Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 4

по курсу «Основы глубокого обучения»

Выполнил:  
студент группы 22ВВИм1:

Милованов А.С.

Приняли:

Митрохин М.А  
Панков А.А.

Пенза 2023

# Порядок выполнения работы:

# Задание:

ЗАДАНИЕ 1: создать сверточную нейронную сеть, решающую задачу классификации животных на классы СОБАКА, КОШКА или ДИКОЕ по их изображению.

# Листинг программы:

# import torch

# import torch.nn as nn

# import torchvision

# import torchvision.transforms as transforms

# import numpy as np

# import matplotlib.pyplot as plt

# import itertools

# import torch.utils.data as data\_utils

# # In[51]:

# device = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')

# # In[52]:

# data\_transforms = transforms.Compose([

# transforms.Resize(256),

# transforms.CenterCrop(224),

# transforms.ToTensor(),

# transforms.Normalize(mean=[0.485, 0.456, 0.406],

# std=[0.229, 0.224, 0.225] )

# ])

# # In[72]

# batch\_size = 10

# #ТРЕЙНИНГ

# train\_dataset = torchvision.datasets.ImageFolder(root='./animals/train',

# transform=data\_transforms)

# #СЛАЙСИНГ (через random\_split)

# len\_sliced = 2000

# tr\_sliced = torch.utils.data.random\_split(train\_dataset, [len\_sliced, len(train\_dataset)-len\_sliced])[0]

# train\_loader = torch.utils.data.DataLoader(tr\_sliced, batch\_size=batch\_size,

# shuffle=True, num\_workers=2)

# #ТЕСТ

# test\_dataset = torchvision.datasets.ImageFolder(root='./animals/val',

# transform=data\_transforms)

# len\_sliced = 1000

# test\_sliced = torch.utils.data.random\_split(test\_dataset, [len\_sliced, len(test\_dataset)-len\_sliced])[0]

# test\_loader = torch.utils.data.DataLoader(test\_sliced, batch\_size=batch\_size,

# shuffle=True, num\_workers=2)

# # In[73]:

# print(train\_dataset.classes)

# print(test\_dataset.classes)

# # In[74]:

# # Сохранение названия классов

# class\_names = train\_dataset.classes

# # In[75]:

# inputs, classes = next(iter(train\_loader))

# inputs.shape

# classes

# img = torchvision.utils.make\_grid(inputs, nrow = 3)

# img = img.numpy().transpose((1, 2, 0))

# plt.imshow(img)

# # In[76]:

# class CNN(torch.nn.Module):

# def \_\_init\_\_(self, num\_classes=3):

# super().\_\_init\_\_()

# self.model = torch.nn.Sequential(

# torch.nn.Conv2d(in\_channels = 3, out\_channels = 6, kernel\_size = 3, padding = 1),

# torch.nn.ReLU(),

# torch.nn.MaxPool2d(kernel\_size=2),

# 

# 

# torch.nn.Conv2d(in\_channels = 6, out\_channels = 6, kernel\_size = 3, padding = 1),

# torch.nn.ReLU(),

# torch.nn.MaxPool2d(kernel\_size=2),

# 

# 

# torch.nn.Conv2d(in\_channels = 6, out\_channels = 6, kernel\_size = 3, padding = 1),

# torch.nn.ReLU(),

# torch.nn.MaxPool2d(kernel\_size=2),

# 

# torch.nn.Flatten(),

# torch.nn.Linear(4704, 512),

# torch.nn.ReLU(),

# torch.nn.Linear(512, num\_classes)

# )

# 

# def forward(self, x):

# return self.model(x)

# # In[77]:

# #гиперпараметры

# num\_epochs = 2

# learning\_rate = 0.001

# weight\_decay = 0.01

# net = CNN().to(device)

# lossFn = nn.CrossEntropyLoss()

# criterion = torch.nn.CrossEntropyLoss()

# optimizer = torch.optim.SGD(net.parameters(), lr=0.01)

# 

# train\_loss\_list = []

# import time

# t = time.time()

# for epoch in range(num\_epochs):

# print(f'Epoch {epoch+1}/{num\_epochs}:', end = ' ')

# train\_loss = 0

# for i, (images, labels) in enumerate(train\_loader):

# images = images.to(device)

# labels = labels.to(device)

# 

# outputs = net(images)

# 

# loss = lossFn(outputs, labels)

# optimizer.zero\_grad()

# loss.backward()

# 

# optimizer.step()

# 

# if i%10==0:

# print('Эпоха ' + str(epoch) + ' из ' + str(num\_epochs) + ' Шаг ' +

# str(i) + ' Ошибка: ', loss.item())

# 

# print('Time:',time.time() - t)

# # In[78]:

# correct\_predictions = 0

# num\_test\_samples = len(test\_sliced)

# with torch.no\_grad():

# for images, labels in test\_loader:

# images = images.to(device)

# labels = labels.to(device)

# pred = net(images)

# \_, pred\_class = torch.max(pred.data, 1)

# correct\_predictions += (pred\_class == labels).sum().item()

# print('Точность модели: ' + str(100 \* correct\_predictions / num\_test\_samples) + '%')

# # In[80]:

# inputs, classes = next(iter(test\_loader))

# pred = net(inputs.to(device))

# \_, pred\_class = torch.max(pred.data, 1)

# for i,j in zip(inputs, pred\_class):

# img = i.numpy().transpose((1, 2, 0))

# mean = np.array([0.485, 0.456, 0.406])

# std = np.array([0.229, 0.224, 0.225])

# img = std \* img + mean

# img = np.clip(img, 0, 1)

# plt.imshow(img)

# plt.title(class\_names[j])

# plt.pause(2)

# # In[81]:

# torch.save(net.state\_dict(), 'net.ckpt')

# Результат работы программы:

# 

Рисунок 1 Процесс обучения с одним скрытым слоем и функцией активации sigmoid

# 

Рисунок 2 Процесс Обучения

# 

Рисунок 3 Точность модели

# 

Рисунок 4 Классификация

# Вывод:

В ходе выполнения лабораторной работы была реализована сверточная нейронная сеть решающая задачу классификации животных на классы СОБАКА, КОШКА или ДИКОЕ по их изображению.