ДОДАТОК А

Фрагменти програмного коду бібліотеки

Первинна обробка зображень

```
import json
import os

from PIL import Image, ImageEnhance
from scipy.misc import imread

IMAGES_PATH = 'D:/uni/cancer_data/'
RESIZED_IMAGES_PATH = 'D:/uni/cancer_data/resized/'

def resize_all():
    target_image_width = 85
    target_image_height = 64

    if not os.path.exists(RESIZED_IMAGES_PATH):
        os.mkdir(RESIZED_IMAGES_PATH)

    for file in os.listdir(IMAGES_PATH):
        filename, extension = os.path.splitext(file)
        if extension == '.jpg':
            im = Image.open(IMAGES_PATH + '\\' + file).convert('L')
            enhancer = ImageEnhance.Contrast(im)
            im = enhancer.enhance(6.0)
            imResize = im.resize((target_image_width, target_image_height),
Image.ANTIALIAS)
            imResize.save(RESIZED_IMAGES_PATH + '\\' + file, 'JPEG', quality=90)
    return target_image_width, target_image_height


def get_diagnosis(filename):
    with open(IMAGES_PATH + '\\' + filename + '.json', 'r') as json_file:
```

```
        data = json.loads(json_file.read())
        return 0 if data['meta']['clinical']['benign_malignant'] == 'benign' else 1


def get_image(file):
    return imread(RESIZED_IMAGES_PATH + '\\' + file + '.jpg', flatten=True)
```

Підготовка тренувальних та тестових даних

```
import os
from random import shuffle

import numpy as np

from cancer_recognition.resize_images import get_diagnosis, get_image,
IMAGES_PATH


def format_data():
    train_images = []
    train_labels = []
    test_images = []
    test_labels = []
    benign_images = []
    malignant_images = []
    data = {}
    for file in os.listdir(IMAGES_PATH):
        filename, extension = os.path.splitext(file)
        if extension == '.jpg':
            if get_diagnosis(filename) == 0:
                benign_images.append(filename)
            else:
                malignant_images.append(filename)
    print('Total benign: %s' % len(benign_images))
    print('Total malignant: %s' % len(malignant_images))
    shuffle(benign_images)
    shuffle(malignant_images)
```

```
    benign_images = benign_images[:len(malignant_images)]
    for image in benign_images:
        data[image] = 0
    for image in malignant_images:
        data[image] = 1
    images = list(data.keys())
    shuffle(images)

    for i in range(0, len(images)-1):
        if i % 5 == 0:
            test_images.append(get_image(images[i]))
            test_labels.append(data[images[i]])
        else:
            train_images.append(get_image(images[i]))
            train_labels.append(data[images[i]])

    print('Total train: %s' % len(train_images))
    print('Total train benign: %s' % train_labels.count(0))
    print('Total test: %s' % len(test_images))
    print('Total test benign: %s' % test_labels.count(0))

    return np.asarray(train_images) / 255.0, np.asarray(train_labels), \
           np.asarray(test_images) / 255.0, np.asarray(test_labels),
```

