МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
кафедра систем штучного інтелекту

Лабораторна робота №3

на тему:

«Класифікація зображень. Застосування нейромереж для пошуку подібних зображень.»

з дисципліни

«Обробка зображень методами штучного інтелекту»

Виконав:
ст. групи КН-408
Романьчук Максим

ЛЬВІВ – 2022

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

Теоретичні відомості

Задачі пошуку подібних зображень передбачають наступні етапи:

1. Поділ зображень на класи.
2. Створення класифікатора (на основі статистичних, ансамблевих моделей чи нейронних мереж різних типів).
3. Порівняння результатів класифікації (бінарне або відносне).

Поділ зображень на класи залежить від тематики обраних зображень та величини вибірки. Чим більш різноманітна тематика зображень, тим більше потрібно класів для досягнення високої точності. А для малих вибірок слід прагнути малої кількості класів.

Складність класифікатора дуже залежить від розмірів та розмаїття вибірки, «складності» завдання та максимально досяжної точності. Складна модель може бути точнішою, вирішувати складніші завдання, але при недостатній вибірці легко перенавчається так і не досягнувши бажаної гнучкості.

Для порівняння результатів часто застосовуються «Сіамські» моделі, суть яких полягає в знаходженні кращої метрики знаходження подібності зображень. Зазвичай така модель «включає» в себе класифікатор, що оцінює належність зображення до класів і вже з результатами класифікатора від двох зображень визначає наскільки вони різні.

Хід роботи

Варіант 6. Побудувати CNN на основі ResNet-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.

1) Для початку було завантажено датасет fashion-mnist та згорткову нейронну мережу ResNet50 і натреновано останню.

```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train = np.array([skimage.transform.resize(image, (32, 32)) for image in np.expand_dims(x_train, -1)])[:10000]
x_test = np.array([skimage.transform.resize(image, (32, 32)) for image in np.expand_dims(x_test, -1)])
y_train = y_train.astype('int')[:10000]
y_test = y_test.astype('int')

train_groups = [x_train[np.where(y_train==i)[0]] for i in np.unique(y_train)]
test_groups = [x_test[np.where(y_test==i)[0]] for i in np.unique(y_train)]

print('train groups:', [x.shape[0] for x in train_groups])
print('test groups:', [x.shape[0] for x in test_groups])
```

✓ 11.3s

```
train groups: [942, 1027, 1016, 1019, 974, 989, 1021, 1022, 990, 1000]
test groups: [1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000]
```

```
%%time
model = ResNet50(classes = 10, weights = None, input_shape=(32, 32, 1))

trainable = False
for layer in model.layers:
    if layer.name == "conv5_block1_out":
        trainable = True
    layer.trainable = trainable

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs = 4, batch_size = 16, validation_split = 0.2, verbose = 1)
```

✓ 5m 40.1s

```
Epoch 1/4
500/500 [==============================] - 91s 178ms/step - loss: 1.3732 - accuracy: 0.6139 - val_loss: 2.3960 - val_accuracy: 0.1610
Epoch 2/4
500/500 [==============================] - 83s 166ms/step - loss: 0.9636 - accuracy: 0.6926 - val_loss: 1.2229 - val_accuracy: 0.6030
Epoch 3/4
500/500 [==============================] - 83s 165ms/step - loss: 0.8250 - accuracy: 0.7335 - val_loss: 0.9222 - val_accuracy: 0.6880
Epoch 4/4
500/500 [==============================] - 82s 165ms/step - loss: 0.7497 - accuracy: 0.7483 - val_loss: 2.1827 - val_accuracy: 0.6805
CPU times: total: 25min 12s
Wall time: 5min 40s
```

