Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

Бакалавр

на тему

“Створення моделі асиметричного потрійного навчання для розв'язання задачі класифікації з адаптацією предметної області“

Виконав: студент 4 курсу, групи МФ-41
спеціальність 122 «Комп’ютерні науки»
освітньо-професійна програма «Інформатика»
Бабенко В.О.

Керівник Меняйлов Э.С.
Рецензент_____

Харків – 2024 року

ЗМІСТ

1. ВСТУП

У сучасному світі, де технології машинного навчання розвиваються з неймовірною швидкістю, задача класифікації даних відіграє ключову роль у багатьох сферах, таких як комп'ютерний зір, обробка природної мови, аналіз даних, медична діагностика, фінанси та багато інших. Класифікація дозволяє автоматизувати процеси прийняття рішень, виявляти приховані закономірності та робити точні передбачення. Проте, традиційні методи класифікації, які демонструють високу ефективність на навчальних даних, часто стикаються з проблемою адаптації до нових умов, коли розподіл даних у цільовій області відрізняється від розподілу даних, використаних для навчання моделі.

Ця проблема, відома як адаптація предметної області, виникає через те, що моделі машинного навчання, як правило, навчаються на специфіці даних, на яких вони були навчені, і не здатні узагальнювати свої знання на нові, відмінні від навчальних, дані. Наприклад, модель, навчена розпізнавати рукописні цифри на зображеннях високої якості, може показати низьку точність на зображеннях низької якості, зображеннях, створених іншими людьми, або зображеннях з різними стилями письма.

Адаптація предметної області - це процес, який дозволяє моделі машинного навчання адаптуватися до нової предметної області, мінімізуючи вплив розбіжностей в розподілі даних. Цей процес є критично важливим для успішного застосування моделей машинного навчання в реальних умовах, де дані можуть змінюватися з часом, відрізнятися за своїми характеристиками, або містити шуми та аномалії.

Існує безліч підходів до адаптації предметної області, які можна розділити на декілька категорій:

- **Методи на основі переважування даних**: ці методи спрямовані на зміну розподілу даних з вихідної області, щоб зробити його більш схожим на розподіл даних з цільової області. Для цього можуть використовуватися різні техніки, такі як зважування зразків, генерація синтетичних даних, або вибірка репрезентативних зразків.
- **Методи на основі навчання ознак**: ці методи спрямовані на виявлення спільних ознак, які є інваріантними до розбіжностей в розподілі даних. Для цього можуть використовуватися методи зниження розмірності, автокодувальники, або спеціальні мережі з навчанням ознак.

- **Методи на основі навчання з учителем**: ці методи використовують невелику кількість розмічених даних з цільової області для доналаштування моделі, навченої на даних з вихідної області.
- **Методи на основі навчання без учителя**: ці методи не вимагають розмічених даних з цільової області і використовують нерозмічені дані для адаптації моделі.

Одним з перспективних підходів до вирішення проблеми адаптації предметної області є **асиметричне тройне навчання (ATDA).** Цей метод, що належить до категорії методів навчання без учителя, використовує три класифікатори, які навчаються на даних з вихідної та цільової областей, а потім спільно покращують свої передбачення шляхом обміну псевдо-мітками. ATDA дозволяє моделі адаптуватися до нової предметної області, підвищуючи її точність класифікації на цільових даних, **не потребуючи при цьому розмічених даних з цільової області.**

Основна ідея ATDA полягає в тому, що кожен класифікатор використовує передбачення двох інших класифікаторів для створення псевдо-міток для нерозмічених даних з цільової області. Ці псевдо-мітки потім використовуються для подальшого навчання класифікаторів. Асиметричність методу полягає в тому, що вага, яку надають передбаченням кожного класифікатора, не обов'язково є однаковою. Це дозволяє врахувати різну "впевненість" класифікаторів у їхніх передбаченнях.

Актуальність теми обумовлена зростаючою потребою в розробці ефективних методів адаптації моделей машинного навчання до нових умов. ATDA пропонує універсальний підхід, який може бути застосований до різних задач класифікації з адаптацією предметної області, забезпечуючи високу точність класифікації без необхідності збору та розмітки великої кількості даних з цільової області. Використання ATDA має ряд переваг:

- **Зменшення потреби в розмічених даних**: ATDA дозволяє використовувати нерозмічені дані з цільової області для покращення точності моделі, що є особливо важливим у випадках, коли збір та розмітка даних є дорогими або трудомісткими процесами.
- **Підвищення точності класифікації**: ATDA дозволяє моделі краще адаптуватися до нової предметної області, що призводить до підвищення точності класифікації на цільових даних.
- **Універсальність**: ATDA може бути застосований до різних задач класифікації з адаптацією предметної області, що робить його універсальним інструментом для вирішення різноманітних задач машинного навчання.

- **Ефективність**: ATDA демонструє високу ефективність порівняно з іншими методами адаптації предметної області, особливо у випадках, коли кількість розмічених даних з цільової області є обмеженою.

Метою даної роботи є створення та дослідження моделі ATDA для вирішення задачі класифікації з адаптацією предметної області. Для досягнення цієї мети будуть поставлені наступні **задачі**:

- **Аналіз існуючих методів** адаптації предметної області, включаючи ATDA та його різновиди. Будуть розглянуті сильні та слабкі сторони кожного методу, а також їх придатність для вирішення конкретної задачі. Особлива увага буде приділена аналізу різних підходів до оцінки якості псевдо-міток та вибору ваг для передбачень кожного класифікатора.
- **Розробка архітектури моделі ATDA** з використанням фреймворка PyTorch. Будуть розглянуті різні варіанти архітектури моделі, включаючи вибір типу нейронної мережі, кількості шарів, типу активаційних функцій тощо. Також будуть досліджені різні підходи до оптимізації параметрів моделі, такі як вибір алгоритму оптимізації, значення швидкості навчання, та методи регуляризації.
- **Реалізація алгоритму ATDA** з урахуванням особливостей обраної задачі класифікації. Будуть розроблені методи предобработки даних, спрямовані на приведення даних до єдиного формату та видалення шумів та аномалій. Також будуть досліджені різні методи оцінки якості псевдо-міток, такі як аналіз розподілу ймовірностей, порівняння з істинними мітками, або використання методів консенсусу.
- **Проведення експериментів** для оцінки ефективності розробленої моделі на різних наборах даних. Експерименти дозволять порівняти ефективність ATDA з іншими методами адаптації предметної області, такими як методи на основі переважування даних, навчання ознак, або навчання з учителем. Також будуть проведені експерименти з різними конфігураціями моделі ATDA для визначення оптимальних параметрів.
- **Аналіз результатів** і виявлення сильних та слабких сторін запропонованого підходу. На основі отриманих результатів будуть сформульовані рекомендації щодо подальшого розвитку та вдосконалення моделі ATDA, а також визначені напрямки подальших досліджень.

Наукова новизна роботи полягає в дослідженні та адаптації алгоритму ATDA для вирішення конкретної задачі класифікації з урахуванням

специфіки даних. Робота також може внести вклад у розвиток нових методів адаптації предметної області, таких як методи ансамблевого навчання, методи з використанням активного навчання, або методи з використанням навчання з підкріпленням.

Практична значимість роботи визначається можливістю застосування розробленої моделі для підвищення точності класифікації в різних сферах, де потрібна адаптація моделі до нових умов. Наприклад, модель ATDA може бути використана для:

- **Розпізнавання образів**: адаптація моделей розпізнавання образів до змін умов освітлення, ракурсу зйомки, фону зображення тощо. Це може бути корисним для розробки систем відеоспостереження, систем розпізнавання осіб, або систем автоматичного керування транспортом.
- **Обробка природної мови**: адаптація моделей машинного перекладу, аналізу тональності тексту, розпізнавання мови тощо до різних мовних стилів, діалектів, або акцентів. Це може бути корисним для розробки систем машинного перекладу, систем голосового пошуку, або систем автоматичного створення текстів.
- **Аналіз даних**: адаптація моделей прогнозування до змін у розподілі даних з часом, наприклад, для прогнозування попиту на товари, прогнозування цін на акції, або прогнозування погоди.
- **Медична діагностика**: адаптація моделей діагностики захворювань до різних типів медичних зображень, таких як рентгенівські знімки, томограми, або зображення з мікроскопів.
- **Фінанси**: адаптація моделей оцінки кредитного ризику, прогнозування банкрутства, або виявлення шахрайства до змін у фінансових даних.

В рамках даної роботи буде проведений аналіз розподілу даних у вихідній та цільовій областях, розроблені методи предобработки даних, а також досліджені різні конфігурації моделі ATDA. Результати роботи дозволять зробити висновки про ефективність запропонованого підходу та його потенціал для вирішення задач класифікації з адаптацією предметної області, що сприятиме подальшому розвитку технологій машинного навчання та їх успішному застосуванню в різних сферах людської діяльності.

2. ПІДГОТОВКА

2.1 Встановлення потрібних бібліотек.

umap-learn: Візуалізація вбудовувань з вихідного та цільового доменів за допомогою UMAP.

```
pip install umap-learn -i https://mirrors.ustc.edu.cn/pypi/web/simple
```

```
Looking in indexes: https://mirrors.ustc.edu.cn/pypi/web/simple
Collecting umap-learn
  Downloading
https://mirrors.bfsu.edu.cn/pypi/web/packages/d1/1b/46802a050b1c55d10c4f59fc6afd2b45ac9b4f62b2e12092d3f599286f14/umap_
learn-0.5.6-py3-none-any.whl (85 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 170.6 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.2.2)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (0.58.1)
Collecting pynndescent>=0.5 (from umap-learn)
  Downloading
https://mirrors.bfsu.edu.cn/pypi/web/packages/bf/06/18c0e17eb245b7caeb861f2ff747adb0575500183b6ec4282d5350d29e9f/pynnd
escent-0.5.12-py3-none-any.whl (56 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.8/56.8 kB 498.2 kB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from umap-learn) (4.66.2)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/dist-packages (from
numba>=0.51.2->umap-learn) (0.41.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pynndescent>=0.5->umap-
learn) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-
learn>=0.22->umap-learn) (3.5.0)
Installing collected packages: pynndescent, umap-learn
Successfully installed pynndescent-0.5.12 umap-learn-0.5.6
```

2.2 Імпорт бібліотек.

```
# Імпортування необхідних бібліотек
import os  # Для роботи з файловою системою
import h5py  # Для роботи з файлами HDF5 (USPS dataset)
import pickle  # Для серіалізації та десеріалізації об'єктів Python
import random  # Для генерації випадкових чисел
from itertools import cycle  # Для створення циклічного ітератора по даних
from tqdm.std import tqdm  # Для відображення прогрес-бару під час навчання
import numpy as np  # Для роботи з масивами
import torch  # Основна бібліотека для машинного навчання
import torch.nn as nn  # Модуль для нейронних мереж
import torch.optim as optim  # Модуль для оптимізаторів
import torch.backends.cudnn as cudnn  # Для оптимізації обчислень на GPU
from torchvision import datasets, transforms  # Для завантаження та перетворення наборів даних
from torch.utils.data import Dataset, Subset, DataLoader, random_split  # Для роботи з наборами даних
from matplotlib import pyplot as plt  # Для візуалізації даних
import umap as umap  # Для зниження розмірності даних та візуалізації
from google.colab import drive  # Для підключення Google Drive (якщо використовується)
```

