Deep Learning

Natural scene classification

U Kang
Seoul National University

In This Lecture

- Natural Scene Data
- Image classification with CNN

Outline

- **Problem Definition**
- Preprocessing Codes

Dataset

- Image data from natural scenes around the world
- 2 types of data
  - Train: data to train
  - Test: data to test

Problem Definition

- Given image, predict it's label among
  - Building, forest, glacier, mountain, sea, street

Method

- Use CNN as our predictor
- CNN can capture local connectivity between each region in image

Outline

- ☑ Problem Definition
- ➡ ☐ **Preprocessing Codes**

Import libraries (1)

- We will use following libraries
  - Tensorflow
  - matplotlib
  - Scikit-learn
  - Opencv-python
  - Numpy
  - Tqdm
  - Seaborn
  - Pandas

Import libraries (2)

- Import libraries

```
import tensorflow.keras.layers as Layers
import tensorflow.keras.models as Models
import tensorflow.keras.optimizers as Optimizer

import os
import matplotlib.pyplot as plot
import seaborn as sns
import pandas as pd
import cv2
import numpy as np

from sklearn.metrics import classification_report
from sklearn.utils import shuffle
from random import randint
```

Prepare data (1)

- Load train and test images
- Configure data path

```
def get_images():
    # read train, test images
    # make it imbalance
    # then merge, shuffle, split

    dir_train = './data/seg_train'
    dir_test = './data/seg_test'

    label_str_to_int = {
        'buildings': 0,
        'forest': 1,
        'glacier': 2,
        'mountain': 3,
        'sea': 4,
        'street': 5,
    }

    train_images = []
    train_labels = []
    test_images = []
    test_labels = []

```

Prepare data (2)

- Since images have different sizes, resize image to fixed size

```
    # read train images
    for label_str in os.listdir(dir_train):
        label_int = label_str_to_int[label_str]
        image_dir = os.path.join(dir_train, label_str)

        for image_file in os.listdir(image_dir):
            image = cv2.imread(os.path.join(image_dir, image_file)) # Read image.
            image = cv2.resize(image, (150, 150)) # Resize images. Some images are different sizes.
            train_images.append(image)
            train_labels.append(label_int)

    # read test images
    for label_str in os.listdir(dir_test):
        label_int = label_str_to_int[label_str]
        image_dir = os.path.join(dir_test, label_str)

        for image_file in os.listdir(image_dir):
            image = cv2.imread(os.path.join(image_dir, image_file)) # Read image.
            image = cv2.resize(image, (150, 150)) # Resize images. Some images are different sizes.
            test_images.append(image)
            test_labels.append(label_int)


    x_train, y_train = shuffle(train_images, train_labels)
    x_test, y_test = shuffle(test_images, test_labels)

    return np.array(x_train), np.array(y_train), np.array(x_test), np.array(y_test)
```

Prepare data (3)

- You can see the results below

```
1 x_train, y_train, x_test, y_test = get_images()
```

```
1 print(x_train.shape, y_train.shape)
2 print(x_test.shape, y_test.shape)
```

```
(10793, 150, 150, 3) (10793,)
(3604, 150, 150, 3) (3604,)
```

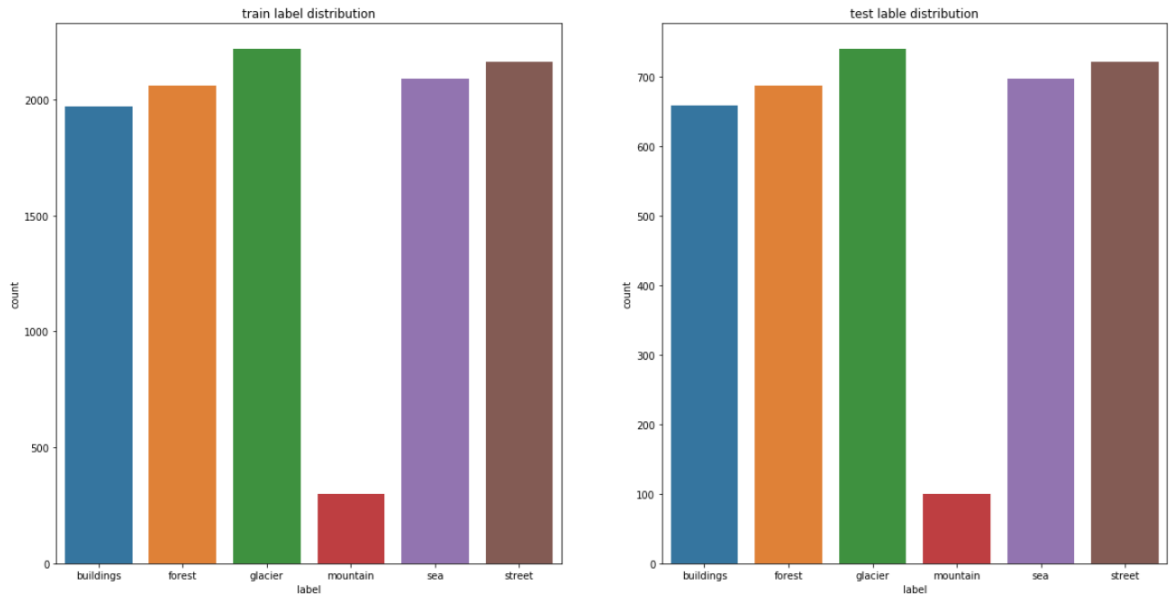
Prepare data (4)

- The ‘mountain’ label is imbalanced

```
# label distribution
train_label_str = [get_classlabel(label_int) for label_int in y_train.tolist()]
test_label_str = [get_classlabel(label_int) for label_int in y_test.tolist()]
labels = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']

df_train_label = pd.DataFrame(train_label_str, columns=['label'])
df_test_label = pd.DataFrame(test_label_str, columns=['label'])
f, ax = plot.subplots(1, 2, figsize=(20, 10))
sns.countplot(x='label', data=df_train_label, ax=ax[0], order=labels)
ax[0].set_title('train label distribution')
sns.countplot(x='label', data=df_test_label, ax=ax[1], order=labels)
ax[1].set_title('test lable distribution')
plot.show()

```



Prepare data (5)

- Also, check some images in dataset

```
def get_classlabel(class_code):
    labels = {2:'glacier', 4:'sea', 0:'buildings', 1:'forest', 5:'street', 3:'mountain'}

    return labels[class_code]
```

```
# some images
f, ax = plot.subplots(5, 5)
f.subplots_adjust(0, 0, 3, 3)
for i in range(0, 5, 1):
    for j in range(0, 5, 1):
        rnd_number = randint(0, x_train.shape[0])
        ax[i, j].imshow(x_train[rnd_number])
        ax[i, j].set_title(get_classlabel(y_train[rnd_number]))
        ax[i, j].axis('off')
```

Questions?