МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 4

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил:

Туз И. С.

Брест – 2024

**Ход работы**

**Задание:** в рамках данной работы необходимо реализовать и обучить свёрточную нейронную сеть для классификации изображений.

class Cnn:

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

self.conv1 = Conv2d(1, 2, 3, 1)

self.conv2 = Conv2d(2, 5, 2, 2, padding=1)

self.max\_pool = Maxpool2d(2, 2, padding=1)

self.fc1 = Linear(320, 100)

self.fc2 = Linear(100, 10)

self.flatten = Flatten()

self.relu = ReLU()

self.sigmoid1 = Sigmoid()

self.sigmoid2 = Sigmoid()

self.softmax = Softmax()

def load\_weights(self, load\_path):

self.conv1.load\_weights('conv\_w\_1', 'conv\_b\_1', load\_path)

self.conv2.load\_weights('conv\_w\_2', 'conv\_b\_2', load\_path)

self.fc1.load\_weights('fc\_w\_1', 'fc\_b\_1', load\_path)

self.fc2.load\_weights('fc\_w\_2', 'fc\_b\_2', load\_path)

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

x = self.conv1(x)

x = self.relu(x)

x = self.max\_pool(x)

x = self.conv2(x)

x = self.sigmoid1(x)

x = self.flatten.matrices2vector(x)

x = self.fc1(x)

x = self.sigmoid2(x)

x = self.fc2(x)

x = self.softmax(x)

return x

def backprop(self, x, lr=0.01):

x = self.softmax.backprop(x)

x = self.fc2.backprop(x, lr)

x = self.sigmoid2.backprop(x)

x = self.fc1.backprop(x, lr)

x = self.flatten.vector2matrices(x)

x = self.sigmoid1.backprop(x)

x = self.conv2.backprop(x, lr)

x = self.max\_pool.backprop(x)

x = self.relu.backprop(x)

x = self.conv1.backprop(x, lr)

def train\_loop(dataset, model, criterion, print\_log\_freq, lr):

loss\_log = []

acc\_log = []

losses = []

accuracies = []

start\_time = time.time()

for idx, (image, target) in enumerate(dataset):

pred = model([image])

loss = criterion(target, pred)

x = criterion.backprop(target, pred)

model.backprop(x, lr=lr)

loss\_log.append(loss.sum())

acc\_log.append((pred.argmax() == target.argmax()).item())

if idx % print\_log\_freq == 0:

loss\_avg = sum(loss\_log[-print\_log\_freq:]) / print\_log\_freq

acc\_avg = sum(acc\_log[-print\_log\_freq:]) / print\_log\_freq

losses.append(loss\_avg)

accuracies.append(acc\_avg)

loop\_time = time.time() - start\_time

start\_time = time.time()

print(f'Train step {idx}, Loss: {loss\_avg:.5f}, '

f'Acc: {acc\_avg:.4f}, time: {loop\_time:.1f}')

Train step 0, Loss: 0.00019, Acc: 0.0400, time: 0.4

Train step 25, Loss: 0.02705, Acc: 1.0000, time: 11.2

Train step 50, Loss: 0.02875, Acc: 1.0000, time: 11.4

Train step 75, Loss: 0.05460, Acc: 0.9600, time: 11.4

Train step 100, Loss: 0.01264, Acc: 1.0000, time: 11.7

Train step 125, Loss: 0.01281, Acc: 1.0000, time: 11.5

Train step 150, Loss: 0.01436, Acc: 1.0000, time: 11.2

Train step 175, Loss: 0.00505, Acc: 1.0000, time: 12.5