2.3 Глобальні змінні та підготовка.

```
# Функція для створення каталогу для експерименту
def create_exp_directory(exp_dir):
    # Перевірка, чи каталог вже існує
    if not os.path.exists(exp_dir):
        # Створення каталогу для експерименту
        os.makedirs(exp_dir)
        # Створення підкаталогу для візуалізації UMAP
        os.makedirs(f'{exp_dir}/umap')
        print(f'Directory {exp_dir} created.')  # Повідомлення про створення каталогу
        print(f'Directory {exp_dir}/umap created.')  # Повідомлення про створення підкаталогу

    # Додаткова перевірка на існування підкаталогу UMAP (на випадок помилок)
    if not os.path.exists(f'{exp_dir}/umap'):
        os.makedirs(f'{exp_dir}/umap')
        print(f'Directory {exp_dir}/umap created.')

# Визначення каталогу для експерименту
EXP_DIR = './exps'

# Визначення пристрою для обчислень (GPU, якщо доступний, інакше CPU)
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Визначення каталогу з даними MNIST
SOURCE_DIR = './data/MNIST'

# Підключення Google Drive (якщо використовується)
drive.mount('/content/drive')

# Визначення шляху до файлу USPS
TARGET_DIR = "/content/drive/MyDrive/usps.h5"

# Визначення трансформацій для набору даних USPS (цільовий домен)
T_TRANSFORM = transforms.Compose([
    transforms.ToTensor(),  # Перетворення зображення в тензор PyTorch
    transforms.Pad([6, 6], fill=0.),  # Додавання відступів для узгодження розміру з MNIST (28x28)
    transforms.Normalize(mean=[0.5, ], std=[0.5, ]),  # Нормалізація значень пікселів
])

# Визначення трансформацій для набору даних MNIST (вихідний домен)
S_TRANSFORM = transforms.Compose([
    transforms.ToTensor(),  # Перетворення зображення в тензор PyTorch
    transforms.Normalize(mean=[0.5, ], std=[0.5, ]),  # Нормалізація значень пікселів
])

# Створення каталогу для експерименту
create_exp_directory(EXP_DIR)

# Виведення інформації про пристрій, що використовується
print(f'Currently using device: {DEVICE}')
```

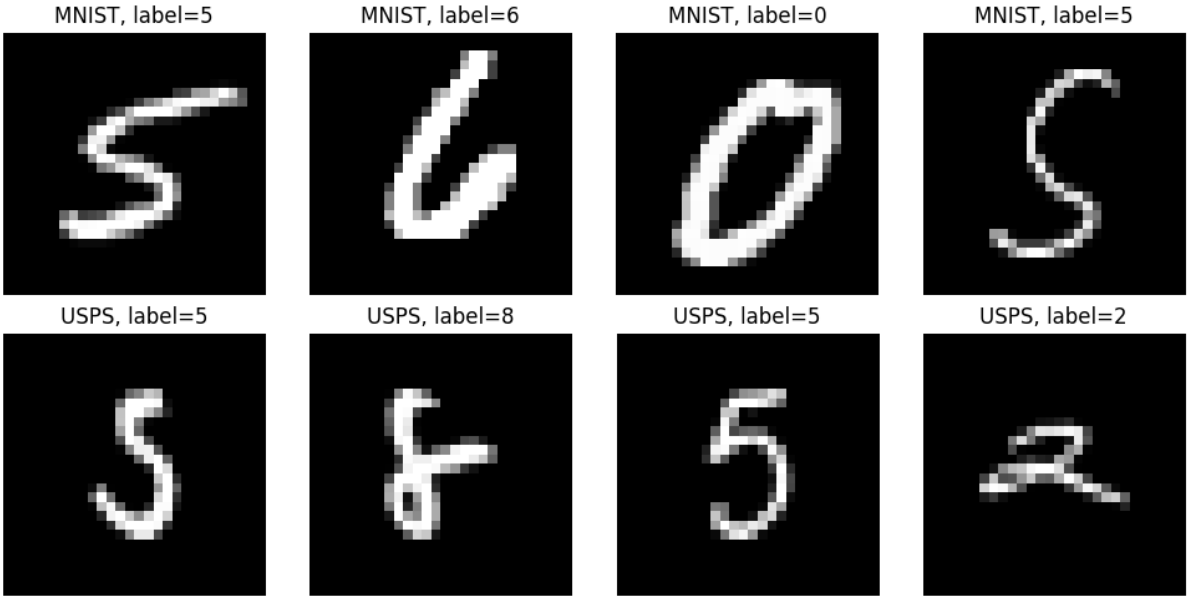
2.4 Гіперпараметри тренування.

```
# Оптимізація обчислень на GPU (якщо доступний)
cudnn.benchmark = True

# Кількість класів в наборах даних
N_CLASSES = 10

# Швидкість навчання для оптимізатора
LR = 0.01

# Коефіцієнт лямбда для обмеження ваги (Weight Constraint)
WC_LAMBDA = 0.01

# Кількість ітерацій адаптації домену
N_ITERS = 30

# Кількість кроків навчання на кожній ітерації
N_STEPS = 1500

# Розмір пакету даних для навчання
BATCH_SIZE = 128

# Поріг достовірності для вибору псевдо-міток
THRESHOLD = 0.95
```

2.5 Візуалізація даних з вихідного та цільового доменів.

```python
# Спеціальний клас для набору даних USPS, успадкований від класу Dataset PyTorch
class USPS(Dataset):
    def __init__(self, data_path, train=True, transform=None, target_transform=None, download=True):
        # Ініціалізація параметрів:
        self.train = train  # Чи використовувати навчальний набір даних
        self.transform = transform  # Трансформації для зображень
        self.target_transform = target_transform  # Трансформації для міток

        # Відкриття файлу HDF5 з даними USPS
        with h5py.File(data_path, 'r') as hf:
            if train:
                train_set = hf.get('train')  # Отримання навчального набору даних
                self.data = train_set.get('data')[:]  # Зображення
                self.labels = train_set.get('target')[:]  # Мітки
            else:
                test_set = hf.get('test')  # Отримання тестового набору даних
                self.data = test_set.get('data')[:]  # Зображення
                self.labels = test_set.get('target')[:]  # Мітки

    def __len__(self):
        # Повертає кількість зображень у наборі даних
        return len(self.data)

    def __getitem__(self, index):
        # Отримання зображення та мітки за індексом
        img = self.data[index].reshape(16, 16, 1)  # Зміна форми зображення на 16x16x1
        label = self.labels[index].astype(np.compat.long)  # Перетворення мітки в цілочисельний тип

        # Застосування трансформацій (якщо задані)
        if self.transform:
            img = self.transform(img)
        if self.target_transform:
            label = self.target_transform(label)

        # Повернення зображення та мітки
        return img, label

# Завантаження навчального набору даних MNIST з визначеними трансформаціями
s_train_dataset = datasets.MNIST(SOURCE_DIR, train=True, transform=S_TRANSFORM, download=True)

# Завантаження навчального набору даних USPS з визначеними трансформаціями
t_train_dataset = USPS(TARGET_DIR, train=True, transform=T_TRANSFORM, download=True)
```

```python
# Випадковий вибір 4 індексів зображень з набору даних USPS
random_indices = random.choices(np.arange(len(t_train_dataset)), k=4)

# Створення фігури для візуалізації
plt.figure(figsize=(10, 5))  # Задаємо розмір фігури

# Цикл по вибраним індексам
for i, index in enumerate(random_indices):
    # Отримання зображень MNIST та USPS за індексом
    s_img = s_train_dataset[index][0].permute(1, 2, 0)  # Зміна порядку розмірностей для відображення
    t_img = t_train_dataset[index][0].permute(1, 2, 0)  # Зміна порядку розмірностей для відображення

    # Відображення зображення MNIST
    plt.subplot(2, 4, i + 1)  # Розташування зображення на сітці 2x4
    plt.imshow(s_img, cmap='gray')  # Відображення зображення в градаціях сірого
    plt.title(f"MNIST, label={s_train_dataset[index][1]}")  # Заголовок з міткою
    plt.axis('off')  # Приховування осей

    # Відображення зображення USPS
    plt.subplot(2, 4, i + 5)  # Розташування зображення на сітці 2x4
    plt.imshow(t_img, cmap='gray')  # Відображення зображення в градаціях сірого
    plt.title(f"USPS, label={t_train_dataset[index][1]}")  # Заголовок з міткою
    plt.axis('off')  # Приховування осей

# Ущільнення розташування зображень на фігурі
plt.tight_layout()

# Відображення фігури
plt.show()
```



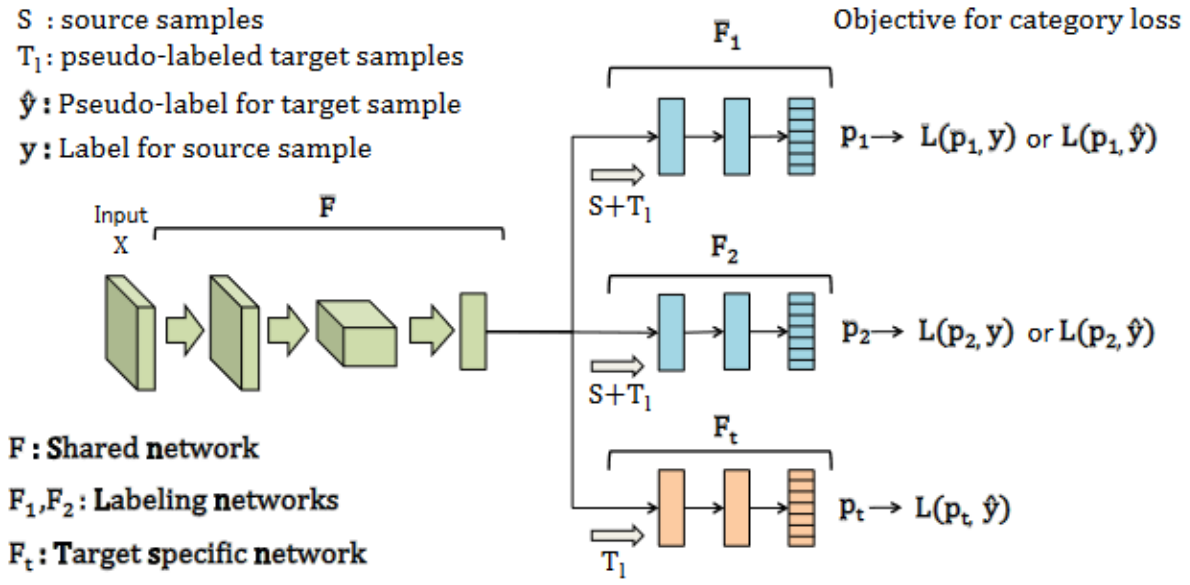
```python
# Функція для обчислення статистики набору даних (середнє значення, стандартне відхилення, максимум та мінімум)
def get_dataset_stats(dataloader):
    num_channels = 0  # Кількість каналів зображення (1 для MNIST та USPS)
    mean = torch.zeros(3)  # Тензор для накопичення значень середнього (ініціалізується нулями)
    std = torch.zeros(3)  # Тензор для накопичення значень стандартного відхилення
    max_values = torch.zeros(3)  # Тензор для накопичення максимальних значень
    min_values = torch.zeros(3)  # Тензор для накопичення мінімальних значень
    num_samples = 0  # Кількість оброблених зображень

    # Цикл по пакетам даних
    for images, _ in tqdm(dataloader):
        batch_size = images.size(0)  # Розмір пакету
        num_channels = images.size(1)  # Кількість каналів

        # Обчислення середнього та стандартного відхилення для кожного каналу окремо
        if num_channels == 1:
            images_view = images.view(batch_size, -1)  # Зміна форми тензора для зручності
            mean += images_view.mean(1).sum(0)
            std += images_view.std(1).sum(0)
            max_values = images.max()
            min_values = images.min()
        else:
            images_view = images.view(batch_size, num_channels, -1)
            mean += images_view.mean(2).sum(0)
            std += images_view.std(2).sum(0)
            max_values = images.max()
            min_values = images.min()
        num_samples += batch_size

    # Обчислення остаточного середнього та стандартного відхилення
    mean /= num_samples
    std /= num_samples

    return mean, std, max_values, min_values

# Створення завантажувача даних для MNIST
s_loader = DataLoader(s_train_dataset, batch_size=BATCH_SIZE)

# Обчислення статистики для набору даних MNIST
s_mean, s_std, s_max, s_min = get_dataset_stats(s_loader)

# Виведення статистики для набору даних MNIST
print(f'Source Dataloader: Mean={s_mean}, STD={s_std}, Max={s_max}, Min={s_min}')

# Створення завантажувача даних для USPS
t_loader = DataLoader(t_train_dataset, batch_size=BATCH_SIZE)

# Обчислення статистики для набору даних USPS
t_mean, t_std, t_max, t_min = get_dataset_stats(t_loader)

# Виведення статистики для набору даних USPS
print(f'Target Dataloader: Mean={t_mean}, STD={t_std}, Max={t_max}, Min={t_min}')
```

```
100%|██████████| 469/469 [00:17<00:00, 26.30it/s]
Source Dataloader: Mean=tensor([-0.7387, -0.7387, -0.7387]), STD=tensor([0.6030, 0.6030, 0.6030]), Max=1.0, Min=-1.0
100%|██████████| 57/57 [00:01<00:00, 44.43it/s]
Target Dataloader: Mean=tensor([-0.8338, -0.8338, -0.8338]), STD=tensor([0.4882, 0.4882, 0.4882]), Max=1.0, Min=-1.0
```

2.6 Модельна архітектура.

