note

November 23, 2025

0.1 About classes

[2]:
```
mapped_classes = {'background': 0, 'asphalt': 5, 'dirt': 1, 'mud': 2, 'water':␣
 ↪3, 'gravel': 4, 'other-terrain': 5, 'tree-trunk': 6, 'tree-foliage': 7,␣
 ↪'brush': 8, 'fence': 9, 'structure': 10, 'pole': 13, 'vehicle': 0, 'rock':␣
 ↪11, 'log': 12, 'other-object': 13, 'sky': 14, 'grass': 15}
new_classes = [[] for _ in range(16)]
for key in mapped_classes:
    value = mapped_classes[key]
    # print(value)
    new_classes[value].append(key)
print(new_classes)
```

```
[['background', 'vehicle'], ['dirt'], ['mud'], ['water'], ['gravel'],
['asphalt', 'other-terrain'], ['tree-trunk'], ['tree-foliage'], ['brush'],
['fence'], ['structure'], ['rock'], ['log'], ['pole', 'other-object'], ['sky'],
['grass']]
```

- First we count the pixel counts of all different classes

[3]:
```
import numpy as np
def compute_map(nb_of_classes, merge_dict):
    '''
    construct mapping between old class and new class
    old classes index: 1-18 (18 classes)
    new classes index: 0-15 (16 classes, 0 represents bg, excluded from␣
 ↪evluation)
    '''
    nb_of_classes += len(merge_dict)
    map_list = [i for i in range(nb_of_classes)]
    shift = 0
    for i in range(nb_of_classes):
        if i in merge_dict:
            shift += 1
        map_list[i] -= shift
    for k, v in merge_dict.items():
        map_list[k] = map_list[v]
    return np.array(map_list)
```

[4]:
```python
import os
import cv2
from concurrent.futures import ThreadPoolExecutor, as_completed
from tqdm import tqdm

def count_gray_values(image_path, gray_values):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    counts = {value: (image == value).sum() for value in gray_values}
    return counts

def process_folder(folder_path, gray_values):
    image_paths = [os.path.join(folder_path, fname) for fname in os.
 ↪listdir(folder_path) if fname.endswith(('.png'))]
    total_counts = {value: 0 for value in gray_values}

    with ThreadPoolExecutor() as executor:
        futures = {executor.submit(count_gray_values, image_path, gray_values):␣
 ↪image_path for image_path in image_paths}
        with tqdm(total=len(futures), desc="Processing images") as pbar:
            for future in as_completed(futures):
                try:
                    result = future.result()
                    for value, count in result.items():
                        total_counts[value] += count
                except Exception as exc:
                    print(f'Error processing {futures[future]}: {exc}')
                pbar.update(1)

    return total_counts

folder_path = '../../data/processed/train/indexLabel'
n_oldclasses = 18
gray_value = range(1, n_oldclasses + 1)
total_gray_value_count = process_folder(folder_path, gray_value)
# total_gray_value_count = {1: 1794, 2: 1600840094, 3: 6857813, 4: 48116742, 5:␣
 ↪103405446, 6: 860988, 7: 2639454701, 8: 10507673624, 9: 203633644, 10:␣
 ↪1620768, 11: 60550301, 12: 3377421, 13: 118755, 14: 14315444, 15: 65429710,␣
 ↪16: 15645583, 17: 1350206976, 18: 1822499988}
print(f'Total count of gray value {gray_value}: {total_gray_value_count}')
merge_classes = {
    1: 6, # asphalt -> other-terrain
    12: 16, # pole -> other-object
    13: 0, # exclude from evaluation
}
n_newclasses = 16
mapping = compute_map(n_newclasses, merge_classes)
new_count = [0] * n_newclasses
```

```
for i in range(1, n_oldclasses + 1):
    new_count[mapping[i]] += total_gray_value_count[i]
print(new_count)
new_classes = ['background', 'dirt', 'mud', 'water', 'gravel', 'other-terrain',␣
 ↪'tree-trunk', 'tree-foliage', 'brush', 'fence', 'structure', 'rock', 'log',␣
 ↪'other-object', 'sky', 'grass']
count_per_class = {new_classes[i]: new_count[i] for i in range(n_newclasses)}
for cls, count in count_per_class.items():
    print(cls, count)
```

```
Processing images: 100%|      | 6051/6051 [00:33<00:00, 178.90it/s]

Total count of gray value range(1, 19): {1: 1794, 2: 1600840094, 3: 6857813, 4:
48116742, 5: 103405446, 6: 860988, 7: 2639454701, 8: 10507673624, 9: 203633644,
10: 1620768, 11: 60550301, 12: 3377421, 13: 118755, 14: 14315444, 15: 65429710,
16: 15645583, 17: 1350206976, 18: 1822499988}
[118755, 1600840094, 6857813, 48116742, 103405446, 862782, 2639454701,
10507673624, 203633644, 1620768, 60550301, 14315444, 65429710, 19023004,
1350206976, 1822499988]
background 118755
dirt 1600840094
mud 6857813
water 48116742
gravel 103405446
other-terrain 862782
tree-trunk 2639454701
tree-foliage 10507673624
brush 203633644
fence 1620768
structure 60550301
rock 14315444
log 65429710
other-object 19023004
sky 1350206976
grass 1822499988
```

