise contained in self-generated artificial labels. Moreover, the model's judgment becomes noisier in real-world applications with extensive out-of-distribution data. To address this issue, we propose a general method named Class-aware Contrastive Semi-Supervised Learning (CCSSL), which is a drop-in helper to improve the pseudo-label quality and enhance the model's robustness in the real-world setting. Rather than treating real-world data as a union set, our method separately handles reliable in-distribution data with class-wise clustering for blending into downstream tasks and noisy out-of-distribution data with image-wise contrastive for better generalization. Furthermore, by applying target reweighting, we successfully emphasize clean label learning and simultaneously reduce noisy label learning. Despite its simplicity, our proposed CCSSL has significant performance improvements over the state-of-the-art SSL methods on the standard datasets CIFAR100 [18] and STL10 [8]. On the real-world dataset Semi-iNat 2021 [27], we improve FixMatch [25] by 9.80% and CoMatch [19] by 3.18%. Code is available https://github.com/TencentYoutuResearch/Classification-SemiCLS.

Published in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Date of Conference: 18-24 June 2022

INSPEC Accession Number: 22095223

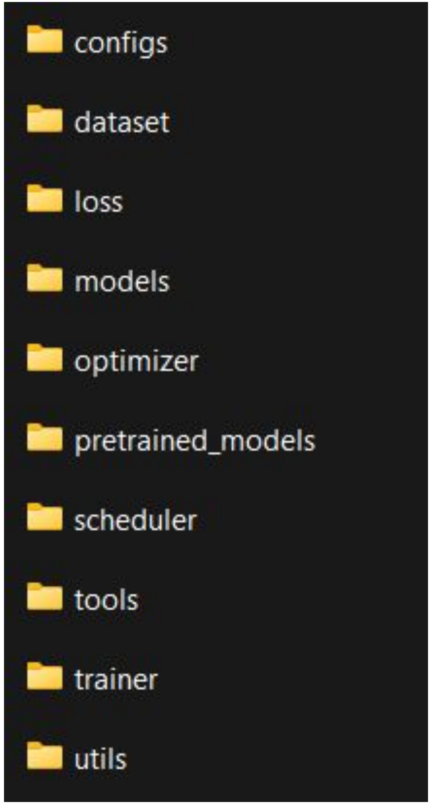
Date Added to IEEE *Xplore*: 27 September 2022

DOI: 10.1109/CVPR52688.2022.01402

[8] F. Yang et al., "Class-Aware Contrastive Semi-Supervised Learning," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 14401-14410, doi: 10.1109/CVPR52688.2022.01402.

Semi-supervised learning toolbox

https://github.com/TencentYoutuResearch/Classification-SemiCLS



[8] F. Yang et al., "Class-Aware Contrastive Semi-Supervised Learning," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 14401-14410, doi: 10.1109/CVPR52688.2022.01402.

Dataset

UCM [7] Dataset [download]:

- 21 clases
- 100 images per class
- 256 x 256 pixels (0.3 m)
- Released 2010

[7] Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.

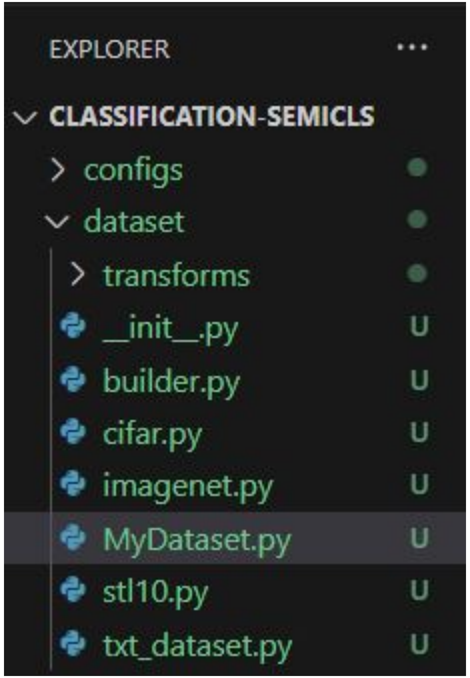
Sampling strategy

- 50 % training and 50% testing (Cheng, G. et al, 2020) [2].
- When using supervised 1,050 samples for training and testing.
- When using semi-supervised: 4 labels per class or 84 labeled samples in total for training (Sohn, K. et al, 2020) [6].
- Labeled to unlabeled ratio: 1-7
    - Depending on the batch size, if Batch labeled=8 then Batch unlabeled=56.

Dataset	Class	# Labels per class	# Training data	# Testing data	Train/Test split
UCM	21	4/25/40	84/525/840	1, 050	50% / 50%

[7] Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.

Creating train and test .txt




```
#val, l-train dataset
class MyDataset(Dataset):
    """
    Interface provided for customized data sets

    names_file: a txt file, each line in the form of "image_path label"

    transform: transform pipline for mydataset

    """
```