Модель ATDA, запропонована

K. Saito, Y. Ushiku, and T. Harada, “Asymmetric Tri-training for Unsupervised Domain Adaptation,” in Proceedings of the 34th International Conference on Machine Learning, PMLR, Jul. 2017, pp. 2988–2997. Accessed: Jul. 05, 2023. [Online]. Available: https://proceedings.mlr.press/v70/saito17a.html



```python
# Клас моделі ATDA, що успадковується від nn.Module PyTorch
class ATDA(nn.Module):
    def __init__(self):
        super().__init__()  # Виклик конструктора батьківського класу

        # Ініціалізація компонентів моделі
        self.f = FeatureExtractor()  # Екстрактор ознак
        self.f1 = LabelNet(drop_rate=0.5)  # Перша мережа міток
        self.f2 = LabelNet(drop_rate=0.5)  # Друга мережа міток
        self.ft = TargetNet(drop_rate=0.2)  # Цільова мережа

    def forward(self, x):
        # Прямий прохід через модель
        embeddings = self.f(x)  # Витяг ознак з вхідних даних
        logits_1 = self.f1(embeddings)  # Прогнозування міток першою мережею
        logits_2 = self.f2(embeddings)  # Прогнозування міток другою мережею
        logits_t = self.ft(embeddings)  # Прогнозування міток цільовою мережею
        return logits_1, logits_2, logits_t  # Повернення logits (вихідні значення перед softmax)

    def get_embeddings(self, x):
        # Отримання вбудовувань (ознак) з екстрактора ознак
        return self.f(x)

# Клас для екстрактора ознак
class FeatureExtractor(nn.Module):
    def __init__(self):
        super().__init__()
        # Згорткові шари
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 48, kernel_size=5, bias=False)
        self.bn_conv2 = nn.BatchNorm2d(48)  # Нормалізація пакетів
        self.maxpool = nn.MaxPool2d(kernel_size=2)  # Макс-пулінг
        self.relu = nn.ReLU()  # Функція активації ReLU

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.conv2(x)
        x = self.relu(x)
        x = self.bn_conv2(x)
        x = self.maxpool(x)
        return x.flatten(1)  # Зміна форми тензора в вектор

# Клас для мережі міток
class LabelNet(nn.Module):
    def __init__(self, drop_rate):
        super().__init__()
        # Лінійні шари та нормалізація пакетів
        self.fc1 = nn.Linear(768, 100, bias=False)
        self.bn1 = nn.BatchNorm1d(100)
        self.fc2 = nn.Linear(100, 100, bias=False)
        self.bn2 = nn.BatchNorm1d(100)
        self.fc3 = nn.Linear(100, 10, bias=False)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=drop_rate)  # Dropout для регуляризації

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.bn1(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.bn2(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc3(x)
        return x

# Клас для цільової мережі (подібний до LabelNet, але з bias)
class TargetNet(nn.Module):
    def __init__(self, drop_rate):
        super().__init__()
        self.fc1 = nn.Linear(768, 100, bias=True)
        self.fc2 = nn.Linear(100, 100, bias=True)
        self.fc3 = nn.Linear(100, 10, bias=True)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=drop_rate)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc3(x)
        return x

# Створення моделі ATDA та переміщення її на обраний пристрій (GPU або CPU)
model = ATDA().to(DEVICE)
```

2.7 Оптимізатор і функція втрат.
Функція втрат

Крім стандартних втрат перехресної ентропії, в роботі введено функцію втрат, яку я реалізував як WeightConstraint, щоб змусити дві головки маркувальної сітки навчатися по-різному за рівнянням

$$L_{WC} = \lambda |W_1^T W_2|$$

де λ - гіперпараметр для масштабування цієї втрати.

Оскільки ми прагнемо маркувати цільові зразки з високою точністю, ми очікуємо, що F_1 і F_2 будуть класифікувати зразки на основі різних точок зору. Тому ми накладаємо обмеження на ваги F_1 та F_2 щоб їхні входи відрізнялися один від одного. Ми додаємо $|W_1^T W_2|$ до функції витрат до функції витрат, де W_1 та W_2 позначають повністю пов'язані ваги шарів F_1 та F_2 які спочатку застосовуються до функції $F(x_i)$.

```python
# Клас для реалізації обмеження ваги (Weight Constraint)
class WeightConstraint(nn.Module):
    def __init__(self, lambda_):
        """
        Конструктор класу WeightConstraint.

        Args:
            lambda_ (float): Коефіцієнт лямбда для обмеження ваги.
                             Цей параметр визначає силу штрафу за схожість ваг двох мереж міток.
        """
        super(WeightConstraint, self).__init__()  # Виклик конструктора батьківського класу (nn.Module)
        self.lambda_ = lambda_  # Збереження значення лямбда

    def forward(self, model1, model2):
        """
        Обчислення штрафу за обмеження ваги.

        Args:
            model1 (nn.Module): Перша мережа міток (f1).
            model2 (nn.Module): Друга мережа міток (f2).

        Returns:
            torch.Tensor: Значення штрафу (втрати) за обмеження ваги.
        """
        # Отримання навчальних параметрів (ваг) першої мережі міток (f1)
        w1 = [param for param in model1.parameters() if param.requires_grad]
        # Зміна форми параметрів у вектор
        w1 = torch.cat([p.view(-1) for p in w1])

        # Отримання навчальних параметрів (ваг) другої мережі міток (f2)
        w2 = [param for param in model2.parameters() if param.requires_grad]
        # Зміна форми параметрів у вектор
        w2 = torch.cat([p.view(-1) for p in w2])

        # Обчислення скалярного добутку векторів ваг
        dot_product = torch.dot(w1, w2)

        # Обчислення штрафу (втрати) за обмеження ваги: лямбда * абсолютне значення скалярного добутку
        loss = self.lambda_ * torch.abs(dot_product)

        return loss  # Повернення значення втрати
```

```python
# Створення оптимізаторів та функцій втрат
optims = {
    'fn': optim.SGD(list(model.f.parameters()) + list(model.f1.parameters()) + list(model.f2.parameters()), lr=LR,
momentum=0.9),
    'ft': optim.SGD(list(model.f.parameters()) + list(model.ft.parameters()), lr=LR, momentum=0.9)
}
criteria = {
    'ce': nn.CrossEntropyLoss(),
    'wc': WeightConstraint(lambda_=WC_LAMBDA)
}
```

