A picture containing text

Description automatically generated**МІНІСТЕРСТВО ОСВІТИ і НАУКИ УКРАЇНИ**

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп’ютерних систем

**Лабораторна робота №2**  
**з дисципліни «Штучні нейронні мережі»**

Виконали:

студенти групи КВ-81

Поляков Є. А.

Ядуха Б. В.

Завдання

1. Сформувати навчальну вибірку для навчання згорткової нейронної мережі (ЗНМ), призначеної для розпізнавання емоцій на основі нейромережевого аналізу зображенні обличчя людини. Навчальна вибірка має складатись із навчальних прикладів, що представляють собою фотографії зображень обличчя людей з різними базовими емоціями. На фотографіях не має бути ключових точок. Обсяг навчальної вибірки повинен перевищувати 50 прикладів. Для формування можливо використовувати вільнодоступні бази даних, однак не менше ніж 10 прикладів мають бути сформовані самостійно.
2. За допомогою програмного забезпечення (ПЗ), розробленого при виконанні лаб.1, нанести ключові точки на зображення.
3. Розробити ПЗ, призначене для перетворення сформованих прикладів до виду придатному для подачі на вхід ЗНМ.

4. Розробити ПЗ для:

a. Реалізації ЗНМ1, призначеної для розпізнавання емоцій по ключовим точкам.

b. Реалізації ЗНМ2, призначеної для розпізнавання емоцій по зображенням обличчя (ключові точки не використовуються)

1. Провести комп’ютерні експерименти спрямовані на верифікацію розробленого ПЗ, що стосується ЗНМ1 та ЗНМ2. ПЗ для ЗНМ1 вважається верифікованим, якщо точність розпізнавання тренувальних прикладів >0,8, а точність розпізнавання тестових прикладів >0,7.

6. Оформити звіт в якому відобразити:

a. Прізвище, ініціали та номер групи виконавця.

b. Назву та завдання лабораторної роботи.

c. Інструкцію для використання розробленого ПЗ.

d. Скріншоти, що демонструють використання розробленого ПЗ.

e. Висновки.

Лабораторну роботу можливо виконувати в складі групи, що складається не більше ніж з 4 студентів.

Використаний датасет із минулої лабораторної роботи: <https://www.kaggle.com/datasets/manishshah120/facial-expression-recog-image-ver-of-fercdataset>

Програма для тренування нейронної мережі яка визначає емоції на основі ключових точок із вибірки минулого датасету:

**ImagesMarksAnalyzer.py**

import numpy as np

import tensorflow as tf

def load\_data(file\_name='../data/images.npy'):

    np\_image\_data = np.load(file\_name, allow\_pickle=True)

    np.random.shuffle(np\_image\_data)

    landmarks = []

    labels = []

    for (label, image, points) in np\_image\_data:

        landmarks.append(points.reshape(136, 1))

        labels.append(label)

    return np.array(landmarks), np.array(labels)

def split\_train\_test(landmarks, labels):

    landmarks\_20percent\_size = len(landmarks) // 5

    labels\_20percent\_size = len(labels) // 5

    test\_landmarks = landmarks[:landmarks\_20percent\_size]

    test\_labels = labels[:labels\_20percent\_size]

    train\_landmarks = landmarks[landmarks\_20percent\_size:]

    train\_labels = labels[labels\_20percent\_size:]

    return train\_landmarks, train\_labels, test\_landmarks, test\_labels

def create\_model():

    md = tf.keras.models.Sequential([

        tf.keras.layers.InputLayer(input\_shape=[136, 1]),

        tf.keras.layers.Conv1D(filters=64, kernel\_size=5),

        tf.keras.layers.Conv1D(filters=32, kernel\_size=3),

        tf.keras.layers.Flatten(),

        tf.keras.layers.Dense(16),

        tf.keras.layers.Dense(7, activation='softmax')

    ])

    md.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

               loss='sparse\_categorical\_crossentropy',

               metrics=['accuracy'])

    return md

np\_landmarks, np\_labels = load\_data()

np\_train\_landmarks, np\_train\_labels, np\_test\_landmarks, np\_test\_labels = split\_train\_test(np\_landmarks, np\_labels)

model = create\_model()

model.fit(np\_train\_landmarks, np\_train\_labels, batch\_size=64, epochs=1000)

print("Evaluating on test data:")

model.evaluate(np\_test\_landmarks, np\_test\_labels)

model.save('models/lab2/lab2.keras')