[5]:
```
# construct a torch tensor
import torch
print(count_per_class.values())
pixel_count = torch.tensor(list(count_per_class.values()))
print(pixel_count)
```

```
dict_values([118755, 1600840094, 6857813, 48116742, 103405446, 862782,
2639454701, 10507673624, 203633644, 1620768, 60550301, 14315444, 65429710,
19023004, 1350206976, 1822499988])
tensor([     118755,  1600840094,     6857813,    48116742,   103405446,
             862782,  2639454701, 10507673624,   203633644,     1620768,
```

```
       60550301,    14315444,    65429710,    19023004,  1350206976,
      1822499988])
```

0.2 Test for checkpoints

[22]:
```
from utils.data_loading import WSDataset
from pathlib import Path
import torch
import cv2
from utils.evaluate import test
from unet.unet_model import UNet

# test_dir = Path('../../data/processed/test')
test_dir = Path('../../data/processed/val')
test_img_dir = test_dir / 'image'
test_label_dir = test_dir / 'indexLabel'
merge_classes = {
    1: 6, # asphalt -> other-terrain
    12: 16, # pole -> other-object
    13: 0, # exclude from evaluation
}
test_set = WSDataset(test_img_dir, test_label_dir, 0.5, 16, merge_classes, True)
new_classes = ['background', 'dirt', 'mud', 'water', 'gravel', 'other-terrain',␣
 ↪'tree-trunk', 'tree-foliage', 'brush', 'fence', 'structure', 'rock', 'log',␣
 ↪'other-object', 'sky', 'grass']

wl_checkpoint_path = './checkpoints/wl_30/checkpoint_wl_epoch30.pth'
ce_checkpoint_path = './checkpoints/ce_30/checkpoint_epoch30.pth'

if torch.cuda.is_available():
    device = torch.device('cuda')
elif torch.backends.mps.is_available():
    device = torch.device('mps')
else:
    device = torch.device('cpu')
device = torch.device('cpu')

state_dict = torch.load(ce_checkpoint_path, map_location=device)
net1 = UNet(n_channels=3, n_classes=16, bilinear=False)
net1.load_state_dict(state_dict)
print(device.type)

test(net1, test_set, device, new_classes)
print()

net2 = UNet(n_channels=3, n_classes=16, bilinear=False)
state_dict = torch.load(wl_checkpoint_path, map_location=device)
```

```
print('Ious for weighted loss training:')
test(net2, test_set, device, new_classes)
```

```
100%|      | 283/283 [00:03<00:00, 85.69it/s]
Class ['background'] IoU: 0.0000
Class ['dirt'] IoU: 0.40439
Class ['mud'] IoU: 0.0000
Class ['water'] IoU: 0.0000
Class ['gravel'] IoU: 0.0000
Class ['other-terrain'] IoU: 0.0000
Class ['tree-trunk'] IoU: 0.37265
Class ['tree-foliage'] IoU: 0.6337
Class ['bush'] IoU: 1.8124e-06
Class ['fence'] IoU: 0.0000
Class ['structure'] IoU: 0.00017964
Class ['rock'] IoU: 0.0000
Class ['log'] IoU: 0.0000
Class ['other-object'] IoU: 0.00011568
Class ['sky'] IoU: 0.69367
Class ['grass'] IoU: 0.2816
Mean IoU: 0.15908714216

Ious for weighted loss training:
Class ['background'] IoU: 0.0
Class ['dirt'] IoU: 0.4158
Class ['mud'] IoU: 0.02104
Class ['water'] IoU: 0.10708
Class ['gravel'] IoU: 0.0001
Class ['other-terrain'] IoU: 0.0
Class ['tree-trunk'] IoU: 0.46887
Class ['tree-foliage'] IoU: 0.6978
Class ['bush'] IoU: 0.0001
Class ['fence'] IoU: 0.0
Class ['structure'] IoU: 0.1037
Class ['rock'] IoU: 0.0
Class ['log'] IoU: 0.1308
Class ['other-object'] IoU: 0.0271
Class ['sky'] IoU: 0.6733
Class ['grass'] IoU: 0.3271
Mean IoU: 0.185799375
```