МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 5

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил:

Туз И. С.

Брест – 2024

**Ход работы**

**Задание:** в рамках данной работы необходимо использовать сверточные автоэнкодеры для классификации изображений.

class Autoencoder:

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

self.layers = []

self.layers.append(Conv2D((1, 28, 28), 3, 1))

self.layers.append(ReLU())

self.layers.append(Conv2D((1, 26, 26), 3, 1, transposed=True))

self.layers.append(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 train\_loop(dataset, model, criterion, learning\_rate):

losses = []

for idx, image in enumerate(dataset):

pred = model([image])

loss = criterion(image, pred)

losses.append(loss)

x = criterion.backward()

model.backward(x, learning\_rate)

return losses

num\_epochs = 1

learning\_rate = 0.01

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

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

criterion = MeanSquaredErrorLoss()

correct\_guesses = [0] \* len(train\_class\_dict.keys())

incorrect\_guesses = [0] \* len(train\_class\_dict.keys())

all\_losses = []

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

for epoch in range(num\_epochs):

all\_losses.append(train\_loop(train\_class\_dict[cls], model, criterion, learning\_rate))

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

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

for image in test\_class\_dict[true\_cls]:

loss = []

for idx, model in enumerate(models):

pred = model([image])

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

predicted\_cls = np.argmin(loss)

if predicted\_cls == true\_cls:

correct\_guesses[true\_cls] += 1

else:

incorrect\_guesses[true\_cls] += 1

for cls, (correct, incorrect) in enumerate(zip(correct\_guesses, incorrect\_guesses)):

total = correct + incorrect

print(f"{cls}:\t{correct} / {total} \t{correct / total \* 100}%")

total\_correct = sum(correct\_guesses)

total\_images = sum(correct\_guesses) + sum(incorrect\_guesses)

total\_accuracy = total\_correct / total\_images \* 100

print(f"\nTotal Accuracy: {total\_correct} / {total\_images} \t{total\_accuracy:.2f}%")

|  |  |
| --- | --- |
|  | 0: 668 / 980 68.16326530612244%  1: 1112 / 1135 97.97356828193833%  2: 456 / 1032 44.18604651162791%  3: 595 / 1010 58.91089108910891%  4: 839 / 982 85.43788187372708%  5: 502 / 892 56.27802690582959%  6: 741 / 958 77.34864300626305%  7: 761 / 1028 74.0272373540856%  8: 561 / 974 57.59753593429158%  9: 661 / 1009 65.51040634291377%  Total Accuracy: 6896 / 10000 68.96% |