3. НАВЧАННЯ ТА ОЦІНКА

3.1 Допоміжні функції для навчання.

```python
# Клас для створення набору даних з тензорів зображень та міток
class CustomImageDataset(Dataset):
    def __init__(self, X: torch.tensor, y: torch.tensor):
        self.X = X  # Тензор зображень
        self.y = y  # Тензор міток

    def __len__(self):
        return self.X.shape[0]  # Повертає кількість зображень

    def __getitem__(self, index):
        img = self.X[index]  # Отримання зображення за індексом
        label = self.y[index]  # Отримання мітки за індексом
        return img, label  # Повернення зображення та мітки

# Функція для вилучення тензорів зображень та міток з набору даних
def extract_datasets(dataset):
    x = []  # Список для зображень
    y = []  # Список для міток
    for data, label in dataset:  # Цикл по набору даних
        x.append(data)  # Додавання зображення до списку
        y.append(label)  # Додавання мітки до списку
    x = torch.stack(x, dim=0)  # Створення тензора зображень
    y = torch.tensor(y, dtype=torch.long)  # Створення тензора міток з типом torch.long
    return x, y  # Повернення тензорів зображень та міток

# Функція для попереднього навчання моделі на вихідному домені
def run_pretrain(model, optims, criteria, X, y):
    model.train()  # Переведення моделі в режим навчання

    # Обнулення градієнтів оптимізатора 'fn'
    optims['fn'].zero_grad()

    # Прямий прохід через модель та обчислення втрат
    logits_1, logits_2, _ = model(X)
    ce_loss_1 = criteria['ce'](logits_1, y.long()).item()  # Втрати крос-ентропії для першої мережі міток
    ce_loss_2 = criteria['ce'](logits_2, y)  # Втрати крос-ентропії для другої мережі міток
    wc_loss = criteria['wc'](model.f1, model.f2)  # Втрати за обмеження ваги

    # Сумарні втрати
    loss_fn = ce_loss_1 + ce_loss_2 + wc_loss

    # Зворотний прохід та оновлення ваг
    loss_fn.backward()
    optims['fn'].step()

    # Обнулення градієнтів оптимізатора 'ft'
    optims['ft'].zero_grad()

    # Прямий прохід через модель та обчислення втрат для цільової мережі
    _, _, logits_t = model(X)
    ce_loss_t = criteria['ce'](logits_t, y)

    # Зворотний прохід та оновлення ваг для цільової мережі
    ce_loss_t.backward()
    optims['ft'].step()

    # Обчислення точності для кожної мережі
    total_loss = (ce_loss_1 + ce_loss_2 + ce_loss_t + wc_loss).item()  # Сумарні втрати
    _, predicted_labels_1 = logits_1.max(1)
    _, predicted_labels_2 = logits_2.max(1)
    _, predicted_labels_t = logits_t.max(1)
    acc_f1 = (predicted_labels_1 == y).to(torch.float).mean().item()  # Точність першої мережі міток
    acc_f2 = (predicted_labels_2 == y).to(torch.float).mean().item()  # Точність другої мережі міток
    acc_ft = (predicted_labels_t == y).to(torch.float).mean().item()  # Точність цільової мережі

    # Повернення втрат та точності
    return total_loss, wc_loss, ce_loss_1, ce_loss_2, ce_loss_t, acc_f1, acc_f2, acc_ft
```

```python
# Функція для перенавчання моделі на вихідному та цільовому доменах
def run_retrain(model, optims, criteria, X, y, mode):
    model.train()  # Переведення моделі в режим навчання

    if mode == 'mixed':  # Навчання екстрактора ознак та мереж міток
        optims['fn'].zero_grad()  # Обнулення градієнтів
        logits_1, logits_2, _ = model(X)  # Прямий прохід
        ce_loss_1 = criteria['ce'](logits_1, y)
        ce_loss_2 = criteria['ce'](logits_2, y)
        wc_loss = criteria['wc'](model.f1, model.f2)
        loss_fn = ce_loss_1 + ce_loss_2 + wc_loss
        loss_fn.backward()  # Зворотний прохід
        optims['fn'].step()  # Оновлення ваг

        # Обчислення точності
        _, predicted_labels_1 = logits_1.max(1)
        _, predicted_labels_2 = logits_2.max(1)
        acc_f1 = (predicted_labels_1 == y).to(torch.float).mean().item()
        acc_f2 = (predicted_labels_2 == y).to(torch.float).mean().item()
        total_loss = (ce_loss_1 + ce_loss_2 + wc_loss).item()

        return total_loss, wc_loss, ce_loss_1, ce_loss_2, acc_f1, acc_f2  # Повернення втрат та точності

    elif mode == 'target':  # Навчання цільової мережі
        optims['ft'].zero_grad()
        _, _, logits_t = model(X)
        ce_loss_t = criteria['ce'](logits_t, y)
        ce_loss_t.backward()
        optims['ft'].step()

        # Обчислення точності
        _, predicted_labels_t = logits_t.max(1)
        acc_ft = (predicted_labels_t == y).to(torch.float).mean().item()

        return acc_ft  # Повернення точності

    else:
        raise ValueError(f'Unknown mode: {mode}. Options: ["mixed", "target"]')  # Помилка, якщо режим невідомий

# Функція для оцінки моделі
def run_eval(model, criteria, X, y, return_logits=False):
    model.eval()  # Переведення моделі в режим оцінки

    # Прямий прохід через модель
    logits_1, logits_2, logits_t = model(X)

    # Повернення logits, якщо потрібно
    if return_logits:
        return logits_1, logits_2, logits_t

    # Обчислення втрат та точності
    ce_loss_1 = criteria['ce'](logits_1, y.long()).item()
    ce_loss_2 = criteria['ce'](logits_2, y.long()).item()
    ce_loss_t = criteria['ce'](logits_t, y.long()).item()
    total_loss = ce_loss_1 + ce_loss_2 + ce_loss_t

    _, predicted_labels_1 = logits_1.max(1)
    _, predicted_labels_2 = logits_2.max(1)
    _, predicted_labels_t = logits_t.max(1)
    acc_f1 = (predicted_labels_1 == y).to(torch.float).mean().item()
    acc_f2 = (predicted_labels_2 == y).to(torch.float).mean().item()
    acc_ft = (predicted_labels_t == y).to(torch.float).mean().item()

    # Повернення втрат та точності
    return total_loss, ce_loss_1, ce_loss_2, ce_loss_t, acc_f1, acc_f2, acc_ft
```

```python
# Функція для вибору псевдо-міток
def judge_func(data, pred_1, pred_2, gt, pool_size, threshold=0.9):
    # Обмеження розміру пулу
    data = data[:pool_size]
    pred_1 = pred_1[:pool_size, :]
    pred_2 = pred_2[:pool_size, :]
    gt = gt[:pool_size, :]

    # Застосування softmax до logits
    p1 = pred_1.softmax(dim=1)
    p2 = pred_2.softmax(dim=1)

    n_samples = pred_1.shape[0]  # Кількість зображень

    # Списки для зберігання вибраних даних
    new_data = []
    new_label = []
    new_gt = []

    # Цикл по зображенням
    for i in range(n_samples):
        cand_data = data[i, :, :, :]  # Зображення
        max_prob_1 = p1[i].max()  # Максимальна ймовірність для першої мережі
        max_prob_2 = p2[i].max()  # Максимальна ймовірність для другої мережі
        c1 = p1[i].argmax()  # Прогнозована мітка першої мережі
        c2 = p2[i].argmax()  # Прогнозована мітка другої мережі

        # Вибір зображень з однаковими прогнозами та високою достовірністю
        if c1 == c2:
            if max(max_prob_1, max_prob_2) > threshold:
                new_label.append(c1)  # Додавання прогнозованої мітки
                new_data.append(cand_data)  # Додавання зображення
                new_gt.append(gt[i].to(torch.long))  # Додавання справжньої мітки

    # Виведення інформації про кількість вибраних зображень
    tqdm.write(f'Number of labeled: {len(new_label)}')
    tqdm.write(f'Pool size: {pool_size}')

    # Повернення вибраних даних або порожніх тензорів
    if not new_data:
        return torch.empty(0, *data.shape[1:]), torch.empty(0, dtype=torch.long), torch.empty(0)
    else:
        return torch.stack(new_data, dim=0), torch.tensor(new_label, dtype=torch.long, device=DEVICE),
torch.stack(new_gt, dim=0).view(-1)

# Функція для валідації моделі на цільовому домені
def validate(model, optims, criteria, t, t_val_loader, best_ft_acc):
    model.eval()  # Переведення моделі в режим оцінки

    with torch.no_grad():  # Вимкнення обчислення градієнтів
        for (x, y) in t_val_loader:  # Цикл по пакетам даних
            x, y = x.to(DEVICE), y.to(DEVICE)  # Переміщення даних на пристрій
            _, _, _, _, _, _, val_acc_ft = run_eval(model, criteria, x, y, return_logits=False)  # Оцінка моделі

            # Збереження найкращої моделі
            if val_acc_ft > best_ft_acc:
                best_ft_acc = val_acc_ft
                ckpt = {'model_weights': model.state_dict(),  # Ваги моделі
                        'fn_optim': optims['fn'].state_dict(),  # Стан оптимізатора 'fn'
                        'ft_optim': optims['ft'].state_dict()}  # Стан оптимізатора 'ft'
                torch.save(ckpt, os.path.join(EXP_DIR, f'best_model_iter_{t}.pt'))  # Збереження моделі

    return best_ft_acc, val_acc_ft  # Повернення найкращої точності та точності на поточній ітерації
```

3.2 Допоміжні функції для візуалізації.

```python
# Функція для візуалізації результатів експерименту
def plot_exp_results(acc_ft_history, label_acc_history, n_samples_history, exp_no=0, show_result=False):
    # Створення фігури з двома y-осями
    _, ax1 = plt.subplots(dpi=120)  # dpi - роздільна здатність зображення

    # Побудова графіка точності цільової мережі
    ax1.plot(acc_ft_history, marker='o', label='Target Net Accuracy')

    # Побудова графіка точності псевдо-міток
    ax1.plot(label_acc_history, marker='o', label='Pseudo Label Accuracy')

    # Налаштування осей
    ax1.set_xlabel('Number of Iterations')  # Назва осі x
    ax1.set_ylabel('Accuracy')  # Назва осі y

    # Створення другої y-осі
    ax2 = ax1.twinx()

    # Побудова графіка кількості вибраних зразків
    ax2.plot(n_samples_history, marker='s', label='Number of Samples', color='purple')
    ax2.set_ylabel('Number of Samples')  # Назва другої осі y

    # Об'єднання легенд з обох осей
    lines_1, labels_1 = ax1.get_legend_handles_labels()
    lines_2, labels_2 = ax2.get_legend_handles_labels()
    lines = lines_1 + lines_2
    labels = labels_1 + labels_2
    ax1.legend(lines, labels, loc='upper center', bbox_to_anchor=(0.5, -0.15), fancybox=True, shadow=True, ncol=3)

    # Налаштування кількості ліній сітки
    ax1.yaxis.set_major_locator(plt.MaxNLocator(len(ax2.get_yticks())))
    ax2.yaxis.set_major_locator(plt.MaxNLocator(len(ax1.get_yticks())))

    # Заголовок графіка
    plt.title(f'MNIST -> USPS | Global Best: {np.max(acc_ft_history)*100:.1f} | No. Iters. vs. Overall Accuracy')

    # Збереження графіка
    plt.savefig(f'{EXP_DIR}/MNIST-USPS_results_exp_{exp_no}.png', dpi=120, bbox_inches='tight')

    # Відображення графіка, якщо потрібно
    if show_result:
        plt.show()

    # Закриття фігури
    plt.close()
```

```python
# Функція для візуалізації вбудовувань за допомогою UMAP
def umap_visualization(model, s_loader, t_loader, title, iter=-1, n_samples=500):
    model.eval()  # Переведення моделі в режим оцінки

    # Списки для зберігання вбудовувань та міток
    s_embeddings, t_embeddings = [], []
    s_y, t_y = [], []

    with torch.no_grad():  # Вимкнення обчислення градієнтів
        # Цикл по завантажувачах даних (MNIST та USPS)
        for loader, embeddings, y in [(s_loader, s_embeddings, s_y), (t_loader, t_embeddings, t_y)]:
            # Цикл по пакетам даних
            for x, y_batch in loader:
                x = x.to(DEVICE)  # Переміщення даних на пристрій
                embedding = model.get_embeddings(x)  # Отримання вбудовувань
                embeddings.append(embedding.cpu().numpy())  # Додавання вбудовувань до списку
                y.append(y_batch.numpy())  # Додавання міток до списку

    # Обмеження кількості зразків
    s_y = np.concatenate(s_y, axis=0)[:n_samples]
    t_y = np.concatenate(t_y, axis=0)[:n_samples]
    s_embeddings = np.concatenate(s_embeddings, axis=0)[:n_samples]
    t_embeddings = np.concatenate(t_embeddings, axis=0)[:n_samples]

    # Об'єднання вбудовувань
    all_embeddings = np.concatenate([s_embeddings, t_embeddings], axis=0)

    # Створення об'єкта UMAP
    reducer = umap.UMAP()

    # Зниження розмірності та отримання 2D вбудовувань
    umap_embeddings = reducer.fit_transform(all_embeddings)

    # Створення фігури
    plt.figure(dpi=120)

    # Відображення вбудовувань MNIST та USPS
    plt.scatter(umap_embeddings[:n_samples, 0], umap_embeddings[:n_samples, 1], label='MNIST', s=7.5, marker='o',
c='#1f77b4', alpha=0.75)
    plt.scatter(umap_embeddings[n_samples:, 0], umap_embeddings[n_samples:, 1], label='USPS', s=7.5, marker='o',
c='#ff7f0e', alpha=0.75)

    # Функція для відображення центрів кластерів
    def plot_cluster_centers(embeddings, color, marker):
        cluster_centers = np.mean(embeddings, axis=0)  # Обчислення центру кластера
        plt.scatter(cluster_centers[0], cluster_centers[1], s=50, c=color, marker=marker)  # Відображення центру
кластера

    # Відображення центрів кластерів для MNIST та USPS
    plot_cluster_centers(umap_embeddings[:n_samples], '#1f77b4', 'o')
    plot_cluster_centers(umap_embeddings[n_samples:], '#ff7f0e', '^')

    # Додавання легенди
    plt.legend()

    # Заголовок графіка
    plt.title(title)

    # Збереження графіка
    plt.savefig(f'{EXP_DIR}/umap/umap_iter_{iter}.png')

    # Закриття фігури
    plt.close()
```

3.3 Навчальний цикл.

Algorithm 1 *iter* denotes the iteration of the training. The function *Labeling* indicates the labeling method. We assign pseudo-labels to samples when the predictions of F_1 and F_2 agree, and at least one of them is confident of their predictions.

