НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Звіт до комп’ютерного практикуму 4 з дисципліни: “**Програмні засоби проектування та реалізації нейромережевих систем**”

**Виконав**

**ІП-01 Черпак А.В.**

**Перевірив:**

**Шимкович В.М.**

Київ – 2023

**Комп’ютерний практикум 4**

**Тема:** Згорткові нейронні мережі для розпізнавання зображень

**Завдання:**

Написати програму що реалізує згорткову нейронну мережу AlexNet для розпізнавання об’єктів з датасету ImageNet цифр.

**Виконання:**

Завантаження та обробка даних:

import tensorflow as tf  
from tensorflow import keras  
import matplotlib.pyplot as plt  
from tensorflow.python.data.ops.dataset\_ops import DatasetV1  
from numpy import ndarray  
  
  
def get\_data() -> tuple[DatasetV1, DatasetV1, DatasetV1]:  
 (train\_images, train\_labels), (test\_images, test\_labels) = keras.datasets.cifar10.load\_data()  
  
 validation\_images, validation\_labels = train\_images[:5000], train\_labels[:5000]  
 train\_images, train\_labels = train\_images[5000:], train\_labels[5000:]  
  
 train\_ds = tf.data.Dataset.from\_tensor\_slices((train\_images, train\_labels))  
 test\_ds = tf.data.Dataset.from\_tensor\_slices((test\_images, test\_labels))  
 validation\_ds = tf.data.Dataset.from\_tensor\_slices((validation\_images, validation\_labels))  
 return train\_ds, test\_ds, validation\_ds  
  
  
def visualize\_data(train\_ds: DatasetV1, classes: list[str]) -> None:  
 plt.figure(figsize=(20, 20))  
 for i, (image, label) in enumerate(train\_ds.take(25)):  
 ax = plt.subplot(5, 5, i + 1)  
 plt.imshow(image)  
 plt.title(classes[label.numpy()[0]])  
 plt.axis('off')  
  
 plt.show()  
  
  
def process\_images(image: ndarray, label: str) -> tuple[ndarray, str]:  
 image = tf.image.per\_image\_standardization(image)  
 image = tf.image.resize(image, (227, 227))  
 return image, label  
  
  
def get\_ds\_size(ds: DatasetV1) -> int:  
 return tf.data.experimental.cardinality(ds).numpy()  
  
  
def process\_ds(ds: DatasetV1) -> DatasetV1:  
 return ds.map(process\_images).shuffle(buffer\_size=get\_ds\_size(ds)).batch(batch\_size=32, drop\_remainder=True)

Побудова та компіляція моделі:

import tensorflow as tf  
from keras.layers import Conv2D, BatchNormalization, MaxPool2D, Flatten, Dense, Dropout  
from keras.models import Sequential  
  
  
def AlexNet(class\_number) -> Sequential:  
 model = Sequential([  
 Conv2D(filters=96, kernel\_size=(11, 11), strides=(4, 4),  
 activation='relu', input\_shape=(227, 227, 3)),  
 BatchNormalization(),  
 MaxPool2D(pool\_size=(3, 3), strides=(2, 2)),  
 Conv2D(filters=256, kernel\_size=(5, 5), strides=(1, 1),  
 activation='relu', padding='same'),  
 BatchNormalization(),  
 MaxPool2D(pool\_size=(3, 3), strides=(2, 2)),  
 Conv2D(filters=384, kernel\_size=(3, 3), strides=(1, 1),  
 activation='relu', padding='same'),  
 BatchNormalization(),  
 Conv2D(filters=384, kernel\_size=(3, 3), strides=(1, 1),  
 activation='relu', padding='same'),  
 BatchNormalization(),  
 Conv2D(filters=256, kernel\_size=(3, 3), strides=(1, 1),  
 activation='relu', padding='same'),  
 BatchNormalization(),  
 MaxPool2D(pool\_size=(3, 3), strides=(2, 2)),  
 Flatten(),  
 Dense(4096, activation='relu'),  
 Dropout(0.5),  
 Dense(4096, activation='relu'),  
 Dropout(0.5),  
 Dense(class\_number, activation='softmax')  
 ])  
  
 return model  
  
  
def compile\_model(model: Sequential):  
 model.compile(  
 loss='sparse\_categorical\_crossentropy',  
 optimizer=tf.optimizers.SGD(learning\_rate=0.001),  
 metrics=['accuracy']  
 )  
 model.summary()  
 return model