/content/MyDrive/SSL4RS/Classification-SemiCLS/data/UCM/Images

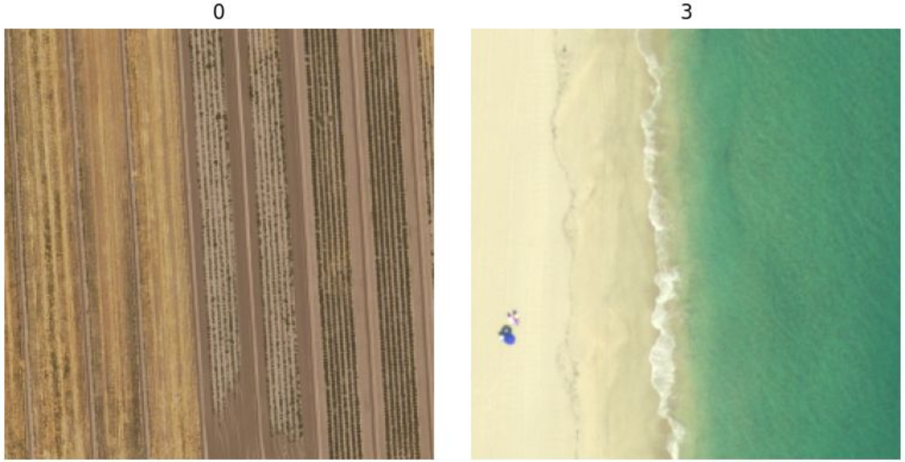
UCM_test.txt

UCM_train.txt

```
./data/UCM/Images/agricultural/agricultural70.tif 0
./data/UCM/Images/agricultural/agricultural59.tif 0
```

Transformation pipeline - No Augmentation

```
names_file = './data/UCM/Images/UCM_train.txt'
dataset = MyDataset(names_file, transform=None)
```



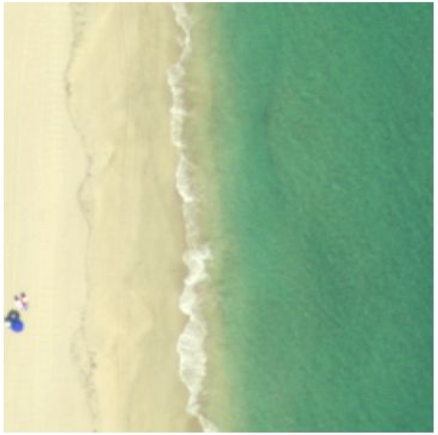
Transformation pipeline - Weak Augmentation

```
transform_w = transforms.Compose([
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.Resize(size=256),
    transforms.CenterCrop(size=224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],std=[0.229, 0.224, 0.225])])
dataset_w = MyDataset(names_file, transform=transform_w)
```

`torchvision.transforms`

0

3



Transformation pipeline - Strong Augmentation 1

```
from dataset.transforms.randaugment import RandAugmentMC
transform_s1 = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomResizedCrop(size=224, scale=(0.2, 1.0)),
    transforms.Lambda(lambda img: RandAugmentMC(n=2, m=10)(img)),   # Use RandAugmentMC directly with desired parameters
    transforms.ToTensor(),
    transforms.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225))
])
dataset_s1 = MyDataset(names_file, transform=transform_s1)
```

`Dataset.transforms.randaugment`

`Class RandAugmentMC`

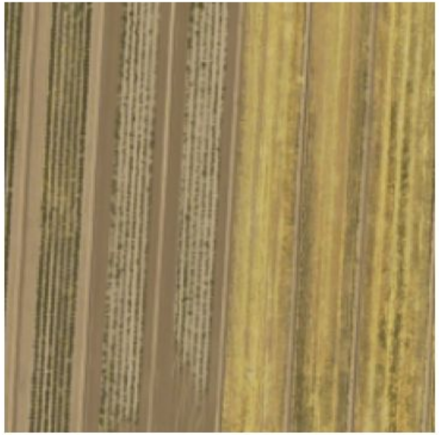
0

3

Transformation pipeline - Strong Augmentation 2

```
transform_s2 = transforms.Compose([
    transforms.RandomResizedCrop(size=224, scale=(0.2, 1.0)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomApply([
        transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1)], p=0.8),
    transforms.RandomGrayscale(p=0.2),
    transforms.ToTensor(),
    transforms.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225))
])
dataset_s2 = MyDataset(names_file, transform=transform_s2)
```

0



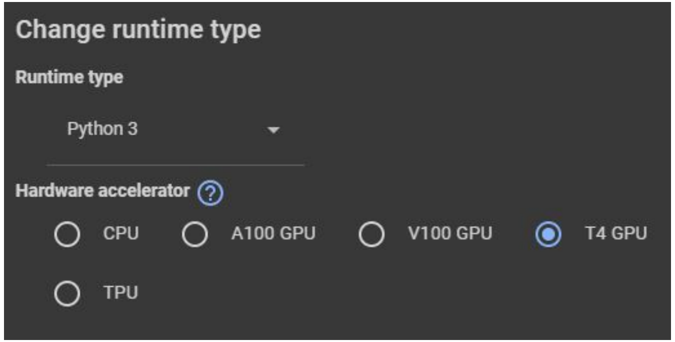
3

Model training

Set up environment, adapt the configuration files and launch the training

16:30 - 17:00 (30 min)

Checking GPU availability




```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   40C    P8     9W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

GPU's memory usage in MiB (mebibytes) 0MiB / 15360MiB.

≈ 16 GB RAM

Modifications: dataset/builder.py

```python
    else:

        # check if .ipynb_checkpoints is in root, and exclude it
        root = os.path.join(cfg.root, '')
        if os.path.isdir(os.path.join(root, ".ipynb_checkpoints")):
            exclude_dir = os.path.join(root, ".ipynb_checkpoints")
        else:
            exclude_dir = None
```

Google colab creates temporary .ipynb inside the image folders and the dataset class gives error when other than .tif extensions are found.

Modifications: train_semi.py

```python
        data_x = labeled_iter.next()
        #data_x = next(labeled_iter)
    except Exception:
        if args.world_size > 1:
            labeled_epoch += 1
            labeled_trainloader.sampler.set_epoch(labeled_epoch)
        labeled_iter = iter(labeled_trainloader)
        data_x = labeled_iter.next()
        #data_x = next(labeled_iter)

    try:
        data_u = unlabeled_iter.next()
        #data_u = next(unlabeled_iter)
    except Exception:
        if args.world_size > 1:
            unlabeled_epoch += 1
            unlabeled_trainloader.sampler.set_epoch(unlabeled_epoch)
        unlabeled_iter = iter(unlabeled_trainloader)
        data_u = unlabeled_iter.next()
        #data_u = next(unlabeled_iter)
```

```python
from mmcv import Config
```

```python
from mmengine.config
import Config
```

Modifications: configuration file

```
train = dict()
```

Specifies algorithm (FixMatch or CoMatch), epochs and loss function

```
model= dict()
```

Details about model architecture (ResNet, WideResNet)

```
data = dict()
```

Parameters for loading and preprocessing the dataset, including number of labeled samples, batch size and augmentation pipeline.

