МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 3

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.

import numpy as np

import matplotlib.pyplot as plt

a, b, c, d = 0.3, 0.3, 0.07, 0.3

num\_time\_steps = 110

input\_size = 1

hidden\_size = 12

output\_size = 1

lr = 0.01

epochs = 50

start = 0

time\_steps = np.linspace(start, start + 200, num\_time\_steps)

data = a \* np.cos(b \* time\_steps) + c \* np.sin(d \* time\_steps)

class RNN:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

self.hidden\_size = hidden\_size

self.W\_xh = np.random.randn(hidden\_size, input\_size) \* 0.01

self.W\_hh = np.random.randn(hidden\_size, hidden\_size) \* 0.01

self.W\_hy = np.random.randn(output\_size, hidden\_size) \* 0.01

self.b\_h = np.random.randn(hidden\_size, 1) \* 0.01

self.b\_y = np.random.randn(output\_size, 1) \* 0.01

def forward(self, x, h\_prev):

h\_next = np.tanh(np.dot(self.W\_xh, x) + np.dot(self.W\_hh, h\_prev) + self.b\_h)

y = np.dot(self.W\_hy, h\_next) + self.b\_y

return y, h\_next

rnn = RNN(input\_size, hidden\_size, output\_size)

h\_prev = np.zeros((hidden\_size, 1))

errors = []

for epoch in range(epochs):

for iter in range(200):

x = data[:-1].reshape(num\_time\_steps - 1, 1)

y = data[1:].reshape(num\_time\_steps - 1, 1)

dW\_xh, dW\_hh, dW\_hy = np.zeros\_like(rnn.W\_xh), np.zeros\_like(rnn.W\_hh), np.zeros\_like(rnn.W\_hy)

db\_h, db\_y = np.zeros\_like(rnn.b\_h), np.zeros\_like(rnn.b\_y)

dh\_next = np.zeros\_like(h\_prev)

total\_error = 0

for i in range(num\_time\_steps - 2, -1, -1):

x\_t = x[i].reshape(1, 1)

y\_t = y[i].reshape(1, 1)

pred, h\_prev = rnn.forward(x\_t, h\_prev)

dy = pred - y\_t

total\_error += np.sum(dy \*\* 2)

dW\_hy += np.dot(dy, h\_prev.T)

db\_y += dy

dh = np.dot(rnn.W\_hy.T, dy) + dh\_next

dh\_raw = (1 - h\_prev \* h\_prev) \* dh

db\_h += dh\_raw

dW\_xh += np.dot(dh\_raw, x\_t.T)

dW\_hh += np.dot(dh\_raw, h\_prev.T)

dh\_next = np.dot(rnn.W\_hh.T, dh\_raw)

for dparam in [dW\_xh, dW\_hh, dW\_hy, db\_h, db\_y]:

np.clip(dparam, -7, 7, out=dparam)

rnn.W\_xh -= lr \* dW\_xh

rnn.W\_hh -= lr \* dW\_hh

rnn.W\_hy -= lr \* dW\_hy

rnn.b\_h -= lr \* db\_h

rnn.b\_y -= lr \* db\_y

errors.append(total\_error)

print(f"Epoch {epoch} \tError: {total\_error:.5f}")

x = data[:-1].reshape(num\_time\_steps - 1, 1)

preds = []

h\_prev = np.zeros((hidden\_size, 1))

for i in range(num\_time\_steps - 1):

x\_t = x[i].reshape(1, 1)

pred, h\_prev = rnn.forward(x\_t, h\_prev)

preds.append(pred.ravel()[0])

plt.plot(range(len(errors)), errors)

plt.xlabel('Iterations')

plt.ylabel('Total Error')

plt.title('Total Error over Training')

plt.show()

plt.scatter(time\_steps[:-1], x.ravel(), s=10, label='original')

plt.plot(time\_steps[:-1], x.ravel())

plt.scatter(time\_steps[1:], preds, label='predicted')

plt.legend()

plt.show()

Epoch 0 Error: 5.17795

Epoch 1 Error: 5.08382

Epoch 2 Error: 4.57153

Epoch 3 Error: 2.53252

Epoch 4 Error: 0.66637

Epoch 5 Error: 0.48998

Epoch 6 Error: 0.43700

Epoch 7 Error: 0.36091

Epoch 8 Error: 0.26448

Epoch 9 Error: 0.18000

Epoch 10 Error: 0.14907

Epoch 11 Error: 0.14649

Epoch 12 Error: 0.14661

Epoch 13 Error: 0.14662

Epoch 14 Error: 0.14653

Epoch 15 Error: 0.14645

Epoch 16 Error: 0.14638

Epoch 17 Error: 0.14631

Epoch 18 Error: 0.14623

Epoch 19 Error: 0.14616

Epoch 20 Error: 0.14608

Epoch 21 Error: 0.14600

Epoch 22 Error: 0.14593

Epoch 23 Error: 0.14585

Epoch 24 Error: 0.14577

Epoch 25 Error: 0.14568

Epoch 26 Error: 0.14560

Epoch 27 Error: 0.14551

Epoch 28 Error: 0.14542

Epoch 29 Error: 0.14533

Epoch 30 Error: 0.14524

Epoch 31 Error: 0.14514

Epoch 32 Error: 0.14504

Epoch 33 Error: 0.14493

Epoch 34 Error: 0.14483

Epoch 35 Error: 0.14471

Epoch 36 Error: 0.14460

Epoch 37 Error: 0.14448

Epoch 38 Error: 0.14435

Epoch 39 Error: 0.14422

Epoch 40 Error: 0.14408

Epoch 41 Error: 0.14393

Epoch 42 Error: 0.14378

Epoch 43 Error: 0.14363

Epoch 44 Error: 0.14346

Epoch 45 Error: 0.14329

Epoch 46 Error: 0.14311

Epoch 47 Error: 0.14292

Epoch 48 Error: 0.14272

Epoch 49 Error: 0.14251