Тренування та тестування моделі:

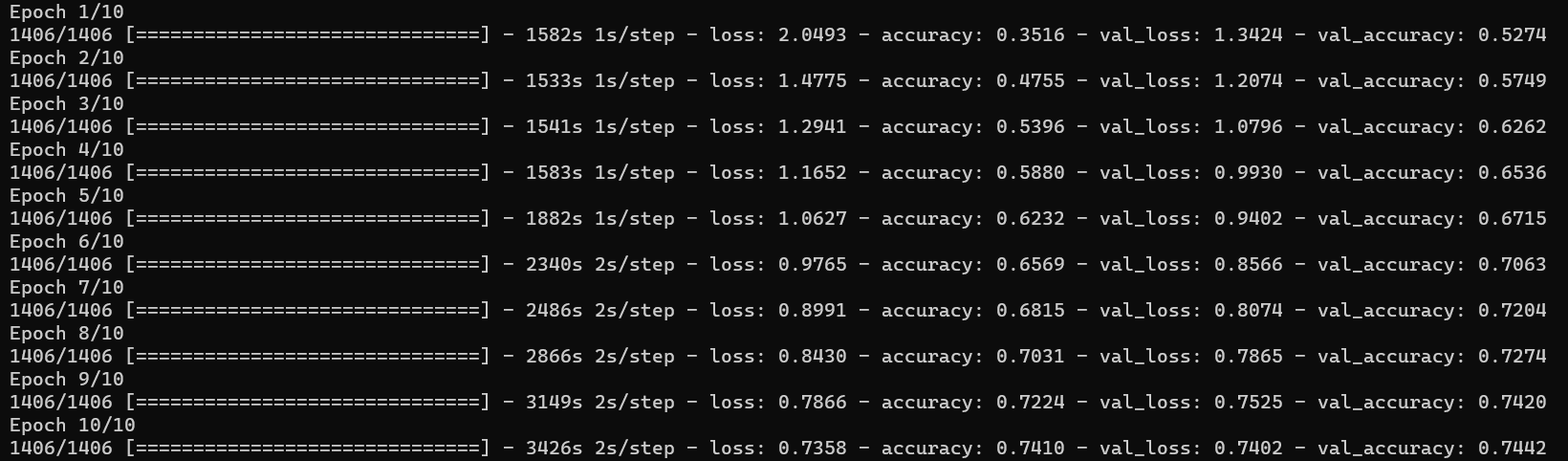
import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
from neural\_network\_types import FeedForwardBackprop, CascadeForwardBackprop, ElmanBackprop, NeuralNetworkModel  
from training\_data\_generation import data\_split, generate\_data  
  
  
def get\_learning\_rate(epochs, batch\_size):  
 initial\_learning\_rate = 10 \*\* (-3)  
 final\_learning\_rate = 10 \*\* (-7)  
 learning\_rate\_decay\_factor = (final\_learning\_rate / initial\_learning\_rate) \*\* (1 / epochs)  
 steps\_per\_epoch = int(len(train) / batch\_size)  
 return tf.keras.optimizers.schedules.ExponentialDecay(  
 initial\_learning\_rate=initial\_learning\_rate,  
 decay\_steps=steps\_per\_epoch,  
 decay\_rate=learning\_rate\_decay\_factor  
 )  
  
  
def train\_model(model\_type: type(NeuralNetworkModel), hidden\_neurons, train\_data, test\_data, epochs\_num, batch\_sz, learning\_rate):  
 model\_t = model\_type(hidden\_neurons)  
 model = model\_t.model  
 model.compile(loss='mean\_squared\_error', optimizer=tf.keras.optimizers.SGD(learning\_rate=learning\_rate))  
 model.summary()  
 return model.fit(np.reshape(train\_data[:, :2], (-1, 2)), train\_data[:, 2], epochs=epochs\_num, batch\_size=batch\_sz,  
 validation\_data=(np.reshape(test\_data[:, :2], (-1, 2)), test\_data[:, 2]), verbose=1).history, model\_t.nn\_model\_name  
  