```
trainer
  comatch_ccssl.py
  comatch.py
  fixmatch_ccssl.py
  fixmatch.py
```

```
models
  builder.py
  ema.py
  resnet.py
  wideresnet.py
```

Modifications: configuration file

```
train = dict(eval_step=1024,
             total_steps=2**20, #1024*20,
             trainer=dict(type="FixMatch",
                          threshold=0.95,
                          T=1.,
                          lambda_u=1.,
                          loss_x=dict(
                              type="cross_entropy",
                              reduction="mean"),
                          loss_u=dict(
                              type="cross_entropy",
                              reduction="none"),
                          ))
num_classes = 10 #21
```

1. How often model performance is evaluated.
2. Overall duration of the training process.

`Creating:`

`./Classification-SemiCLS/configs/fm_ucm.py`

Modifications: configuration file

```
model = dict(
        type="wideresnet",
        depth=28,
        widen_factor=2,
        dropout=0,
        num_classes=num_classes,
)
```

```
cifar10_mean = (0.4914, 0.4822, 0.4465)
cifar10_std = (0.2471, 0.2435, 0.2616)

#ucm_mean = (0.485, 0.456, 0.406)
#ucm_std = (0.229, 0.224, 0.225)
```

WideResNet - Number of filters in the ResNet increased by a value of 2.

The wider the network, the more filters, the more GPU.

ImageNet mean and std of the pixel intensities for each color channel (RGB).

[9] S. Zagoruyko and N. Komodakis, "Wide residual networks," in Proceedings of the British Machine Vision Conference (BMVC), 2017, pp. 87.1-87.12, doi: 10.22024/UniKent/01.02.69550, arXiv:1605.07146.

Modifications: configuration file

```
data = dict(
    # CIFAR10SSL, CIFAR100SSL
    type="CIFAR10SSL", #"MyDataset",
    num_workers=4, #0,
    num_labeled=250, #84,
    num_classes=num_classes,
    batch_size=64, #8,
    expand_labels=False,
    mu=7,

    root="./data/CIFAR",
    #root="./UCMerced_LandUse/Images",
    #labeled_names_file="./UCMerced_LandUse/Images/UCM_train.txt",
    #test_names_file="./UCMerced_LandUse/Images/UCM_test.txt",
```

4 labeled examples per class.

Path to images

Modifications: configuration file

```
lpipelines=[[
    # 50% chances that the image is horizontally flipped
    dict(type="RandomHorizontalFlip"),
    # RandomCrop crops a fixed size whereas RandomResizedCrop crops and then resizes.
    dict(type="RandomCrop",
          size=32,
          padding=int(32 * 0.125),
          padding_mode='reflect'),
    #dict(type="RandomResizedCrop", size=224, scale=(0.2, 1.0)),
    dict(type="ToTensor"),
    dict(type="Normalize", mean=cifar10_mean, std=cifar10_std)
    #dict(type="Normalize", mean=ucm_mean, std=ucm_std)
]],
```

```
upipelinse=[[
    dict(type="RandomHorizontalFlip"),
    dict(type="RandomCrop",
         size=32,
         padding=int(32 * 0.125),
         padding_mode='reflect'),
    #dict(type="Resize", size=256),
    #dict(type="CenterCrop", size=224),
    dict(type="ToTensor"),
    dict(type="Normalize", mean=cifar10_mean, std=cifar10_std)
    ],
    [
    dict(type="RandomHorizontalFlip"),
    dict(type="RandomCrop",
         size=32,
         padding=int(32 * 0.125),
         padding_mode='reflect'),
    #dict(type="RandomResizedCrop", size=224, scale=(0.2, 1.0)),
    dict(type="RandAugmentMC", n=2, m=10),
    dict(type="ToTensor"),
    dict(type="Normalize", mean=cifar10_mean, std=cifar10_std)
    #dict(type="Normalize", mean=ucm_mean, std=ucm_std)
]],
```

Modifications: configuration file

Augmentation strategy

- **Weak:**
  - Flip
  - Crop
- **Strong1:**
  - RandomAugment
  - CutOut
- **Strong2:**
  - Random Color Jittering
  - Grayscale Conversion

Modifications: configuration file

```
vpipeline=[
    #dict(type="Resize", size=256),
    dict(type="ToTensor"),
    dict(type="Normalize", mean=cifar10_mean, std=cifar10_std)
    #dict(type="Normalize", mean=ucm_mean, std=ucm_std)
])
```

Some UCM samples are have inconsistent spatial sizes, whiles most have a shape of 256*256*3 some are 253*256*3.

Launch training

```
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 368.00 MiB (GPU 0; 14.75 GiB total capacity; 14.45 GiB already allocated; 168.81 MiB free; 14.46 GiB reserved in total by PyTorch) If reserved memory is >> allocated memory try setting max_split_size_mb to avoid fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