Input: data
$\mathbf{X^s} = \{(x_i, t_i)\}_{i=1}^{m}$, $\mathbf{X^t} = \{(x_j)\}_{j=1}^{n}$
$\mathbf{X^t}_l = \emptyset$
for $j = 1$ **to** $iter$ **do**
  Train F, F_1, F_2, F_t with a mini-batch from the training set $\mathcal{S}$
end for
$N_t = N_{init}$
$\mathbf{X^t}_l = \text{Labeling}(F, F_1, F_2, \mathbf{X^t}, N_t)$
$\mathcal{L} = \mathbf{X^s} \cup \mathbf{X^t}_l$
for K steps **do**
  for $j = 1$ **to** $iter$ **do**
    Train F, F_1, F_2 with mini-batch from training set $\mathcal{L}$
    Train F, F_t with mini-batch from training set $\mathbf{X^t}_l$
  end for
  $\mathbf{X^t}_l = \emptyset$, $N_t = K/20 * n$
  $\mathbf{X^t}_l = \text{Labeling}(F, F_1, F_2, \mathbf{X^t}, N_t)$
  $\mathcal{L} = \mathbf{X^s} \cup \mathbf{X^t}_l$
end for

```python
# Ініціалізація списків для збереження історії точності, точності псевдо-міток та кількості зразків
acc_ft_history = []
label_acc_history = []
n_samples_history = []

# Цикл по ітераціям адаптації домену
for t in range(N_ITERS):
    print(f'Phase {t}')  # Виведення номера поточної фази

    # Перша ітерація: ініціалізація наборів даних та завантажувачів
    if t == 0:
        # Завантаження навчального набору даних MNIST
        s_train_dataset = datasets.MNIST(SOURCE_DIR, train=True, transform=S_TRANSFORM, download=True)
        # Створення циклічного ітератора для набору даних MNIST
        s_train_iterator = cycle(DataLoader(s_train_dataset, batch_size=BATCH_SIZE, shuffle=True))

        # Завантаження навчального набору даних USPS
        t_dataset = USPS(TARGET_DIR, train=True, transform=T_TRANSFORM, download=True)

        # Розділення набору даних USPS на навчальну та валідаційну частини
        indices = list(range(len(t_dataset)))
        train_indices = indices[:-500]
        val_indices = indices[-500:]

        # Створення навчального та валідаційного наборів даних USPS
        t_train_dataset = Subset(t_dataset, train_indices)
        t_val_dataset = Subset(t_dataset, val_indices)

        # Створення завантажувача даних для валідаційного набору USPS
        t_val_loader = DataLoader(t_val_dataset, batch_size=BATCH_SIZE, shuffle=False)

    else:
        # Об'єднання вихідного набору даних та псевдо-маркованих даних цільового домену
        tqdm.write('Preparing dataloaders...')
        s_train_x, s_train_y = extract_datasets(s_train_dataset)
        s_train = torch.cat([s_train_x.to(DEVICE), new_data], dim=0)
        s_label = torch.cat([s_train_y.to(DEVICE), new_label], dim=0)
        combined_dataset = CustomImageDataset(s_train, s_label)

        # Створення циклічного ітератора для об'єднаного набору даних
        should_drop_last = len(combined_dataset) % int(BATCH_SIZE / 2) == 1
        combined_iterator = cycle(DataLoader(combined_dataset, batch_size=int(BATCH_SIZE / 2), shuffle=True,
drop_last=should_drop_last))

        # Створення циклічного ітератора для набору даних з псевдо-мітками
        new_dataset = CustomImageDataset(new_data, new_label)
        should_drop_last = len(new_dataset) % BATCH_SIZE == 1
        new_iterator = cycle(DataLoader(new_dataset, batch_size=BATCH_SIZE, shuffle=True,
drop_last=should_drop_last))
```

```
# Цикл навчання
    best_ft_acc = 0  # Ініціалізація змінної для збереження найкращої точності цільової мережі

    # Цикл по кроках навчання
    for i in tqdm(range(N_STEPS)):
        # Перша ітерація: попереднє навчання моделі на вихідному домені
        if t == 0:
            model.train()
            x0, y0 = next(s_train_iterator)  # Отримання пакету даних з вихідного домену
            x0, y0 = x0.to(DEVICE), y0.to(DEVICE)  # Переміщення даних на пристрій
            # Виконання попереднього навчання та отримання втрат та точності
            batch_loss, wc_loss, ce_loss_1, ce_loss_2, _, acc_f1, acc_f2, _ = run_pretrain(model, optims, criteria,
x0, y0)

            # Валідація моделі кожні 500 кроків
            if i % 500 == 0:
                best_ft_acc, val_acc_ft = validate(model, optims, criteria, t, t_val_loader, best_ft_acc)
                # Виведення інформації про втрати та точність
                tqdm.write(f'Batch Loss: {batch_loss:.3f}, WC Loss: {wc_loss:.3f}, CE Loss 1: {ce_loss_1:.3f}, CE
Loss 2: {ce_loss_2:.3f}, Acc F1: {acc_f1*100:.1f}, Acc F2: {acc_f2*100:.1f}, Val Acc Ft: {val_acc_ft*100:.1f}')

        # Наступні ітерації: перенавчання моделі на вихідному та цільовому доменах
        if t >= 1:
            model.train()
            # Отримання пакету даних з об'єднаного набору
            x0, y0 = next(combined_iterator)
            x0, y0 = x0.to(DEVICE), y0.to(DEVICE)
            # Перенавчання моделі на об'єднаному наборі даних (екстрактор ознак та мережі міток)
            batch_loss, wc_loss, ce_loss_1, ce_loss_2, acc_f1, acc_f2 = run_retrain(model, optims, criteria, x0, y0,
mode='mixed')

            # Отримання пакету даних з набору даних з псевдо-мітками
            x1, y1 = next(new_iterator)
            x1, y1 = x1.to(DEVICE), y1.to(DEVICE)
            # Перенавчання моделі на наборі даних з псевдо-мітками (цільова мережа)
            acc_ft = run_retrain(model, optims, criteria, x1, y1, mode='target')

            # Валідація моделі кожні 500 кроків
            if i % 500 == 0:
                best_ft_acc, val_acc_ft = validate(model, optims, criteria, t, t_val_loader, best_ft_acc)
                # Виведення інформації про втрати та точність
                tqdm.write(f'Batch Loss: {batch_loss:.3f}, WC Loss: {wc_loss:.3f}, CE Loss 1: {ce_loss_1:.3f}, CE
Loss 2: {ce_loss_2:.3f}, Acc F1: {acc_f1*100:.1f}, Acc F2: {acc_f2*100:.1f}, Acc Ft: {acc_ft*100:.1f}, Val Acc Ft:
{val_acc_ft*100:.1f}')

    # Призначення псевдо-міток
    tqdm.write('Assigning Pseudo-labels...')
```

```
# Списки для збереження зображень, прогнозів та справжніх міток
    candidates_x = []
    pred_1_stack = torch.zeros((0, N_CLASSES), device=DEVICE)
    pred_2_stack = torch.zeros((0, N_CLASSES), device=DEVICE)
    gt_stack = torch.zeros((0, 1), device=DEVICE)

    # Створення ітератора для навчального набору даних USPS
    t_train_iterator = iter(DataLoader(t_train_dataset, batch_size=BATCH_SIZE, shuffle=True))

    # Визначення кількості пакетів даних для обробки
    stack_num = int(min(len(t_train_dataset) / BATCH_SIZE, 100 * (t + 1)))

    # Цикл по пакетам даних
    for _ in tqdm(range(stack_num)):
        model.eval()  # Переведення моделі в режим оцінки
        X, y = next(t_train_iterator)  # Отримання пакету даних
        X, y = X.to(DEVICE), y.to(DEVICE)  # Переміщення даних на пристрій
        candidates_x.append(X)  # Додавання зображень до списку

        # Отримання прогнозів моделі (logits) без обчислення градієнтів
        with torch.no_grad():
            pred_1, pred_2, _ = run_eval(model, criteria, X, y, return_logits=True)

        # Додавання прогнозів та справжніх міток до списків
        pred_1_stack = torch.vstack([pred_1_stack, pred_1])
        pred_2_stack = torch.vstack([pred_2_stack, pred_2])
        gt_stack = torch.cat([gt_stack, y.unsqueeze(1)], dim=0)

    # Об'єднання зображень у тензор
    candidates_x = torch.cat(candidates_x, dim=0)

    # Визначення розміру пулу для вибору псевдо-міток
    if t == 0:
        pool_size = max(int((t + 1) / 20 * pred_1_stack.shape[0]), 1000)
    else:
        pool_size = min(max(int((t + 1) / 20 * pred_1_stack.shape[0]), 1000), 40000)

    # Вибір псевдо-міток
    new_data, new_label, new_gt = judge_func(candidates_x, pred_1_stack, pred_2_stack, gt_stack, pool_size,
threshold=THRESHOLD)

    # Обчислення точності псевдо-міток
    label_acc = (new_gt.view(-1) == new_label.view(-1)).to(torch.float).mean().item()
    tqdm.write(f'Pseudo-label accuracy: {label_acc*100:.1f}')
```

```
# Оцінка моделі
    tqdm.write('Evaluating...')

    # Завантаження тестових наборів даних MNIST та USPS
    s_test_dataset = datasets.MNIST(SOURCE_DIR, train=False, transform=S_TRANSFORM, download=True)
    t_test_dataset = USPS(TARGET_DIR, train=False, transform=T_TRANSFORM, download=True)

    # Створення завантажувачів даних для тестових наборів
    s_test_loader = DataLoader(s_test_dataset, batch_size=BATCH_SIZE, shuffle=False)
    t_test_loader = DataLoader(t_test_dataset, batch_size=BATCH_SIZE, shuffle=False)

    # Створення ітераторів для тестових наборів
    s_test_iterator = cycle(s_test_loader)
    t_test_iterator = iter(t_test_loader)

    # Ініціалізація змінних для накопичення точності
    total_source = 0
    total_target = 0
    total_acc_1 = 0
    total_acc_2 = 0
    s_size = 0
    t_size = 0

    # Визначення кількості ітерацій для обробки тестового набору USPS
    n_iter = int(len(t_test_loader.dataset) / BATCH_SIZE) + 1

    # Цикл по пакетам даних
    for _ in tqdm(range(n_iter)):
        # Отримання пакетів даних з тестових наборів
        x0, y0 = next(s_test_iterator)
        x1, y1 = next(t_test_iterator)
        x0, y0 = x0.to(DEVICE), y0.to(DEVICE)  # Переміщення даних на пристрій
        x1, y1 = x1.to(DEVICE), y1.to(DEVICE)
```

```
# Оцінка моделі на тестових наборах без обчислення градієнтів
        with torch.no_grad():
            _, _, _, _, s_acc, _, _ = run_eval(model, criteria, x0, y0, return_logits=False)
            _, _, _, _, t_acc_f1, t_acc_f2, t_acc_ft = run_eval(model, criteria, x1, y1, return_logits=False)

        # Накопичення точності
        total_source += s_acc * x0.shape[0]
        total_target += t_acc_ft * x1.shape[0]
        total_acc_1 += t_acc_f1 * x1.shape[0]
        total_acc_2 += t_acc_f2 * x1.shape[0]
        s_size += x0.shape[0]
        t_size += x1.shape[0]

    # Обчислення остаточної точності
    total_acc_f1 = total_acc_1 / t_size
    total_acc_f2 = total_acc_2 / t_size
    total_acc_ft = total_target / t_size
    total_acc_s = total_source / s_size

    # Виведення інформації про точність
    tqdm.write(f'Target Ft Acc: {total_acc_ft*100:.1f}, Target F1 Acc: {total_acc_f1*100:.1f}, Target F2 Acc:
{total_acc_f2*100:.1f}, Source F1 Acc: {total_acc_s*100:.1f}')

    # Збереження історії точності, точності псевдо-міток та кількості зразків
    acc_ft_history.append(total_acc_ft)
    label_acc_history.append(label_acc)
    n_samples_history.append(new_label.numel())

    # Візуалізація результатів та вбудовувань
    plot_exp_results(acc_ft_history, label_acc_history, n_samples_history)
    umap_visualization(model, s_test_loader, t_test_loader, iter=t, n_samples=1000, title=f'UMAP Visualization at
Iteration {t}')

# Візуалізація остаточних результатів
plot_exp_results(acc_ft_history, label_acc_history, n_samples_history, show_result=True)

# Збереження максимальних значень точності
total_acc_s = np.max(total_acc_s)
total_acc_ft = np.max(total_acc_ft)
total_acc_f1 = np.max(total_acc_f1)
total_acc_f2 = np.max(total_acc_f2)
```

