**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

Регусевич Теодор

*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 10**

**Завдання:** Побудувати CNN на основі ResNeXt-50 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.

################################################################################

class ResidualBlock(Model):

def \_\_init\_\_(self, channel\_in = 64, channel\_out = 256):

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

channel = channel\_out // 4

self.conv1 = Conv2D(channel, kernel\_size = (1, 1), padding = "same")

self.bn1 = BatchNormalization()

self.av1 = Activation(tf.nn.relu)

self.conv2 = Conv2D(channel, kernel\_size = (3, 3), padding = "same")

self.bn2 = BatchNormalization()

self.av2 = Activation(tf.nn.relu)

self.conv3 = Conv2D(channel\_out, kernel\_size = (1, 1), padding = "same")

self.bn3 = BatchNormalization()

self.shortcut = self.\_shortcut(channel\_in, channel\_out)

self.add = Add()

self.av3 = Activation(tf.nn.relu)

def call(self, x):

h = self.conv1(x)

h = self.bn1(h)

h = self.av1(h)

h = self.conv2(h)

h = self.bn2(h)

h = self.av2(h)

h = self.conv3(h)

h = self.bn3(h)

shortcut = self.shortcut(x)

h = self.add([h, shortcut])

y = self.av3(h)

return y

def \_shortcut(self, channel\_in, channel\_out):

if channel\_in == channel\_out:

return lambda x : x

else:

return self.\_projection(channel\_out)

def \_projection(self, channel\_out):

return Conv2D(channel\_out, kernel\_size = (1, 1), padding = "same")

class ResNet50(Model):

def \_\_init\_\_(self, input\_shape, output\_dim):

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

self.\_layers = [

# conv1

Conv2D(64, input\_shape = input\_shape, kernel\_size = (7, 7), strides=(2, 2), padding = "same"),

BatchNormalization(),

Activation(tf.nn.relu),

# conv2\_x

MaxPool2D(pool\_size = (3, 3), strides = (2, 2), padding = "same"),

ResidualBlock(64, 256),

[

ResidualBlock(256, 256) for \_ in range(2)

],

# conv3\_x

Conv2D(512, kernel\_size = (1, 1), strides=(2, 2)),

[

ResidualBlock(512, 512) for \_ in range(4)

],

# conv4\_x

Conv2D(1024, kernel\_size = (1, 1), strides=(2, 2)),

[

ResidualBlock(1024, 1024) for \_ in range(6)

],

# conv5\_x

Conv2D(2048, kernel\_size = (1, 1), strides=(2, 2)),

[

ResidualBlock(2048, 2048) for \_ in range(3)

],

# last part

GlobalAveragePooling2D(),

Dense(32)

# Dense(1000, activation = tf.nn.relu),

#Dense(output\_dim, activation = tf.nn.softmax)

]

def call(self, x):

for layer in self.\_layers:

if isinstance(layer, list):

for l in layer:

x = l(x)

else:

x = layer(x)

return x

model = ResNet50((28, 28, 1), 10)

model.build(input\_shape = (None, 28, 28, 1))

model.summary()

################################################################################

def siam\_gen(in\_groups, batch\_size = 32):

while True:

pv\_a, pv\_b, pv\_sim = gen\_random\_batch(train\_groups, batch\_size//2)

yield [pv\_a, pv\_b], pv\_sim

valid\_a, valid\_b, valid\_sim = gen\_random\_batch(test\_groups, 1024)

loss\_history = similarity\_model.fit(siam\_gen(train\_groups),

steps\_per\_epoch = 50,

validation\_data=([valid\_a, valid\_b],

valid\_sim),

callbacks=[early\_stopping, reduce\_lr],

epochs=10,

verbose=True)

**Висновок**

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