Not all trainers use same amount of GPU

FixMatch

12-24 GB

CoMatch

24-40 GB

+CCSSL

>40 GB

- Image size: 224x224
- Batch size: 8
- WideResNet
- Labeled to unlabeled ratio: 1-7

Launch training

```
2023-10-18 22:14:41,826 - INFO - root -   Train Epoch: 1/  20. Iter:    1/   5. LR: 0.0300 batch_time: 17.171 data_time: 14.079 loss: 3.103 loss_x: 3.103
2023-10-18 22:14:56,879 - INFO - root -   Train Epoch: 1/  20. Iter:    2/   5. LR: 0.0300 batch_time: 16.112 data_time: 13.820 loss: 3.100 loss_x: 3.100
2023-10-18 22:15:10,361 - INFO - root -   Train Epoch: 1/  20. Iter:    3/   5. LR: 0.0300 batch_time: 15.236 data_time: 13.206 loss: 2.877 loss_x: 2.877
2023-10-18 22:15:24,654 - INFO - root -   Train Epoch: 1/  20. Iter:    4/   5. LR: 0.0300 batch_time: 15.000 data_time: 13.081 loss: 3.006 loss_x: 3.006
2023-10-18 22:15:36,954 - INFO - root -   Train Epoch: 1/  20. Iter:    5/   5. LR: 0.0299 batch_time: 14.460 data_time: 12.625 loss: 2.953 loss_x: 2.953
Test Iter:  263/ 263. Data: 0.139s.                    Batch: 0.167s. Loss: 18.7652. top1: 5.24. top5: 23.52. : 100% 263/263 [00:43<00:00,  5.99it/s]
2023-10-18 22:16:20,872 - INFO - root -   Epoch 0 top-1 acc: 5.24
2023-10-18 22:16:20,873 - INFO - root -   Epoch 0 top-5 acc: 23.52
2023-10-18 22:16:21,043 - INFO - root -   Best top-1 acc: 5.24
2023-10-18 22:16:21,043 - INFO - root -   Mean top-1 acc: 5.24
```

Open questions:

- **What does Test Iter mean?**
- **Top-1?**
- **Top-5?**
- **Best top-1 vs. top-1?**

Training output

- 2023-10-20_08:19:55_-1.log
- checkpoint.pth.tar
- fm_ucm.py
- model_best.pth.tar

Your directory structure should now resemble the following:

Classification-SemiCLS

```
results
├ fm_ucm_wres_x2_b1x8_l840
└── fm_ucm_wres_x2_b1x8_l840.log
└── fm_ucm_wres_x2_b1x8_l840.tar
└── fm_ucm_wres_x2_b1x8_l840.py
```

- The `.py` file represents the configuration file.
- The `.log` file contains the training output.
- The `.tar` file contains the trained model.

- The checkpoint is the last epoch and model_best is the epoch that performed the best on the testing set so far.

Modifications: configuration file

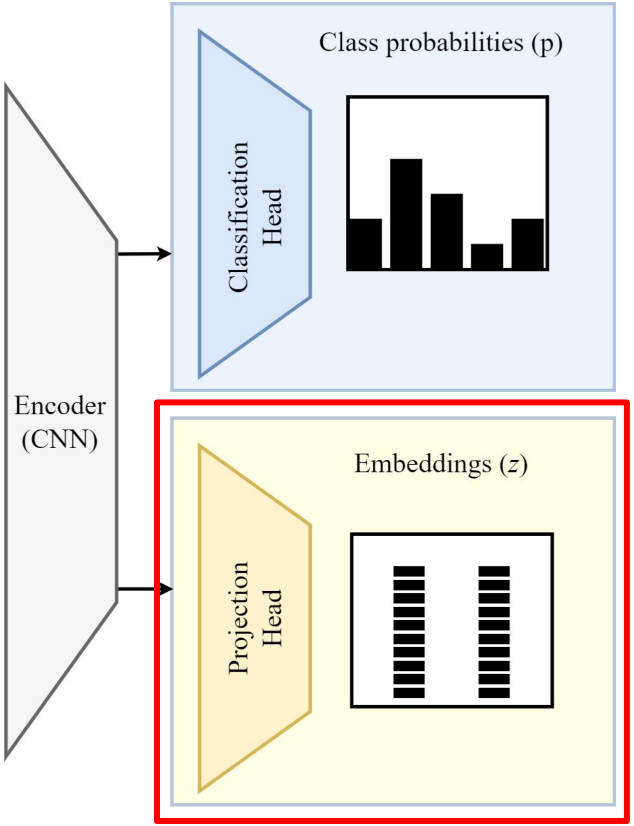
```python
train = dict(
    eval_step=1024,
    total_steps=2**10*512,
    trainer=dict(
        type='CoMatch',
        threshold=0.95, #pseudolabel threshold
        queue_batch=5,  #memory buffer
        contrast_threshold=0.8, #similarity matrix
        da_len=32, #distribution alignment
        T=0.2, # temperature
        alpha=0.9,# 1-alpha for memory smoothed pseudo label
        lambda_u=1.0, #unlabeled loss
        lambda_c=1.0, #contrastive loss
        #supervised loss
        loss_x=dict(type="cross_entropy", reduction="mean"))
```

1. Preparing a new configuration file for CoMatch

`Creating:`

`./Classification-SemiCLS/configs/comatch_ucm.py`

Modifications: configuration file




```
model = dict(
        type="resnet18", #config for resnet purposes
        width=1,
        in_channel=3,
        num_class=num_classes,
        proj=True,
        low_dim=64, # projection head
)
```

Using ResNet18 to reduce GPU usage

Proj=True to get the projection head (embeddings)

<u>Other modifications</u>: reduce image size to 60x60 and augment batch size to 32

[9] S. Zagoruyko and N. Komodakis, "Wide residual networks," in Proceedings of the British Machine Vision Conference (BMVC), 2017, pp. 87.1-87.12, doi: 10.22024/UniKent/01.02.69550, arXiv:1605.07146.