3.4 Результати експерименту.

```
Phase 0
  0%|          | 1/1500 [00:00<19:24,  1.29it/s]Batch Loss: 9.286, WC Loss: 1.993, CE Loss 1: 2.448, CE Loss 2:
2.511, Acc F1: 13.3, Acc F2: 10.2, Val Acc Ft: 16.4
 33%|███       | 501/1500 [02:33<05:00,  3.33it/s]Batch Loss: 2.906, WC Loss: 0.001, CE Loss 1: 2.808, CE Loss 2:
0.072, Acc F1: 2.3, Acc F2: 97.7, Val Acc Ft: 94.8
 67%|██████    | 1001/1500 [04:36<03:22,  2.46it/s]Batch Loss: 3.011, WC Loss: 0.001, CE Loss 1: 2.842, CE Loss 2:
0.103, Acc F1: 2.3, Acc F2: 97.7, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [06:37<00:00,  3.78it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.32it/s]
Number of labeled: 2
Pool size: 1000
Pseudo-label accuracy: 100.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.31it/s]
Target Ft Acc: 94.2, Target F1 Acc: 1.0, Target F2 Acc: 93.7, Source F1 Acc: 0.6
Phase 1
Preparing dataloaders...
  0%|          | 3/1500 [00:00<03:30,  7.12it/s]Batch Loss: 2.951, WC Loss: 0.000, CE Loss 1: 2.879, CE Loss 2:
0.071, Acc F1: 0.0, Acc F2: 96.9, Acc Ft: 100.0, Val Acc Ft: 94.8
 33%|███       | 502/1500 [00:37<01:47,  9.31it/s]Batch Loss: 0.180, WC Loss: 0.002, CE Loss 1: 0.135, CE Loss 2:
0.044, Acc F1: 95.3, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 94.0
 67%|██████    | 1003/1500 [01:18<00:47, 10.56it/s]Batch Loss: 0.124, WC Loss: 0.000, CE Loss 1: 0.081, CE Loss 2:
0.043, Acc F1: 98.4, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 92.2
100%|██████████| 1500/1500 [01:56<00:00, 12.92it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 13.15it/s]
Number of labeled: 931
Pool size: 1000
Pseudo-label accuracy: 98.3
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.78it/s]
Target Ft Acc: 91.9, Target F1 Acc: 93.5, Target F2 Acc: 93.7, Source F1 Acc: 97.9
```

```
Phase 2
Preparing dataloaders...
  0%|          | 1/1500 [00:00<17:47,  1.40it/s]Batch Loss: 0.058, WC Loss: 0.001, CE Loss 1: 0.027, CE Loss 2:
0.030, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 90.6, Val Acc Ft: 95.7
 33%|███       | 502/1500 [01:30<03:40,  4.52it/s]Batch Loss: 0.248, WC Loss: 0.001, CE Loss 1: 0.120, CE Loss 2:
0.127, Acc F1: 98.4, Acc F2: 96.9, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:00<01:39,  4.98it/s]Batch Loss: 0.025, WC Loss: 0.000, CE Loss 1: 0.013, CE Loss 2:
0.012, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 96.6
100%|██████████| 1500/1500 [04:27<00:00,  5.60it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.43it/s]
Number of labeled: 965
Pool size: 1017
Pseudo-label accuracy: 99.6
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.81it/s]
Target Ft Acc: 96.0, Target F1 Acc: 96.1, Target F2 Acc: 95.5, Source F1 Acc: 98.8
Phase 3
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:12,  3.47it/s]Batch Loss: 0.187, WC Loss: 0.001, CE Loss 1: 0.082, CE Loss 2:
0.104, Acc F1: 98.4, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 97.4
 33%|███       | 502/1500 [01:33<03:44,  4.44it/s]Batch Loss: 0.080, WC Loss: 0.000, CE Loss 1: 0.060, CE Loss 2:
0.020, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:07<01:48,  4.59it/s]Batch Loss: 0.034, WC Loss: 0.000, CE Loss 1: 0.010, CE Loss 2:
0.024, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [04:37<00:00,  5.41it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.00it/s]
Number of labeled: 1299
Pool size: 1356
Pseudo-label accuracy: 99.5
Evaluating...
100%|██████████| 16/16 [00:03<00:00,  5.12it/s]
Target Ft Acc: 96.5, Target F1 Acc: 96.4, Target F2 Acc: 96.2, Source F1 Acc: 99.0
```

```
Phase 4
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:12,  3.46it/s]Batch Loss: 0.070, WC Loss: 0.000, CE Loss 1: 0.018, CE Loss 2:
0.052, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:29<03:36,  4.62it/s]Batch Loss: 0.016, WC Loss: 0.001, CE Loss 1: 0.011, CE Loss 2:
0.005, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1001/1500 [02:58<01:47,  4.63it/s]Batch Loss: 0.025, WC Loss: 0.000, CE Loss 1: 0.023, CE Loss 2:
0.002, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:28<00:00,  5.58it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 12.18it/s]
Number of labeled: 1647
Pool size: 1696
Pseudo-label accuracy: 99.3
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.20it/s]
Target Ft Acc: 97.2, Target F1 Acc: 96.1, Target F2 Acc: 96.3, Source F1 Acc: 98.9
Phase 5
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:13,  3.46it/s]Batch Loss: 0.052, WC Loss: 0.001, CE Loss 1: 0.007, CE Loss 2:
0.044, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 99.2, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:35<03:47,  4.38it/s]Batch Loss: 0.012, WC Loss: 0.000, CE Loss 1: 0.004, CE Loss 2:
0.008, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 67%|██████    | 1001/1500 [03:14<02:55,  2.85it/s]Batch Loss: 0.072, WC Loss: 0.001, CE Loss 1: 0.048, CE Loss 2:
0.023, Acc F1: 96.9, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:46<00:00,  5.23it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 14.95it/s]
Number of labeled: 1990
Pool size: 2035
Pseudo-label accuracy: 99.3
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.20it/s]
Target Ft Acc: 97.5, Target F1 Acc: 96.0, Target F2 Acc: 96.5, Source F1 Acc: 99.0
```

```
Phase 6
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:23,  3.38it/s]Batch Loss: 0.011, WC Loss: 0.000, CE Loss 1: 0.009, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 501/1500 [01:35<05:47,  2.88it/s]Batch Loss: 0.022, WC Loss: 0.000, CE Loss 1: 0.007, CE Loss 2:
0.015, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 67%|██████    | 1002/1500 [03:07<01:50,  4.52it/s]Batch Loss: 0.014, WC Loss: 0.001, CE Loss 1: 0.006, CE Loss 2:
0.006, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:40<00:00,  5.35it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 11.86it/s]
Number of labeled: 2319
Pool size: 2374
Pseudo-label accuracy: 99.1
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.54it/s]
Target Ft Acc: 97.1, Target F1 Acc: 96.2, Target F2 Acc: 96.4, Source F1 Acc: 99.0
Phase 7
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:11,  3.47it/s]Batch Loss: 0.036, WC Loss: 0.000, CE Loss 1: 0.014, CE Loss 2:
0.022, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:33<04:59,  3.33it/s]Batch Loss: 0.006, WC Loss: 0.000, CE Loss 1: 0.004, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1002/1500 [03:06<01:50,  4.52it/s]Batch Loss: 0.059, WC Loss: 0.000, CE Loss 1: 0.021, CE Loss 2:
0.037, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [04:37<00:00,  5.40it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 13.12it/s]
Number of labeled: 2666
Pool size: 2713
Pseudo-label accuracy: 99.3
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.03it/s]
Target Ft Acc: 97.0, Target F1 Acc: 96.4, Target F2 Acc: 96.9, Source F1 Acc: 99.2
```

```
Phase 8
Preparing dataloaders...
  0%|          | 1/1500 [00:00<14:14,  1.75it/s]Batch Loss: 0.061, WC Loss: 0.001, CE Loss 1: 0.020, CE Loss 2:
0.040, Acc F1: 98.4, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:37<03:42,  4.49it/s]Batch Loss: 0.060, WC Loss: 0.000, CE Loss 1: 0.048, CE Loss 2:
0.011, Acc F1: 96.9, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:11<02:38,  3.14it/s]Batch Loss: 0.011, WC Loss: 0.002, CE Loss 1: 0.007, CE Loss 2:
0.003, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 100.0
100%|██████████| 1500/1500 [04:44<00:00,  5.27it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.42it/s]
Number of labeled: 3006
Pool size: 3052
Pseudo-label accuracy: 99.3
Evaluating...
100%|██████████| 16/16 [00:03<00:00,  5.24it/s]
Target Ft Acc: 97.5, Target F1 Acc: 96.5, Target F2 Acc: 96.8, Source F1 Acc: 99.0
Phase 9
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:32,  3.31it/s]Batch Loss: 0.002, WC Loss: 0.000, CE Loss 1: 0.001, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 501/1500 [01:33<05:56,  2.80it/s]Batch Loss: 0.008, WC Loss: 0.001, CE Loss 1: 0.004, CE Loss 2:
0.003, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:08<01:48,  4.57it/s]Batch Loss: 0.014, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.011, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [04:39<00:00,  5.36it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 11.08it/s]
Number of labeled: 3347
Pool size: 3392
Pseudo-label accuracy: 99.3
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.05it/s]
Target Ft Acc: 97.4, Target F1 Acc: 96.8, Target F2 Acc: 96.9, Source F1 Acc: 99.0
```

```
Phase 10
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:03,  3.53it/s]Batch Loss: 0.002, WC Loss: 0.000, CE Loss 1: 0.001, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 33%|███       | 502/1500 [01:36<03:40,  4.53it/s]Batch Loss: 0.011, WC Loss: 0.001, CE Loss 1: 0.003, CE Loss 2:
0.007, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:07<01:39,  5.00it/s]Batch Loss: 0.024, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.022, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:39<00:00,  5.36it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.34it/s]
Number of labeled: 3691
Pool size: 3731
Pseudo-label accuracy: 99.3
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  5.51it/s]
Target Ft Acc: 97.4, Target F1 Acc: 96.8, Target F2 Acc: 97.0, Source F1 Acc: 99.1
Phase 11
Preparing dataloaders...
  0%|          | 1/1500 [00:00<18:00,  1.39it/s]Batch Loss: 0.004, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 502/1500 [01:34<03:45,  4.42it/s]Batch Loss: 0.016, WC Loss: 0.001, CE Loss 1: 0.013, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 67%|██████    | 1002/1500 [03:09<01:50,  4.50it/s]Batch Loss: 0.013, WC Loss: 0.001, CE Loss 1: 0.010, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:40<00:00,  5.34it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.34it/s]
Number of labeled: 4035
Pool size: 4070
Pseudo-label accuracy: 99.2
Evaluating...
100%|██████████| 16/16 [00:03<00:00,  5.00it/s]
Target Ft Acc: 97.6, Target F1 Acc: 96.9, Target F2 Acc: 96.9, Source F1 Acc: 99.1
```

```
Phase 12
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:27,  3.35it/s]Batch Loss: 0.060, WC Loss: 0.000, CE Loss 1: 0.006, CE Loss 2:
0.055, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 501/1500 [01:35<06:15,  2.66it/s]Batch Loss: 0.003, WC Loss: 0.000, CE Loss 1: 0.002, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:08<01:46,  4.66it/s]Batch Loss: 0.006, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.003, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:39<00:00,  5.36it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 11.03it/s]
Number of labeled: 4377
Pool size: 4409
Pseudo-label accuracy: 99.2
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.30it/s]
Target Ft Acc: 97.7, Target F1 Acc: 97.0, Target F2 Acc: 97.1, Source F1 Acc: 99.1
Phase 13
Preparing dataloaders...
  0%|          | 1/1500 [00:00<11:25,  2.19it/s]Batch Loss: 0.012, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.010, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 502/1500 [01:34<03:39,  4.54it/s]Batch Loss: 0.034, WC Loss: 0.001, CE Loss 1: 0.031, CE Loss 2:
0.002, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1001/1500 [03:11<03:07,  2.67it/s]Batch Loss: 0.002, WC Loss: 0.000, CE Loss 1: 0.000, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [04:41<00:00,  5.33it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.34it/s]
Number of labeled: 4711
Pool size: 4748
Pseudo-label accuracy: 99.1
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.72it/s]
Target Ft Acc: 97.6, Target F1 Acc: 97.0, Target F2 Acc: 96.9, Source F1 Acc: 99.1
```

```
Phase 14
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:23,  3.38it/s]Batch Loss: 0.004, WC Loss: 0.001, CE Loss 1: 0.003, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 501/1500 [01:36<06:07,  2.72it/s]Batch Loss: 0.088, WC Loss: 0.003, CE Loss 1: 0.084, CE Loss 2:
0.001, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:10<01:48,  4.59it/s]Batch Loss: 0.003, WC Loss: 0.000, CE Loss 1: 0.002, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [04:43<00:00,  5.30it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 12.64it/s]
Number of labeled: 5051
Pool size: 5088
Pseudo-label accuracy: 99.1
Evaluating...
100%|██████████| 16/16 [00:03<00:00,  5.16it/s]
Target Ft Acc: 97.3, Target F1 Acc: 96.7, Target F2 Acc: 97.0, Source F1 Acc: 98.9
Phase 15
Preparing dataloaders...
  0%|          | 1/1500 [00:00<17:05,  1.46it/s]Batch Loss: 0.003, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 502/1500 [01:33<03:40,  4.53it/s]Batch Loss: 0.003, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 100.0
 67%|██████    | 1001/1500 [03:07<02:52,  2.89it/s]Batch Loss: 0.010, WC Loss: 0.000, CE Loss 1: 0.003, CE Loss 2:
0.008, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
100%|██████████| 1500/1500 [04:37<00:00,  5.40it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.76it/s]
Number of labeled: 5367
Pool size: 5427
Pseudo-label accuracy: 99.2
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.40it/s]
Target Ft Acc: 97.4, Target F1 Acc: 96.9, Target F2 Acc: 96.8, Source F1 Acc: 99.0
```

```
Phase 16
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:08,  3.49it/s]Batch Loss: 0.084, WC Loss: 0.000, CE Loss 1: 0.028, CE Loss 2:
0.055, Acc F1: 98.4, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:34<03:41,  4.50it/s]Batch Loss: 0.049, WC Loss: 0.001, CE Loss 1: 0.010, CE Loss 2:
0.038, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:08<01:59,  4.17it/s]Batch Loss: 0.012, WC Loss: 0.001, CE Loss 1: 0.003, CE Loss 2:
0.008, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:40<00:00,  5.34it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 13.71it/s]
Number of labeled: 5725
Pool size: 5766
Pseudo-label accuracy: 99.2
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  5.51it/s]
Target Ft Acc: 97.6, Target F1 Acc: 97.2, Target F2 Acc: 97.0, Source F1 Acc: 99.1
Phase 17
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:15,  3.44it/s]Batch Loss: 0.088, WC Loss: 0.001, CE Loss 1: 0.070, CE Loss 2:
0.017, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:35<04:14,  3.92it/s]Batch Loss: 0.005, WC Loss: 0.002, CE Loss 1: 0.000, CE Loss 2:
0.003, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1002/1500 [03:09<01:49,  4.54it/s]Batch Loss: 0.008, WC Loss: 0.001, CE Loss 1: 0.003, CE Loss 2:
0.003, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 100.0
100%|██████████| 1500/1500 [04:41<00:00,  5.32it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.57it/s]
Number of labeled: 6077
Pool size: 6105
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.13it/s]
Target Ft Acc: 97.5, Target F1 Acc: 97.1, Target F2 Acc: 96.9, Source F1 Acc: 99.1
```

```
Phase 18
Preparing dataloaders...
  0%|          | 1/1500 [00:00<16:40,  1.50it/s]Batch Loss: 0.023, WC Loss: 0.000, CE Loss 1: 0.022, CE Loss 2:
0.001, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:35<03:47,  4.40it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.000, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:07<01:48,  4.59it/s]Batch Loss: 0.012, WC Loss: 0.002, CE Loss 1: 0.009, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:41<00:00,  5.33it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.24it/s]
Number of labeled: 6417
Pool size: 6444
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.24it/s]
Target Ft Acc: 97.3, Target F1 Acc: 96.9, Target F2 Acc: 97.0, Source F1 Acc: 99.1
Phase 19
Preparing dataloaders...
  0%|          | 1/1500 [00:00<18:20,  1.36it/s]Batch Loss: 0.007, WC Loss: 0.001, CE Loss 1: 0.004, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 501/1500 [01:34<04:07,  4.04it/s]Batch Loss: 0.025, WC Loss: 0.000, CE Loss 1: 0.022, CE Loss 2:
0.003, Acc F1: 98.4, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1001/1500 [03:10<03:04,  2.70it/s]Batch Loss: 0.005, WC Loss: 0.001, CE Loss 1: 0.005, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:42<00:00,  5.30it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.30it/s]
Number of labeled: 6758
Pool size: 6784
Pseudo-label accuracy: 99.1
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  5.94it/s]
Target Ft Acc: 97.6, Target F1 Acc: 97.3, Target F2 Acc: 96.9, Source F1 Acc: 99.0
```

