ДОДАТОК А

Вихідний код для експериментальних досліджень

```
import torch
import torch.nn as nn
from torch.autograd import Variable
import numpy as np

import torch


import torch.optim as optim


from spotlight.helpers import _repr_model
from spotlight.factorization._components import _predict_process_ids
from spotlight.factorization.representations import BilinearNet


class ConvNet (nn.Module):
    def __init__(self, input_dim, output_dim, stride=1, padding=0):
        super(cnnBlock, self).__init__()
        self.conv1 = nn.Conv2d(input_dim, output_dim, kernel_size=1)
        self.conv2 = nn.Conv2d(output_dim, output_dim, kernel_size=3,
stride=1)
        self.conv3 = nn.Conv2d(output_dim, output_dim, kernel_size=3,
stride=1)


        self.relu = nn.ReLU()
        self.bn1 = nn.BatchNorm2d(output_dim)
        self.bn2 = nn.BatchNorm2d(output_dim)
        self.bn3 = nn.BatchNorm2d(output_dim)



    def forward(self, x):
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.relu(self.bn2(self.conv2(out)))
        out = self.relu(self.bn3(self.conv3(out)))


        return out
```

```
model = ConvNet()


criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
def train(model, device, train_loader, optimizer, criterion, epoch,
steps_per_epoch=N):
  model.train()


  train_loss = 0
  train_total = 0
  train_correct = 0


  for batch_idx, (data, target) in enumerate(train_loader, start=0):


    data, target = data.to(device), target.to(device)


    optimizer.zero_grad()


    output = model(data)


    loss = criterion(output, target)
    train_loss += loss.item()


    scores, predictions = torch.max(output.data, 1)
    train_total += target.size(0)
    train_correct += int(sum(predictions == target))


    optimizer.zero_grad()


    loss.backward()


    optimizer.step()
```

```
  acc = round((train_correct / train_total) * 100, 2)

  print('Epoch [{}], Loss: {}, Accuracy: {}'.format(epoch,
train_loss/train_total, acc), end='')


  wandb.log({'Train Loss': train_loss/train_total, 'Train Accuracy': acc})


output_batch = model(train_batch)

loss = loss_fn(output_batch, labels_batch)


optimizer.zero_grad()


loss.backward()


optimizer.step()




import numpy as np



import torch

import torch.nn.functional as F

import random



activation_getter = {'iden': lambda x: x, 'relu': F.relu, 'tanh':
torch.tanh, 'sigm': torch.sigmoid}



def gpu(tensor, gpu=False):


    if gpu:

        return tensor.cuda()

    else:

        return tensor
```

```
def cpu(tensor):

    if tensor.is_cuda:
        return tensor.cpu()
    else:
        return tensor


def minibatch(*tensors, **kwargs):

    batch_size = kwargs.get('batch_size', 128)

    if len(tensors) == 1:
        tensor = tensors[0]
        for i in range(0, len(tensor), batch_size):
            yield tensor[i:i + batch_size]
    else:
        for i in range(0, len(tensors[0]), batch_size):
            yield tuple(x[i:i + batch_size] for x in tensors)


def shuffle(*arrays, **kwargs):

    require_indices = kwargs.get('indices', False)

    if len(set(len(x) for x in arrays)) != 1:
        raise ValueError('All inputs to shuffle must have '
                         'the same length.')

    shuffle_indices = np.arange(len(arrays[0]))
    np.random.shuffle(shuffle_indices)

    if len(arrays) == 1:
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	83 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Сухомлінова Ю. І.			Дослідження та використання методів побудови систем послідовних рекомендацій	Шифр групи	Код напр./спец.
Перевірив		Рябова Н. В.				СШІм-19-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					