Not all models use same amount of GPU

RN-18

3.5 GB

RN-50

7 GB

WRN-28-2

12 GB

- Image size: 60x60
- Batch size: 32
- CoMatch
- Labeled to unlabeled ratio: 1-7

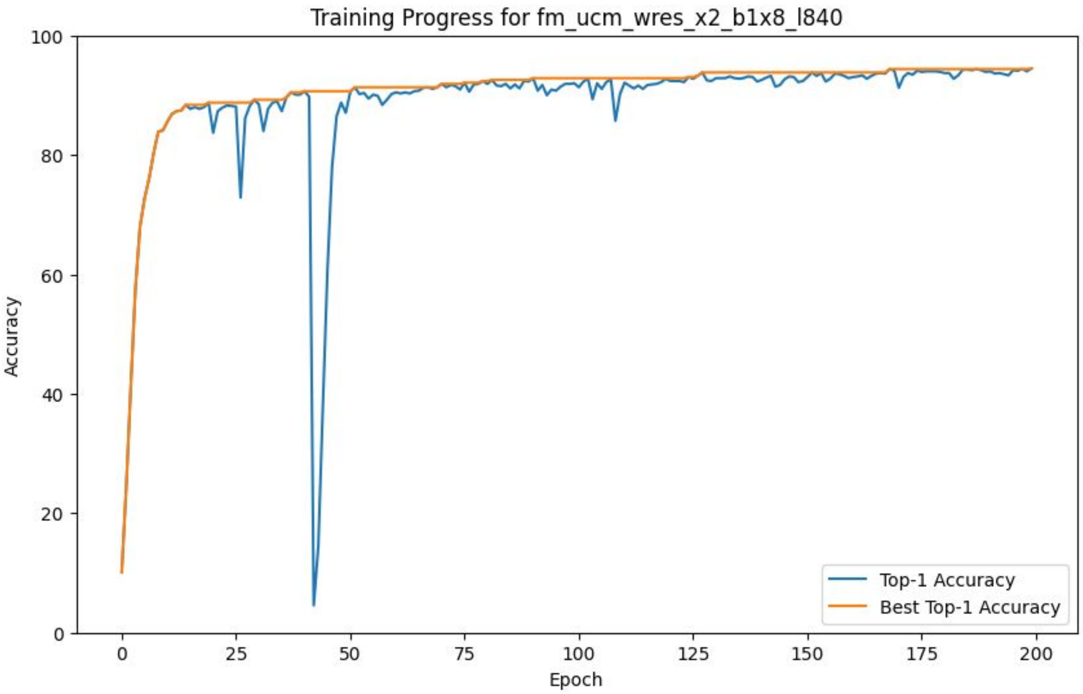
04

Model evaluation and inferencing

Analyze the performance of the model via classification accuracy metrics and embedding visualization

17:00 - 17:30 (30 min)

Classification accuracy (from logs)




```
Best model was obtained on the epoch: 200
The best top-1 accuracy corresponds to: 94.47
```

Case 1:

- Fixmatch on UCM
- 4 labels per class
- WideResNet (Batch=8)

Your directory structure should now resemble the following:

Classification-SemiCLS
results
├ fm_ucm_wres_x2_b1x8_l840
└── fm_ucm_wres_x2_b1x8_l840.log
└── fm_ucm_wres_x2_b1x8_l840.tar
└── fm_ucm_wres_x2_b1x8_l840.py

- The `.py` file represents the configuration file.
- The `.log` file contains the training output.
- The `.tar` file contains the trained model.

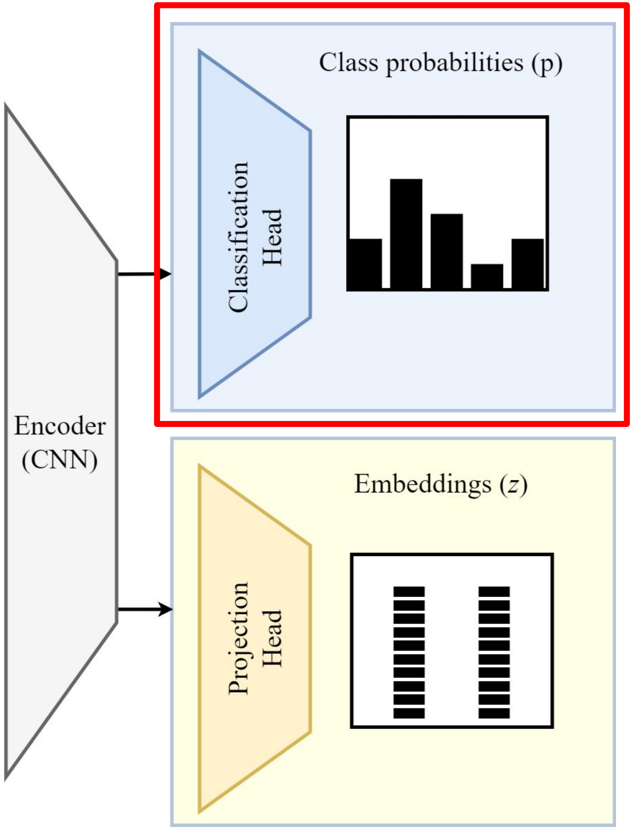
Computational Cost (from logs)

Trainer	Model	Total Parameters	Num Classes Labeled	Mu	Batch size	Training Time
FixMatch	Wideresnet	1.47M	21 4xclass	7	8	2 days, 19:39:18