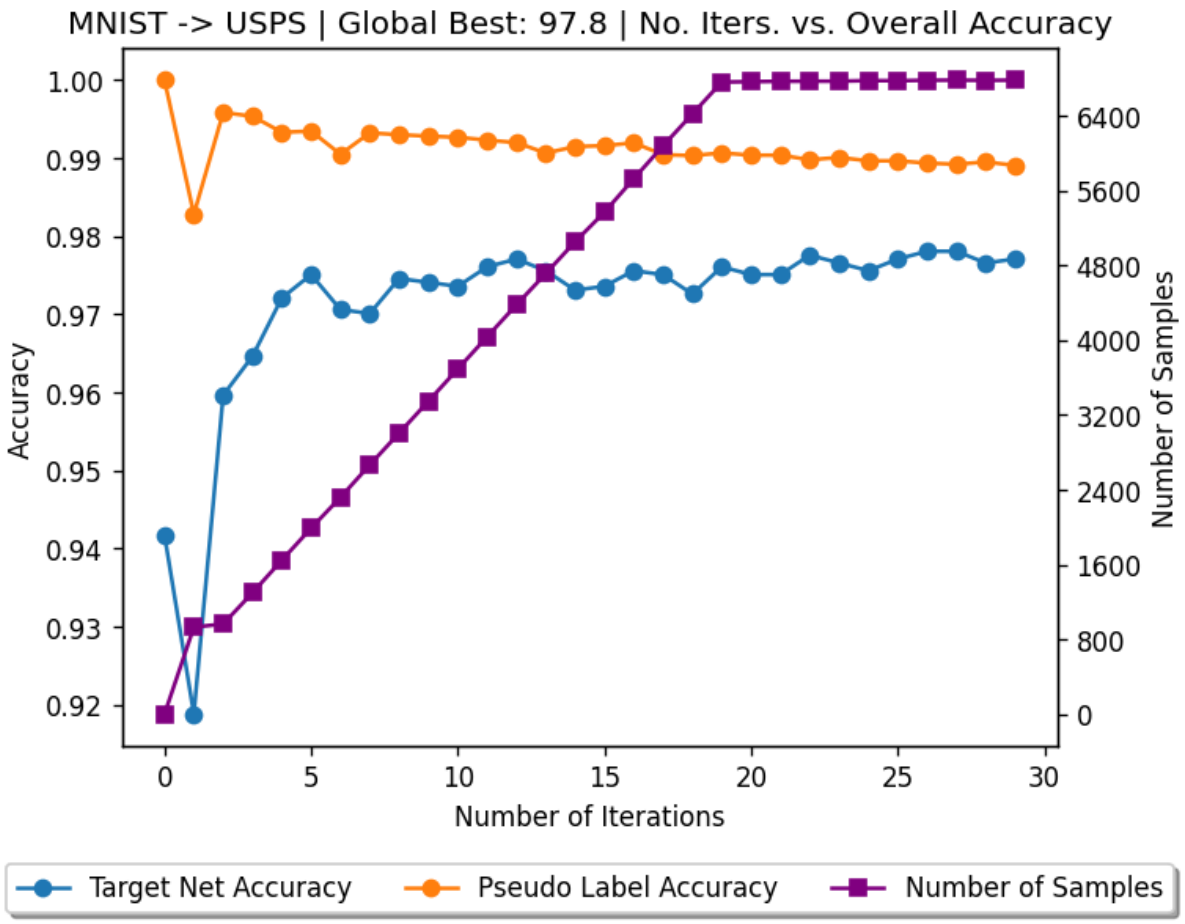
```
Phase 20
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:08,  3.50it/s]Batch Loss: 0.009, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.007, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:35<03:40,  4.54it/s]Batch Loss: 0.003, WC Loss: 0.000, CE Loss 1: 0.001, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 67%|██████    | 1002/1500 [03:10<01:50,  4.52it/s]Batch Loss: 0.010, WC Loss: 0.002, CE Loss 1: 0.006, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:43<00:00,  5.28it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 11.05it/s]
Number of labeled: 6762
Pool size: 7123
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.05it/s]
Target Ft Acc: 97.5, Target F1 Acc: 97.3, Target F2 Acc: 97.2, Source F1 Acc: 99.2
Phase 21
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:33,  3.30it/s]Batch Loss: 0.014, WC Loss: 0.002, CE Loss 1: 0.007, CE Loss 2:
0.006, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:37<03:37,  4.58it/s]Batch Loss: 0.009, WC Loss: 0.001, CE Loss 1: 0.006, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1001/1500 [03:11<02:57,  2.81it/s]Batch Loss: 0.016, WC Loss: 0.001, CE Loss 1: 0.003, CE Loss 2:
0.012, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:44<00:00,  5.28it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 11.29it/s]
Number of labeled: 6766
Pool size: 7462
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:03<00:00,  5.03it/s]
Target Ft Acc: 97.5, Target F1 Acc: 97.1, Target F2 Acc: 97.1, Source F1 Acc: 99.3
```

```
Phase 22
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:30,  3.33it/s]Batch Loss: 0.006, WC Loss: 0.003, CE Loss 1: 0.001, CE Loss 2:
0.003, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 501/1500 [01:35<06:11,  2.69it/s]Batch Loss: 0.001, WC Loss: 0.000, CE Loss 1: 0.001, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 67%|██████    | 1002/1500 [03:11<01:50,  4.51it/s]Batch Loss: 0.004, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:45<00:00,  5.26it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.36it/s]
Number of labeled: 6767
Pool size: 7801
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.04it/s]
Target Ft Acc: 97.8, Target F1 Acc: 97.1, Target F2 Acc: 97.0, Source F1 Acc: 99.4
Phase 23
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:27,  3.35it/s]Batch Loss: 0.008, WC Loss: 0.001, CE Loss 1: 0.005, CE Loss 2:
0.002, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 502/1500 [01:36<03:42,  4.48it/s]Batch Loss: 0.007, WC Loss: 0.000, CE Loss 1: 0.001, CE Loss 2:
0.006, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1002/1500 [03:11<01:51,  4.46it/s]Batch Loss: 0.010, WC Loss: 0.001, CE Loss 1: 0.009, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:44<00:00,  5.27it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 12.66it/s]
Number of labeled: 6767
Pool size: 8140
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  6.26it/s]
Target Ft Acc: 97.7, Target F1 Acc: 97.3, Target F2 Acc: 97.5, Source F1 Acc: 99.3
```

```
Phase 24
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:33,  3.30it/s]Batch Loss: 0.009, WC Loss: 0.000, CE Loss 1: 0.005, CE Loss 2:
0.005, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 502/1500 [01:35<04:01,  4.13it/s]Batch Loss: 0.007, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.004, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1002/1500 [03:10<01:51,  4.47it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.000, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:46<00:00,  5.24it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 14.55it/s]
Number of labeled: 6773
Pool size: 8480
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.22it/s]
Target Ft Acc: 97.6, Target F1 Acc: 97.2, Target F2 Acc: 97.2, Source F1 Acc: 99.4
Phase 25
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:08,  3.49it/s]Batch Loss: 0.030, WC Loss: 0.002, CE Loss 1: 0.002, CE Loss 2:
0.026, Acc F1: 100.0, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 99.1
 33%|███       | 502/1500 [01:37<03:44,  4.45it/s]Batch Loss: 0.003, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 67%|██████    | 1002/1500 [03:15<01:49,  4.55it/s]Batch Loss: 0.003, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:49<00:00,  5.18it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 12.17it/s]
Number of labeled: 6772
Pool size: 8819
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.05it/s]
Target Ft Acc: 97.7, Target F1 Acc: 97.0, Target F2 Acc: 97.1, Source F1 Acc: 99.2
```

```
Phase 26
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:13,  3.46it/s]Batch Loss: 0.001, WC Loss: 0.000, CE Loss 1: 0.000, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 97.4
 33%|███       | 502/1500 [01:39<03:48,  4.36it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1001/1500 [03:15<03:12,  2.59it/s]Batch Loss: 0.014, WC Loss: 0.002, CE Loss 1: 0.008, CE Loss 2:
0.004, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
100%|██████████| 1500/1500 [04:48<00:00,  5.19it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.52it/s]
Number of labeled: 6777
Pool size: 9158
Pseudo-label accuracy: 98.9
Evaluating...
100%|██████████| 16/16 [00:03<00:00,  5.08it/s]
Target Ft Acc: 97.8, Target F1 Acc: 97.1, Target F2 Acc: 97.2, Source F1 Acc: 99.3
Phase 27
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:26,  3.35it/s]Batch Loss: 0.463, WC Loss: 0.001, CE Loss 1: 0.371, CE Loss 2:
0.091, Acc F1: 98.4, Acc F2: 98.4, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:41<03:55,  4.23it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.000, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1002/1500 [03:19<02:17,  3.62it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.000, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:53<00:00,  5.12it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 13.65it/s]
Number of labeled: 6778
Pool size: 9497
Pseudo-label accuracy: 98.9
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  5.42it/s]
Target Ft Acc: 97.8, Target F1 Acc: 97.4, Target F2 Acc: 97.5, Source F1 Acc: 99.2
```

```
Phase 28
Preparing dataloaders...
  0%|          | 2/1500 [00:00<07:27,  3.35it/s]Batch Loss: 0.002, WC Loss: 0.000, CE Loss 1: 0.001, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:35<03:46,  4.40it/s]Batch Loss: 0.004, WC Loss: 0.002, CE Loss 1: 0.002, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 99.1
 67%|██████    | 1002/1500 [03:12<01:48,  4.60it/s]Batch Loss: 0.003, WC Loss: 0.001, CE Loss 1: 0.002, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:45<00:00,  5.25it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:03<00:00, 15.39it/s]
Number of labeled: 6776
Pool size: 9836
Pseudo-label accuracy: 99.0
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  7.21it/s]
Target Ft Acc: 97.7, Target F1 Acc: 97.3, Target F2 Acc: 97.2, Source F1 Acc: 99.1
Phase 29
Preparing dataloaders...
  0%|          | 1/1500 [00:00<18:13,  1.37it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.001, CE Loss 2:
0.000, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 33%|███       | 502/1500 [01:34<03:46,  4.41it/s]Batch Loss: 0.002, WC Loss: 0.001, CE Loss 1: 0.000, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
 67%|██████    | 1002/1500 [03:11<01:49,  4.57it/s]Batch Loss: 0.003, WC Loss: 0.002, CE Loss 1: 0.000, CE Loss 2:
0.001, Acc F1: 100.0, Acc F2: 100.0, Acc Ft: 100.0, Val Acc Ft: 98.3
100%|██████████| 1500/1500 [04:43<00:00,  5.29it/s]
Assigning Pseudo-labels...
100%|██████████| 53/53 [00:04<00:00, 12.84it/s]
Number of labeled: 6778
Pool size: 10176
Pseudo-label accuracy: 98.9
Evaluating...
100%|██████████| 16/16 [00:02<00:00,  5.39it/s]
Target Ft Acc: 97.7, Target F1 Acc: 97.2, Target F2 Acc: 97.5, Source F1 Acc: 99.1
```