Створення та тренування нейронної мережі

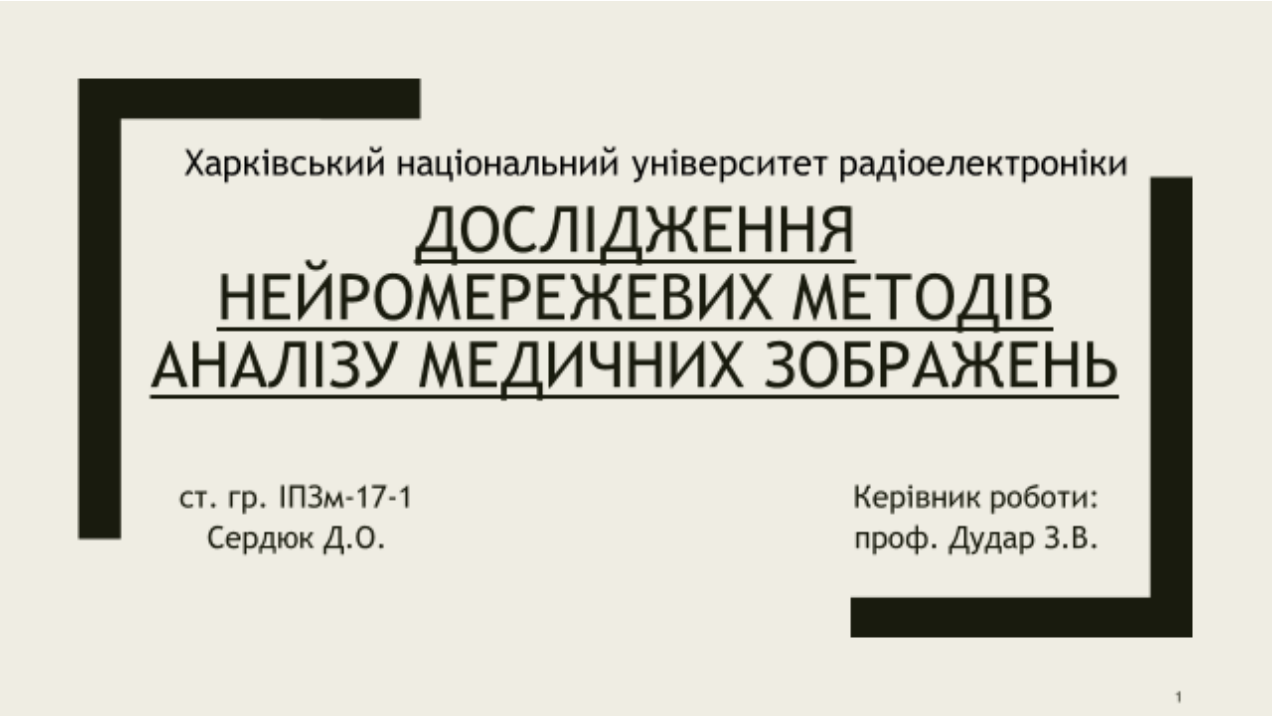
```
import numpy
import tensorflow as tf
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from keras.models import Sequential
from keras.optimizers import SGD
from keras.utils import to_categorical

from cancer_recognition.data_preprocessing import format_data

numpy.random.seed(41)
```

```
callbacks = [
    EarlyStopping(
        monitor='val_acc',
        patience=10,
        mode='max',
        verbose=1),
    ModelCheckpoint('./fm_cnn_BN.h5',
                    monitor='val_acc',
                    save_best_only=True,
                    mode='max',
                    verbose=1)
]


def create_and_train_convolutional_neural_network(width, height):
    (train_images, train_labels, test_images, test_labels) = format_data()
    print('Train data amount: ', train_images.shape)
    print('Test data amount: ', test_images.shape)

    train_labels = to_categorical(train_labels)
    test_labels = to_categorical(test_labels)
    train_images = train_images.reshape(len(train_images), width, height, 1)
    test_images = test_images.reshape(len(test_images), width, height, 1)

    model = Sequential([
        Conv2D(64, kernel_size=3, use_bias=False, padding='same', activation=tf.nn.relu,
               input_shape=(width, height, 1)),
        Dropout(0.25),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(32, kernel_size=3, activation=tf.nn.relu, padding='same', use_bias=False,
),
        Dropout(0.25),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(32, kernel_size=3, activation=tf.nn.relu, padding='same', use_bias=False,
),
```

```
    Dropout(0.25),

    Flatten(),
    Dense(64, activation=tf.nn.relu),
    Dropout(0.25),
    Dense(2, activation=tf.nn.softmax)
])
sgd = SGD(lr=0.01, decay=0.01 / 15, momentum=0.9)
model.compile(optimizer=sgd,
              loss='binary_crossentropy',
              metrics=['accuracy'])
history = model.fit(train_images,
                    train_labels,
                    epochs=50,
                    validation_data=(test_images, test_labels),
                    callbacks=callbacks)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
return model, history
```

ДОДАТОК Б

Слайди презентації



Переваги штучного інтелекту

- Постановка діагнозу на ранніх стадіях
- Зменшення вартості лікування для населення
- Покращення точності діагнозу
- В перспективі, повністю автоматизоване проведення анамнезу

3

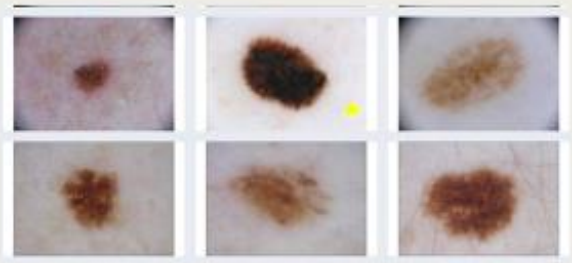
Актуальність дослідження

- Очікується, що у США близько 7230 людей помруть від меланоми за 2019 рік
- Частота діагнозів раку шкіри зростає протягом останніх 30 років
- За оцінками, п'ятирічна виживання пацієнтів, у яких рано виявлена меланома, становить близько 98% і лише 23 відсотки, коли захворювання метастазує в віддалені органи.
- Меланома є головною причиною смерті від раку у молодих жінок у віці 25-30 років

