hw1

Group 10

Introduction

The goal of this homework is to split the dataset and avoid data leakage. Additionally, I randomly split the dataset using a fixed seed. This way, the process can be reproduced. I also check data leakage using pairwise set comparison.

Explanation

load_dataset

Takes file root(str) and seed(int) as input. Outputs a dictionary (key=label, value=filenames).

Here, I traverse all the folders under given file root, and traverse all the png files under the specific folder. Files are loaded into an array, shuffled before storing into the daraset dictionary. Note that I used a seed to ensure the process can be reproduced.

```
def load_dataset(file_root, seed):
    """return dictionary: {label: file_name}"""
    dataset = {}
    for folder in sorted(os.listdir(file_root)):
        folder_path = os.path.join(file_root, folder)
        if not os.path.isdir(folder_path): continue # ensure only folders are processed
        label = folder_path[-1]
        # print(label) output: 0 1 2 ... 9
        files = []
        for file in sorted(os.listdir(folder_path)):
            if not file.endswith(".png"): continue
            files.append(os.path.join(folder, file))
        random.seed(seed)
        random.shuffle(files)
        dataset[label] = files
    return dataset
```

split_dataset

Takes dataset and ratio as input. Returns train, val, test arrays.

In the for loop, I calculated the total length of files from a specific label. Then according to the ratio, split the files into 3 parts. Note that there may be some rounding errors. To fix this issue, I calculated n_train, n_val. Then, n_test will be N - n_train - n_val to ensure no data is missed. Lastly, the files are extended from train, val, test. This way, all digits will be included in the 3 parts.

```
def split_dataset(dataset, train_ratio, val_ratio, test_ratio):
    """based on given ratio, extract files from shuffled dataset"""

    train, val, test = [], [], []
    for label, files in dataset.items():
        N = len(files)
        n_train = round(N * train_ratio)
        n_val = round(N * val_ratio)
        n_test = N - n_train - n_val

        train_files = files[:n_train]
        val_files = files[n_train:n_train+n_val]
        test_files = files[n_train+n_val:]

        train.extend([(f, label) for f in train_files])
        val.extend([(f, label) for f in val_files])
        test.extend([(f, label) for f in test_files])
    return train, val, test
```

save_list

A simple function used to record the paths and labels.

```
def save_list(pairs, filename):
    with open(filename, "w") as f:
        for path, label in pairs:
            f.write(f"{path} {label}\n")
```

check_data_leakage

Here I used pairwise comparison using set. Comparing the intersection of two sets allows checking duplicate files. If no data leakage, this function returns false.

```python
def check_data_leakage(train, val, test):
    """Return True if there is data leakage"""

    train_paths = set(path for path, _ in train)
    val_paths   = set(path for path, _ in val)
    test_paths  = set(path for path, _ in test)

    inter_train_val = train_paths & val_paths
    inter_train_test = train_paths & test_paths
    inter_val_test = val_paths & test_paths

    if inter_train_val or inter_train_test or inter_val_test:
        if inter_train_val:
            print(f"Train & Val: {len(inter_train_val)} duplicates")
        if inter_train_test:
            print(f"Train & Test: {len(inter_train_test)} duplicates")
        if inter_val_test:
            print(f"Val & Test: {len(inter_val_test)} duplicates")
        return True
    else:
        return False
```

main function

First defines variables. Second, uses the functions defined above to load dataset, then immediately split dataset. Third, store the results into 3 .txt files. Lastly, check if there is data leakage.

```python
if __name__ == "__main__":

    # === variables ===
    file_root = "./MNIST"
    seed = 42
    train_ratio, val_ratio, test_ratio = 0.7, 0.15, 0.15
    # === variables ===

    dataset = load_dataset(file_root, seed)
    train, val, test = split_dataset(dataset, train_ratio, val_ratio, test_ratio)

    save_list(train, "train_list.txt")
    save_list(val, "val_list.txt")
    save_list(test, "test_list.txt")
    print(f"Train set size: {len(train)}")
    print(f"Validation set size: {len(val)}")
    print(f"Test set size: {len(test)}")

    data_leakage = check_data_leakage(train, val, test)
    if data_leakage:
        print("Data leakage")
    else:
        print("No data leakage")
```

results

Here is the output of terminal:

```
Train set size: 42001
Validation set size: 9000
Test set size: 8999
No data leakage
```