`Data from:`

`./Classification-SemiCLS/results/computational_costs.csv`

Inference




```
# wideresnet.py (models)
L. 165 for encoder and l.190 for classification head
```

```
#encoder CNN
feat = self.conv1(x)
feat = self.block1(feat)
feat = self.block2(feat)
feat = self.block3(feat)
feat = self.relu(self.bn1(feat))
feat = F.adaptive_avg_pool2d(feat, 1)
feat = feat.view(-1, self.channels)
```

Conv: Convolution layers

Block: Network blocks with multiple convolutional layers for feature extraction

Relu: Activation functions

BN: Batch normalization layers

```
# output
out = self.fc(feat)
return out
```

Fc: fully connected layer followed by a softmax for classification - class predictions.

```
outputs = model(inputs)
```

Classification accuracy (inference)

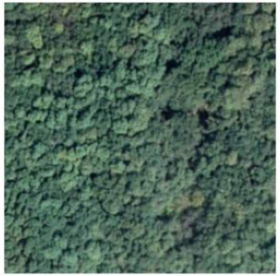
FixMatch - 5 min inference

Kappa: 0.896

True: golfcourse
Predicted: river

True: harbor
Predicted: intersection

True: forest
Predicted: agricultural



Confusion Matrix

True Labels \ Predicted Labels	agricultural	airplane	baseballdiamond	beach	buildings	chaparral	denseresidential	forest	freeway	golfcourse	harbor	intersection	mediumresidential	mobilehomepark	overpass	parkinglot	river	runway	sparseresidential	storagetanks	tenniscourt
agricultural	48	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
airplane	0	49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
baseballdiamond	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
beach	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
buildings	0	0	0	1	41	0	4	0	0	0	0	1	0	0	0	0	0	0	2	1	0
chaparral	0	0	0	0	0	49	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
denseresidential	0	0	3	0	0	0	39	0	0	0	0	7	1	0	0	0	0	0	0	0	0
forest	5	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0
freeway	1	0	0	0	0	0	0	0	49	0	0	0	0	0	0	0	0	0	0	0	0
golfcourse	1	0	1	0	0	0	0	0	0	46	0	0	0	0	0	0	2	0	0	0	0
harbor	0	0	0	1	0	0	0	0	0	0	48	1	0	0	0	0	0	0	0	0	0
intersection	0	0	0	0	1	0	1	0	2	0	0	44	0	0	1	0	0	0	1	0	0
mediumresidential	0	0	0	0	0	0	2	0	0	0	0	4	29	0	0	0	1	0	14	0	0
mobilehomepark	0	0	1	0	0	0	0	0	0	0	0	1	1	44	0	0	0	0	3	0	0
overpass	0	0	0	0	0	0	0	0	2	0	0	0	0	0	48	0	0	0	0	0	0
parkinglot	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	46	2	0	1	0	0
river	2	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	44	0	1	0	0
runway	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0
sparseresidential	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	47	0	0
storagetanks	0	0	4	0	4	0	0	1	0	0	0	6	0	0	0	0	0	0	1	34	0
tenniscourt	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	46

UNIVERSITAT JAUME I · init institute of new imaging technologies · 15TH EDITION GEOMUNDUS CONFERENCE

Classification accuracy (inference)