  
def graph(train\_loss, val\_loss, name):  
 plt.title(name+'`s mean\_squared\_error')  
 plt.xlabel('Epochs')  
 plt.ylabel('Loss')  
 plt.plot(train\_loss, label='Train loss')  
 plt.plot(val\_loss, label='Validation loss')  
 plt.grid()  
 plt.legend()  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 data = generate\_data(1000000)  
 train, test = data\_split(data, 0.7)  
 epochs = 500  
 batch\_size = 10000  
 lr = get\_learning\_rate(epochs, batch\_size)  
 cases = [# (FeedForwardBackprop, [10]), (FeedForwardBackprop, [20]),  
 # (CascadeForwardBackprop, [20]), (CascadeForwardBackprop, [10, 10]),  
 # (ElmanBackprop, [15]),  
 (ElmanBackprop, [10, 10, 10])]  
  
 for (nn\_model, hidden\_neurons\_number) in cases:  
 log, name = train\_model(nn\_model, hidden\_neurons\_number, train, test, epochs, batch\_size, lr)  
 graph(log['loss'], log['val\_loss'], name)

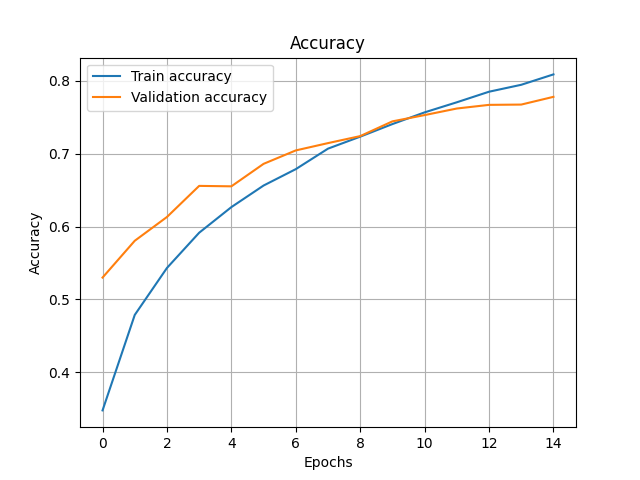
Візуалізація даних:

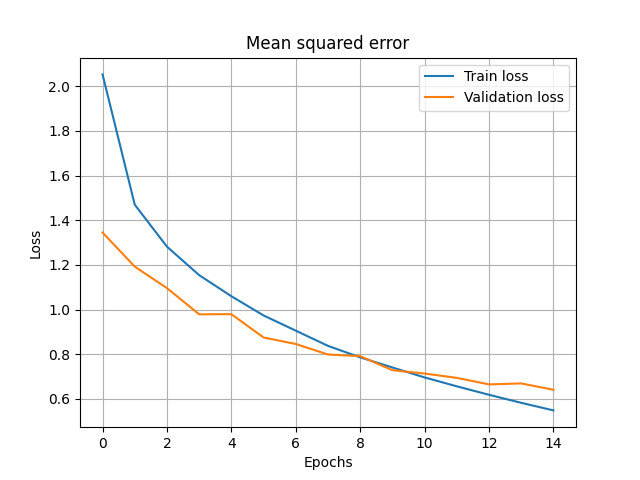




Результати тренування:

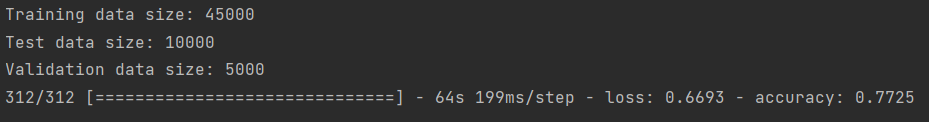






(Насправді було здійснено 15 епох, але останні 5, на жаль, заскрінити не вдалося.)

Результати тестування:



**Висновок:**

Під час виконання комп’ютерного практикуму ми реалізували згорткову нейромережу AlexNet для розпізнавання зображень з дата сету ImageNet. Після 15 епох тренувань ми отримали точність 77,25%, що насправді дуже навіть непогано. Щоправда, на її тренування було затрачено не менше доби.