Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Классификация с помощью автоэнкодеров»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Литвинюк Т. В.

**Проверил:**

Туз И. С.

Брест 2024

**Цель:** использовать сверточные автоэнкодеры для классификации изображений.

**Ход работы**

**Вариант 7**

****

import torchvision

from nn import Conv2D, ReLU, MeanSquaredErrorLoss

from transforms import PIL2numpy, Normalize, OneHot

import numpy as np

from tqdm import tqdm

class Autoencoder:

def \_\_init\_\_(self):

self.layers = [

Conv2D((1, 28, 28), 3, 1),

ReLU(),

Conv2D((1, 26, 26), 3, 1, transposed=True),

ReLU()

]

def \_\_call\_\_(self, x):

for layer in self.layers:

x = layer(x)

return x

def backward(self, x, learning\_rate=0.01):

for layer in self.layers[::-1]:

if isinstance(layer, Conv2D):

x = layer.backward(x, learning\_rate)

else:

x = layer.backward(x)

def get\_class\_dict(train=True):

transforms = torchvision.transforms.Compose([

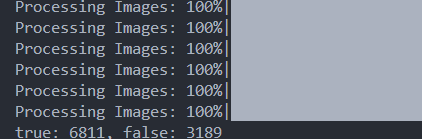
PIL2numpy(),

Normalize(),

])

target\_transform = torchvision.transforms.Compose([

OneHot()

 ])

# dataset = torchvision.datasets.MNIST(

dataset = torchvision.datasets.FashionMNIST(

root='.',

train=train,

download=True,

transform=transforms,

target\_transform=target\_transform

)

class\_dict = {}

for \_, target in dataset:

target = np.argmax(target).item()

if target not in class\_dict:

class\_dict[target] = []

class\_dict[target].append(\_)

return class\_dict

def train\_loop(dataset, model, criterion, learning\_rate):

for idx, image in enumerate(tqdm(dataset, desc="Processing Images")):

pred = model([image])

loss = criterion(image, pred)

x = criterion.backward()

model.backward(x, learning\_rate)

def main(num\_epochs=2, learning\_rate=0.01):

train\_class\_dict = get\_class\_dict(train=True)

models = [Autoencoder() for \_ in range(len(train\_class\_dict.keys()))]

criterion = MeanSquaredErrorLoss()

for model, cls in zip(models, sorted(train\_class\_dict.keys())):

for epoch in range(num\_epochs):

train\_loop(train\_class\_dict[cls], model, criterion, learning\_rate)

test\_class\_dict = get\_class\_dict(train=False)

true = 0

false = 0

for cls in sorted(test\_class\_dict.keys()):

for image in test\_class\_dict[cls]:

loss = []

for idx, model in enumerate(models):

pred = model([image])

loss.append(criterion(image, pred).sum())

if np.argmin(loss) == cls:

true += 1

else:

false += 1

print(f"true: {true}, false: {false}")

if \_\_name\_\_ == '\_\_main\_\_':

main(num\_epochs=1, learning\_rate=0.3)

**Вывод:** в ходе лабораторной работы я реализовал сверточный автоенкодер.