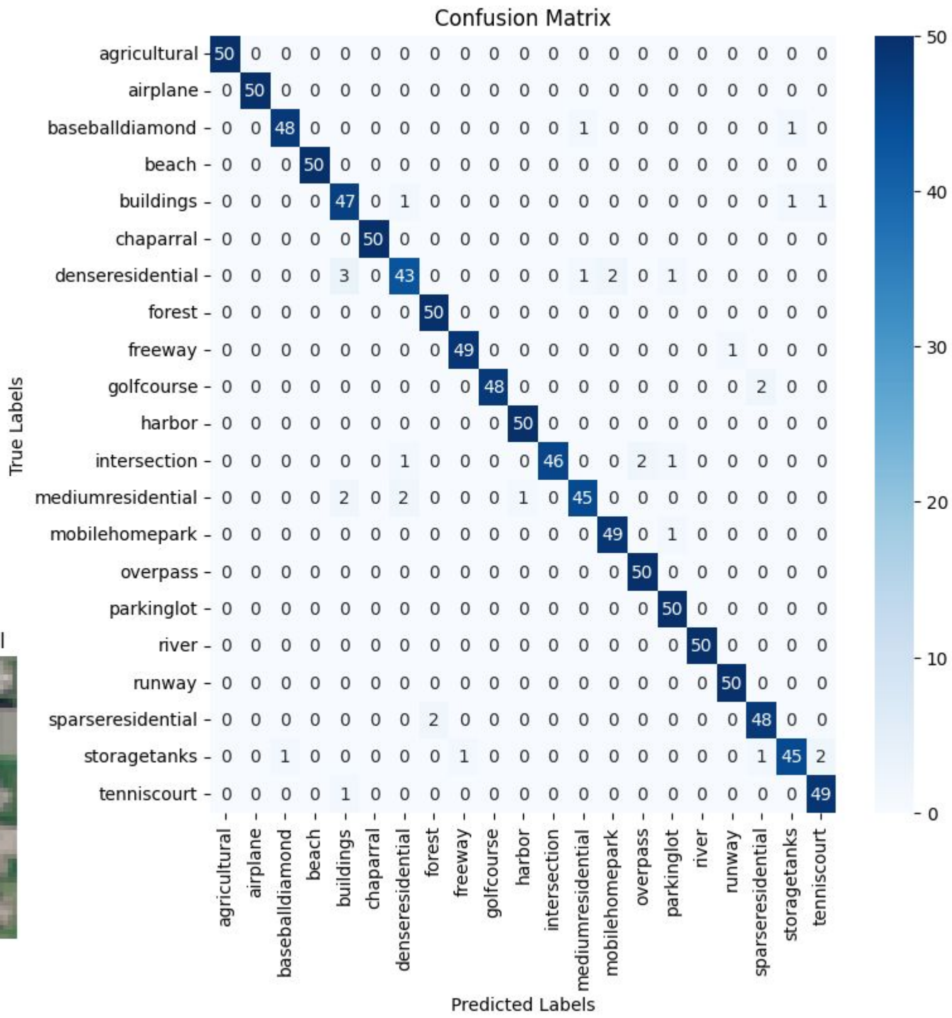
CoMatch - 10 min inference

Kappa: 0.967

True: intersection
Predicted: denseresidential

True: mediumresidential
Predicted: denseresidential

True: mediumresidential
Predicted: denseresidential



UNIVERSITAT JAUME I
init institute of new imaging technologies
15TH EDITION GEOMUNDUS CONFERENCE

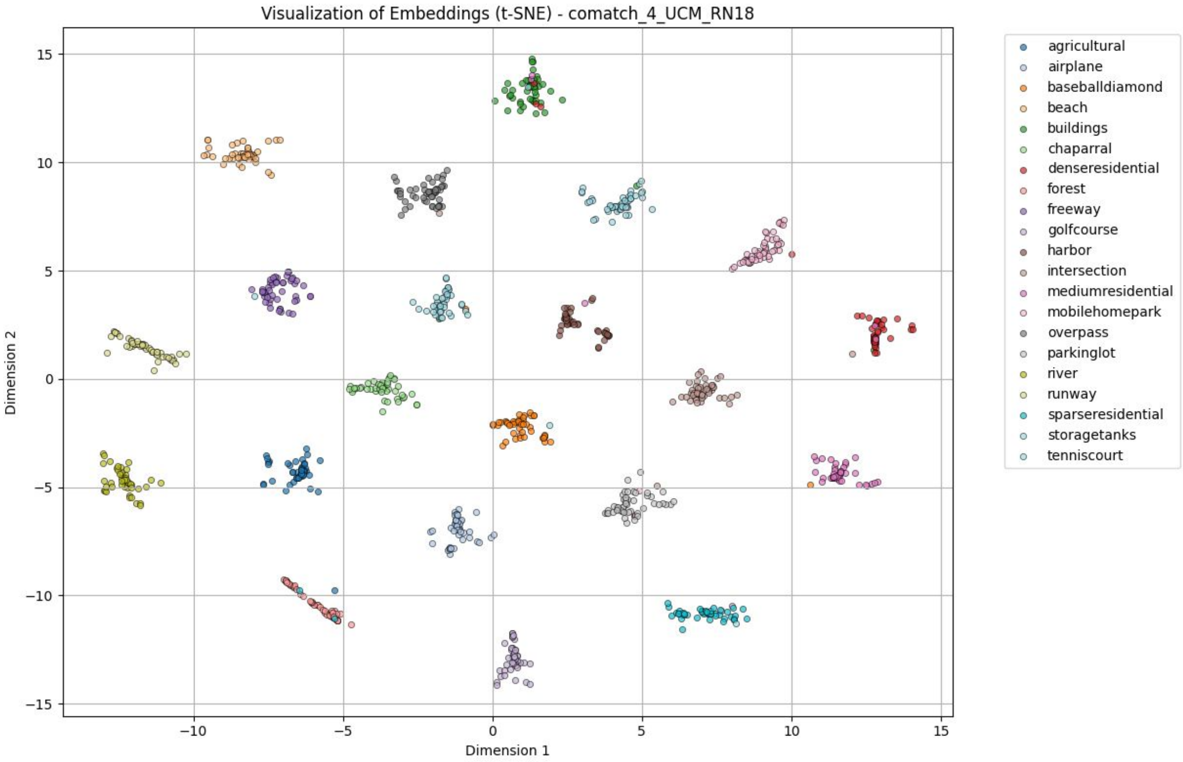
Embeddings

```
#--cfg
model = dict(
    proj=True,
```

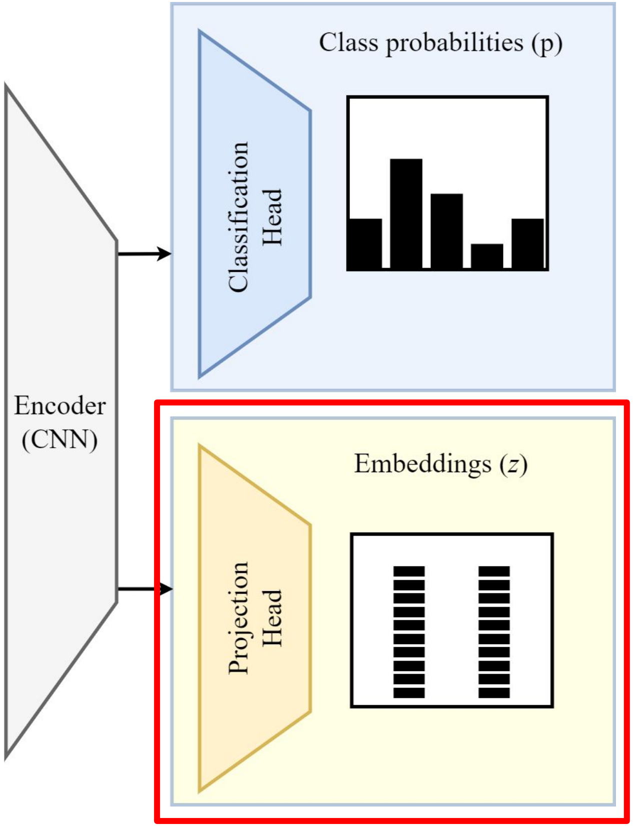
Low-dimensional representation of high-dimensional data.