Результати роботи:

Graphical user interface, text

Description automatically generated

Точність на тренувальних даних 90%, на тестових: ~76%

Програма для тренування нейронної мережі яка визначає емоції на основі обличчя:

**ImagesFaceAnalyzer.py**:

import tensorflow as tf

import numpy as np

import cv2

import PIL.Image

import os

WITH\_EVALUATED\_LANDMARKS\_BASE\_PATH = 'data/with-evaluated-landmarks'

def pairwise(np\_array):

    """s -> (s0, s1), (s2, s3), (s4, s5), ..."""

    return np\_array.reshape(len(np\_array[0]) // 2, 2)

def load\_data(file\_name='data/images.npy'):

    image\_data = np.load(file\_name, allow\_pickle=True)

    np.random.shuffle(image\_data)

    return image\_data

def split\_data(image\_data):

    images = []

    labels = []

    for (label, image, points) in image\_data:

        images.append(image)

        labels.append(label)

    return np.array(images), np.array(labels)

def split\_train\_test(images, labels):

    images\_20percent\_size = len(images) // 5

    labels\_20percent\_size = len(labels) // 5

    test\_images = images[:images\_20percent\_size]

    test\_labels = labels[:labels\_20percent\_size]

    train\_images = images[images\_20percent\_size:]

    train\_labels = labels[labels\_20percent\_size:]

    return train\_images, train\_labels, test\_images, test\_labels

def create\_model():

    md = tf.keras.models.Sequential([

        # Copied from lab1

        tf.keras.layers.InputLayer(input\_shape=[48, 48, 1]),

        tf.keras.layers.Conv2D(filters=32, kernel\_size=[5, 5], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(filters=32, kernel\_size=[5, 5], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.MaxPool2D(pool\_size=[2, 2]),

        tf.keras.layers.Conv2D(filters=64, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(filters=64, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.MaxPool2D(pool\_size=[2, 2]),

        tf.keras.layers.Conv2D(filters=128, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(filters=128, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.MaxPool2D(pool\_size=[2, 2]),

        tf.keras.layers.Conv2D(filters=256, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(filters=256, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.MaxPool2D(pool\_size=[2, 2]),

        tf.keras.layers.Conv2D(filters=512, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(filters=512, kernel\_size=[3, 3], padding='same', use\_bias=False),

        tf.keras.layers.LeakyReLU(alpha=.1),

        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Flatten(),

        tf.keras.layers.Dense(units=512, activation='relu'),

        tf.keras.layers.Dropout(.1),

        tf.keras.layers.Dense(units=136),

        # Some addition to get a classifier

        tf.keras.layers.Dense(16),

        tf.keras.layers.Dense(7, activation='softmax')

    ])

    md.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-4),

               loss='sparse\_categorical\_crossentropy',

               metrics=['accuracy'])

    return md

np\_image\_data = load\_data()

np\_images, np\_labels = split\_data(np\_image\_data)

np\_train\_images, np\_train\_labels, np\_test\_images, np\_test\_labels = split\_train\_test(np\_images, np\_labels)

model = create\_model()

model.fit(np\_train\_images, np\_train\_labels, validation\_split=.1, batch\_size=64, epochs=100)

print("Evaluating on test data:")

model.evaluate(np\_test\_images, np\_test\_labels)

model.save('models/lab2/lab2\_image2class\_1.keras')

Результат роботи:

A screenshot of a computer

Description automatically generated with medium confidence

Точність на тренувальних даних 100%, на тестових: ~70%

**Висновок**

Під час виконання даної лабораторної роботи було розроблено 2 програми для тренування 2-х нейронних мереж для розпізнавання емоцій. Перша на основі ключових точок, друга на основі зображення обличчя людини. Перша видірка показала кращий результат, тому що друга була занадто складна і їй не вистачило навчальної вибірки, щоб навчитись. Продемонстровані результати роботи другої мережі – найкращий результат після кількох десятків спроб запусків.