4

Вибрана база для навчання

- Класифіковані ураження шкіри, предоставлені Міжнародної спілки зображень шкіри (ISIC)
- Кожен приклад містить, власне, зображення та метадату, чи є ураження добро- чи злоякісним



5

Методи нейромереж

- Конволюційні нейромережі
- Попередня обробка зображень
- Налаштування нейромережі для отримання оптимальних результатів
- Оцінка результатів роботи отриманої моделі

6

Технології розробки

7

Процес розробки

- Попередня обробка зображень: переведення до мешного розширення (85x64 пікселів), переведення зображення до чорно-білого, збільшення контрасту
- Нормалізувати значення пікселів з [0, 255] до [0, 1]
- Позначити доброякісні утворення як 0, злоякісні як 1
- Розподілити даних на тренувальну та тестову вибірки
- Провести експеримент з первинною структурою мережі

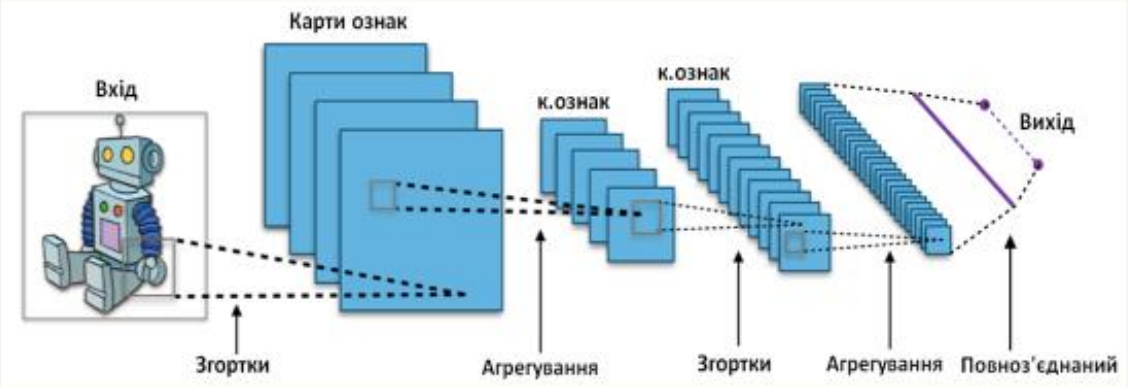
8

Розподіл даних

	Доброякісні	Злоякісні	Всього
Всього	908	479	2774
Тренувальний набір	375 (41.3%)	390 (81.4%)	765 (80%)
Тестовий набір	103 (11.3%)	89 (18.6%)	192 (20%)

9

Згорткова нейронна мережа



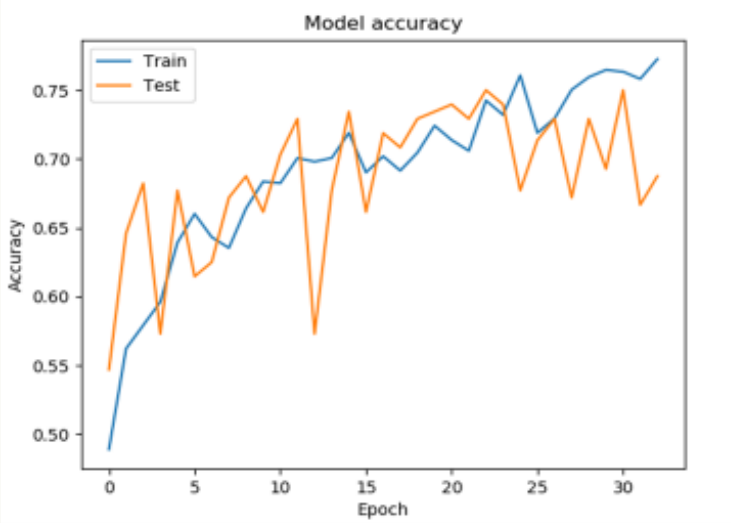
10

Результуюча структура мережі

Convolutional	Convolutional
Dropout (0.25)	Dropout (0.25)
MaxPooling	Flatten
Convolution	Dense (64)
Dropout (0.25)	Dropout (0.25)
MaxPooling	Dense (2,softmax)

11

Графік точності передбачень



12

Висновки

- Проведено дослідження методів нейромереж для вибраної предметної області
- Результуюча точність класифікації становить 75%, час тренування до оптимального результату – 10-15 хв
- На даному датасеті у якості регуляризації достатньо використовувати шар виключення, у той час як L1 та L2-регуляризації не показували значного покращення
- Більше ніж 3 згорткові шари у комбінації з шарами агрегування показують гірший результат, та збільшують час навчання

13

Дякую за увагу

14