CNN generate embeddings that capture the underlying structure and relationship of the data.

Projection head: generating feature embeddings for contrastive learning.



Embeddings



L2norm: Normalization layer

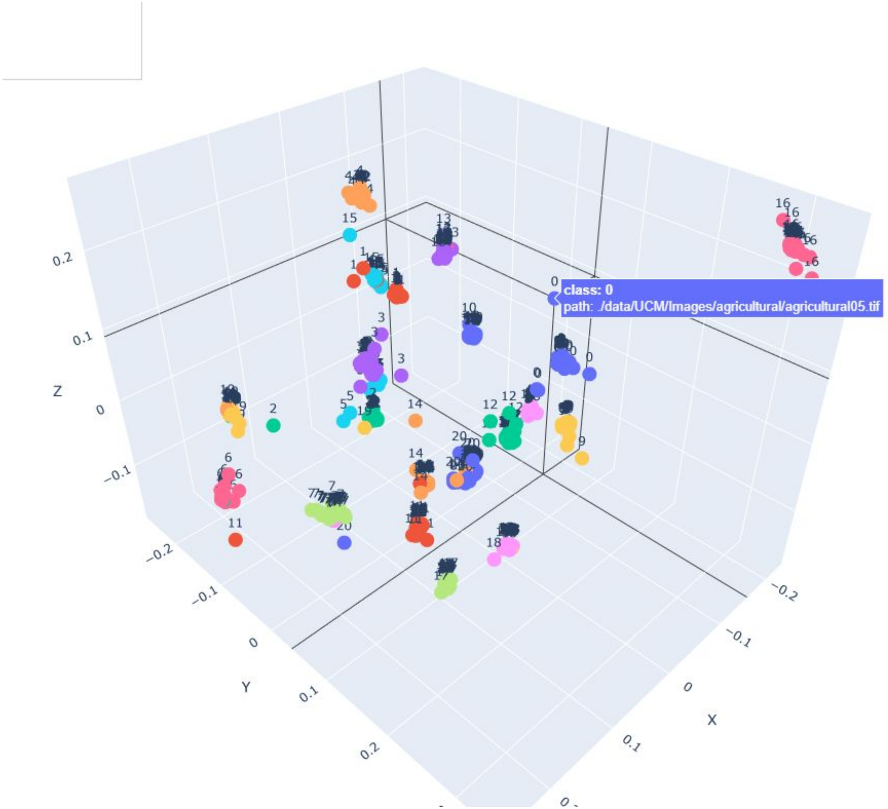
Fc1: Fully connected with LeakyReLU activation

Fc2: Another fully connected to reduce dimensionality of the embeddings.

```
#WideResNet.py
# projection head
if self.proj:
      self.l2norm = Normalize(2)
      self.fc1 = nn.Linear(64 * self.widen_factor, 64 *
self.widen_factor)
      self.relu_mlp = nn.LeakyReLU(inplace=True, negative_slope=0.1)
      self.fc2 = nn.Linear(64 * self.widen_factor, self.low_dim)
```

Embeddings

2D or 3D?



Highlight session

01 Introduction to DSSL

16:00 - 16:15 (15 min)

Feedback:

- ML and DL
- DL RS
- CNN
- SSL Taxonomy
- FixMatch
- CoMatch

Highlight session

02

Data & Code

16:15 - 16:30 (15 min)

Clone GitHub
Download data
Sampling strategy
Train/Test.txt
Augmentation
Pipeline

Feedback:

Highlight session

03 Training model

16:30 - 17:00 (30 min)

Feedback:

Check GPU
Environment setup
Code modifications
Adapt configuration files
Launch training

Highlight session

04 Evaluate model

17:00 - 17:30 (30 min)

Feedback:

Extract data from logs
Classification Accuracy
Computational Costs
Inference with Classification Head (class probabilities)
Inference with Projection Head (embeddings)

Thanks!

Do you have any questions?

isequeir@uji.es

@itzahs

References

Bibliography:

1. Bhavsar, K. (2018, September 14). Understanding your Convolution Network with Visualizations. Towards Data Science. https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b
2. G. Cheng, X. Xie, J. Han, L. Guo and G. -S. Xia, "Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 3735-3756, 2020, doi: 10.1109/JSTARS.2020.3005403 .
3. X. Yang, Z. Song, I. King and Z. Xu, "A Survey on Deep Semi-Supervised Learning," in IEEE Transactions on Knowledge and Data Engineering, 2022, doi: 10.1109/TKDE.2022.3220219.
4. Sharma, A. (2019, June 12). Complete Guide to Data Augmentation for Computer Vision. Towards Data Science. https://towardsdatascience.com/complete-guide-to-data-augmentation-for-computer-vision-1abe4063ad07.
5. Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J Big Data* 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
6. Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. NeurIPS, 33, 2020.
7. Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.
8. F. Yang et al., "Class-Aware Contrastive Semi-Supervised Learning," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 14401-14410, doi: 10.1109/CVPR52688.2022.01402.
9. S. Zagoruyko and N. Komodakis, "Wide residual networks," in Proceedings of the British Machine Vision Conference (BMVC), 2017, pp. 87.1-87.12, doi: 10.22024/UniKent/01.02.69550, arXiv:1605.07146.

Presentation credits:

1. Template by **Slidesgo**
2. Icons by **Flaticon**
3. Infographics and images by **Freepik**