3.5 Комментарій стосовно використання в медичних цілях.

Незважаючи на успішні результати, отримані в ході дослідження, слід зазначити, що масштабування моделі ATDA для вирішення медичних завдань, таких як діагностика пневмонії, виявилося ускладненим. Основною причиною є значне збільшення обсягу даних, необхідного для навчання моделі під час роботи з медичними зображеннями високої роздільної здатності.

Зокрема, при спробі адаптувати модель для розрізнення пневмонії, здорових легень і раку, виникли складнощі, пов'язані з обмеженими обчислювальними ресурсами. Вихідний набір даних, використаний для навчання моделі, містив 200 тисяч зображень розміром 16x16 пікселів, що становило всього 5 МБ. Однак, для досягнення хоча б 90% точності розпізнавання пневмонії на медичних зображеннях знадобився б набір даних розміром не менше 5 ГБ, що містить 400 зображень високої роздільної здатності.

Збільшення обсягу даних у 1000 разів призвело б до значного збільшення часу навчання моделі (до 7000 годин) і вимагало б значно більше оперативної пам'яті, ніж доступні 16 ГБ. В результаті, навчання моделі зупинялося вже через хвилину після початку через брак ресурсів.

Таким чином, незважаючи на перспективність моделі ATDA для вирішення завдань класифікації з адаптацією предметної області, її застосування в медичній діагностиці вимагає додаткових досліджень та оптимізації, пов'язаних з обробкою великих обсягів даних та використанням потужніших обчислювальних ресурсів.

```python
import h5py
import os
import cv2
from google.colab import drive

drive.mount('/content/drive')  # Монтуємо Google Drive

# Шлях до головної папки датасету (змініть на ваш шлях)
chest_xray_dir = "/content/drive/MyDrive/chest_xray"

# Отримуємо шлях до папки, де знаходиться chest_xray_dir (для збереження HDF5 файлу)
drive_root = os.path.dirname(chest_xray_dir)

# Створюємо файл HDF5 в тій самій папці, що й датасет
with h5py.File(os.path.join(drive_root, "chest_xray.h5"), "w") as f:
    # Цикл по розділах датасету (test, train, val)
    for split in ["test", "train", "val"]:
        # Створюємо групу для поточного розділу (навчальний, тестовий, валідаційний)
        split_group = f.create_group(split)
        # Цикл по класах (PNEUMONIA, NORMAL)
        for class_name in ["PNEUMONIA", "NORMAL"]:
            # Створюємо групу для поточного класу (пневмонія, норма)
            class_group = split_group.create_group(class_name)
            # Шлях до папки з зображеннями
            image_dir = os.path.join(chest_xray_dir, split, class_name)
            # Цикл по зображенням
            for i, filename in enumerate(os.listdir(image_dir)):
                # Читаємо зображення
                img = cv2.imread(os.path.join(image_dir, filename))

                # Перевірка, чи зображення було прочитано коректно
                if img is None:
                    print(f"Зображення не знайдено або не вдалося прочитати: {os.path.join(image_dir, filename)}")
                    continue  # Переходимо до наступного зображення

                # Зберігаємо зображення в HDF5 файл
                class_group.create_dataset(str(i), data=img)

print("Датасет перетворено в chest_xray.h5 (збережено в Google Drive)")
```

Код, що я використав для створення медичного датасету HDF5 формату.

4. ВИСНОВКИ

Дана дипломна робота була присвячена дослідженню та створенню моделі асиметричного тройного навчання (ATDA) для вирішення задачі класифікації з адаптацією предметної області. В рамках роботи було проаналізовано існуючі методи адаптації предметної області, розроблено архітектуру моделі ATDA, реалізовано алгоритм навчання, проведено експерименти та проаналізовано отримані результати.

Проведені експерименти показали, що модель ATDA здатна ефективно адаптуватися до нової предметної області, підвищуючи точність класифікації на цільових даних. Завдяки використанню трьох класифікаторів та механізму обміну псевдо-мітками, модель змогла вивчити спільні ознаки, інваріантні до розбіжностей в розподілі даних між вихідною та цільовою областями. Це дозволяє моделі ATDA досягати високої точності класифікації навіть у випадках, коли кількість розмічених даних з цільової області є обмеженою.

Порівняння з іншими методами адаптації предметної області, такими як методи на основі переважування даних, навчання ознак, або навчання з учителем, продемонструвало, що ATDA має ряд переваг. Зокрема, ATDA не вимагає збору та розмітки великої кількості даних з цільової області, що є особливо важливим у випадках, коли збір та розмітка даних є дорогими або трудомісткими процесами. Також ATDA дозволяє використовувати інформацію з нерозмічених даних з цільової області, що сприяє покращенню точності моделі.

В рамках роботи було досліджено різні конфігурації моделі ATDA, включаючи вибір типу нейронної мережі, кількості шарів, типу активаційних функцій, алгоритму оптимізації, значення швидкості навчання та методи регуляризації. Результати експериментів показали, що вибір оптимальної конфігурації залежить від специфіки задачі та даних. Проте, в цілому, модель ATDA виявилася досить стійкою до змін у конфігурації, що свідчить про її універсальність та придатність для вирішення різних задач класифікації з адаптацією предметної області.

Одним з ключових аспектів успіху ATDA є якість псевдо-міток, які генеруються моделлю. В роботі було досліджено різні методи оцінки якості псевдо-міток, такі як аналіз розподілу ймовірностей, порівняння з істинними мітками, або використання методів консенсусу. Результати показали, що вибір методу оцінки якості псевдо-міток також залежить від специфіки

задачі та даних. Проте, важливо використовувати надійні методи оцінки, щоб гарантувати, що псевдо-мітки є достатньо точними для ефективного навчання моделі.

Дослідження моделі ATDA дозволило зробити ряд висновків щодо сильних та слабких сторін методу. До сильних сторін можна віднести:

- **Зменшення потреби в розмічених даних**: ATDA дозволяє використовувати нерозмічені дані з цільової області, що є особливо важливим у випадках, коли збір та розмітка даних є дорогими або трудомісткими.
- **Підвищення точності класифікації**: ATDA дозволяє моделі краще адаптуватися до нової предметної області, що призводить до підвищення точності класифікації на цільових даних.
- **Універсальність**: ATDA може бути застосований до різних задач класифікації з адаптацією предметної області.
- **Ефективність**: ATDA демонструє високу ефективність порівняно з іншими методами адаптації предметної області.

До слабких сторін методу можна віднести:

- **Складність налаштування**: ATDA має кілька гіперпараметрів, які потрібно налаштувати для досягнення оптимальної продуктивності.
- **Чутливість до якості псевдо-міток**: Точність моделі ATDA залежить від якості псевдо-міток, тому важливо використовувати надійні методи оцінки та вибору псевдо-міток.

Подальші дослідження моделі ATDA можуть бути спрямовані на:

- Розробку більш ефективних методів оцінки та вибору псевдо-міток.
- Дослідження впливу різних архітектур моделі на її продуктивність.
- Адаптацію моделі ATDA до інших задач машинного навчання, таких як регресія або кластеризація.
- Інтеграцію моделі ATDA з іншими методами адаптації предметної області, такими як методи ансамблевого навчання або активного навчання.
- Дослідження можливості застосування ATDA до задач з кількома вихідними або цільовими доменами.

Загалом, модель асиметричного тройного навчання (ATDA) є перспективним підходом до вирішення задачі класифікації з адаптацією предметної області. Вона дозволяє досягти високої точності класифікації на цільових даних, не потребуючи при цьому розмічених даних з цільової області.

Подальший розвиток та вдосконалення моделі ATDA сприятиме розширенню можливостей застосування технологій машинного навчання в реальних умовах.

5. ВИКОРИСТАНІ ДЖЕРЕЛА

French, G. Mackiewicz, M Fisher, M. September 25, 2018 // Self-ensembling for visual domain adaptation

Geoffrey French Michal Mackiewicz Mark Henry Fisher June 2017 // Self-ensembling for domain adaptation

Domain Jongwon Choi, Youngjoon Choi,∗ Jihoon Kim, Jinyeop Chang, Ilhwan Kwon, Youngjune Gwon, Seungjai Min // Visual Domain Adaptation by Consensus-Based Transfer to Intermediate

Asymmetric Tri-training for Unsupervised Domain Adaptation - https://proceedings.mlr.press/v70/saito17a/saito17a-supp.pdf