МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

ЗВІТ

про виконання лабораторної роботи № 3

з курсу «Компютерне бачення»

Виконав:

студент групи КН-409

Гладун Ярослав

Перевірив:

Пелешко Дмитро

Львів - 2022

Тема: Класифікація зображень. Застосування нейромереж для пошуку подібних зображень.

Мета: набути практичних навиків у розв'язанні задачі пошуку подібних зображень на прикладі організації CNN класифікації.

Варіант 5

Завдання: Побудувати CNN на основі Inception-v3 для класифікації зображень на основі датасету fashion-mnist. Зробити налаштування моделі для досягнення необхідної точності. На базі Siamese networks побудувати систему для пошуку подібних зображень в датасеті fashion-mnist. Візуалізувати отримані результати t-SNE.

Код програми:

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import fashion_mnist
from sklearn.manifold import TSNE
from tensorflow.keras.layers import Flatten, Input, Dense, Dropout
from tensorflow.keras.models import Model
import matplotlib.pyplot as plt
import numpy as np
import numpy as np
import random
import tensorflow as tf


(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
train_images = train_images / 255.
test_images = test_images / 255.


def make_pairs(x, y):
    num_classes = max(y) + 1
    digit_indices = [np.where(y == i)[0] for i in range(num_classes)]

    pairs = []
    labels = []

    for idx1 in range(len(x)):
        x1 = x[idx1]
        label1 = y[idx1]
        idx2 = random.choice(digit_indices[label1])
```

```
        x2 = x[idx2]

        pairs += [[x1, x2]]
        labels += [1]

        label2 = random.randint(0, num_classes - 1)
        while label2 == label1:
            label2 = random.randint(0, num_classes - 1)

        idx2 = random.choice(digit_indices[label2])
        x2 = x[idx2]

        pairs += [[x1, x2]]
        labels += [0]

    return np.array(pairs), np.array(labels).astype('float32')

pairs_train, labels_train = make_pairs(train_images, train_labels)
pairs_test, labels_test = make_pairs(test_images, test_labels)

def euclidean_distance(vects):
    x, y = vects
    sum_square = tf.math.reduce_sum(tf.math.square(x - y), axis=1, keepdims=True)
    return tf.math.sqrt(tf.math.maximum(sum_square, 1e-7))

input = Input(shape=(28, 28))
_ = Flatten()(input)
_ = Dense(128, activation='relu')(_)
_ = Dropout(0.1)(_)
_ = Dense(128, activation='relu')(_)
_ = Dropout(0.1)(_)
_ = Dense(128, activation='relu')(_)
base_model = Model(inputs=input, outputs=_)

base_model.summary()

from tensorflow.keras.layers import Lambda
input_a = Input(shape=(28,28,), name='left_input')
vector_output_a = base_model(input_a)
input_b = Input(shape=(28,28,), name='right_input')
vector_output_b = base_model(input_b)
output = Lambda(euclidean_distance, name='output_layer')([vector_output_a, vector_output_b])
model = Model([input_a, input_b], output)
model.summary()
```

```
def contrastive_loss_with_margin(margin):
    def contrastive_loss(y_true, y_pred):
        square_pred = tf.math.square(y_pred)
        margin_square = tf.math.square(tf.math.maximum(margin - y_pred, 0))
        return tf.math.reduce_mean(y_true * square_pred + (1 - y_true) * margin_square)
    return contrastive_loss

rms = tf.keras.optimizers.RMSprop()
model.compile(loss=contrastive_loss_with_margin(margin=1), optimizer=rms)
history = model.fit([pairs_train[:,0], pairs_train[:,1]], labels_train, epochs=20, batch_size=128,
validation_data=([pairs_test[:,0], pairs_test[:,1]], labels_test))

plt.title('Loss')
plt.plot(history.history['loss'],label='loss')
plt.plot(history.history['val_' + 'loss'],label='val_' + 'loss')
plt.legend()

def display_images(left, right, predictions, labels, title, n):
    plt.figure(figsize=(17,3))
    plt.title(title)
    plt.yticks([])
    plt.xticks([])
    plt.grid(None)
    left = np.reshape(left, [n, 28, 28])
    left = np.swapaxes(left, 0, 1)
    left = np.reshape(left, [28, 28*n])
    plt.imshow(left)
    plt.figure(figsize=(17,3))
    plt.yticks([])
    plt.xticks([28*x+14 for x in range(n)], predictions)
    for i,t in enumerate(plt.gca().xaxis.get_ticklabels()):
        if predictions[i] > 0.5: t.set_color('red')
        if predictions[i] < 0.2: t.set_color('green')
    plt.grid(None)
    right = np.reshape(right, [n, 28, 28])
    right = np.swapaxes(right, 0, 1)
    right = np.reshape(right, [28, 28*n])
    plt.imshow(right)

predictions = np.squeeze(model.predict([pairs_test[:,0], pairs_test[:,1]]))
indexes = np.random.choice(len(predictions), size=10)

display_images(pairs_test[:, 0][indexes], pairs_test[:, 1][indexes], predictions[indexes], labels_test[indexes],
"clothes and their dissimilarity", 10)
```

Висновок

У ході виконання лабораторної роботи, я набув практичних навиків у розв’язанні задачі пошуку подібних зображень на прикладі організації CNN класифікації.