2) Далі на основі ResNet50 було створено сіамську модель на основі евклідової відстані.

```
def distance(vectors):
    x, y = vectors
    sum_square = tf.math.reduce_sum(tf.math.square(x - y), axis=1, keepdims=True)
    return tf.math.sqrt(tf.math.maximum(sum_square, tf.keras.backend.epsilon()))

img_a_feat = model(tf.keras.layers.Input(shape = x_train.shape[1:], name='Image A input'))
img_b_feat = model(tf.keras.layers.Input(shape = x_train.shape[1:], name='Image B input'))

features = tf.keras.layers.Lambda(distance)([img_a_feat, img_b_feat])
features = tf.keras.layers.Dense(16, activation='relu')(features)
features = tf.keras.layers.BatchNormalization()(features)
features = tf.keras.layers.Activation('relu')(features)
features = tf.keras.layers.Dense(4, activation='relu')(features)
features = tf.keras.layers.Dropout(0.2)(features)
features = tf.keras.layers.Activation('relu')(features)
features = tf.keras.layers.Dense(1, activation='sigmoid')(features)

siamese_model = tf.keras.models.Model(inputs = [img_a_feat, img_b_feat], outputs = [features], name = 'Siamese_model')
siamese_model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['mae'])
```

Activation: Any

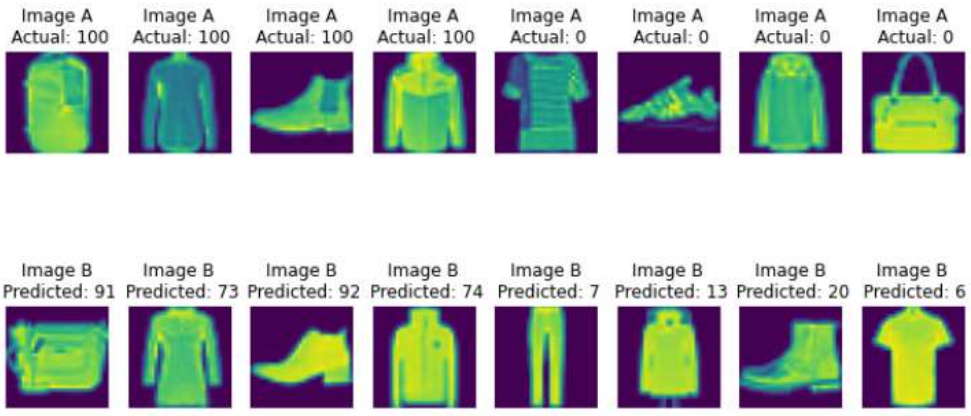
```
def siam_gen(in_groups, batch_size = 32):
    while True:
        pv_a, pv_b, pv_sim = gen_random_batch(in_groups, batch_size//2)
        yield [model.predict(pv_a), model.predict(pv_b)], pv_sim

valid_a, valid_b, valid_sim = gen_random_batch(test_groups, 1024)
loss_history = siamese_model.fit(siam_gen(train_groups),
    steps_per_epoch = 500,
    validation_data=([model.predict(valid_a), model.predict(valid_b)],
    valid_sim),
    epochs = 2,
    verbose = True)
```

✓ 3m 38.7s

```
Epoch 1/2
500/500 [==============================] - 113s 225ms/step - loss: 0.5132 - mae: 0.3748 - val_loss: 0.4402 - val_mae: 0.3189
Epoch 2/2
500/500 [==============================] - 98s 196ms/step - loss: 0.3967 - mae: 0.2618 - val_loss: 0.4017 - val_mae: 0.2549
```

```
_ = show_model_output(4)
```

✓ 0.7s



3) Наступним кроком результати роботи ResNet50 було візуалізовано засобами t-SNE.

```
%%time
from sklearn.manifold import TSNE
x_test_features = model.predict(x_test, verbose = True, batch_size=128)

tsne_obj = TSNE(n_components=2,
    init='pca',
    random_state=101,
    method='barnes_hut',
    n_iter=500,
    verbose=1)
tsne_features = tsne_obj.fit_transform(x_test_features)
```

```
obj_categories = [
                    'T-shirt/top','Trouser','Pullover','Dress',
                    'Coat','Sandal','Shirt','Sneaker','Bag','Ankle boot'
                  ]
colors = plt.cm.rainbow(np.linspace(0, 1, 10))
plt.figure(figsize=(10, 10))

for c_group, (c_color, c_label) in enumerate(zip(colors, obj_categories)):
    plt.scatter(tsne_features[np.where(y_test == c_group), 0],
        tsne_features[np.where(y_test == c_group), 1],
        marker='o',
        color=c_color,
        linewidth=1,
        alpha=0.8,
        label=c_label)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('t-SNE on Testing Samples')
plt.legend(loc='best')
plt.savefig('clothes-dist.png')
plt.show(block=False)
```

✓ 0.5s

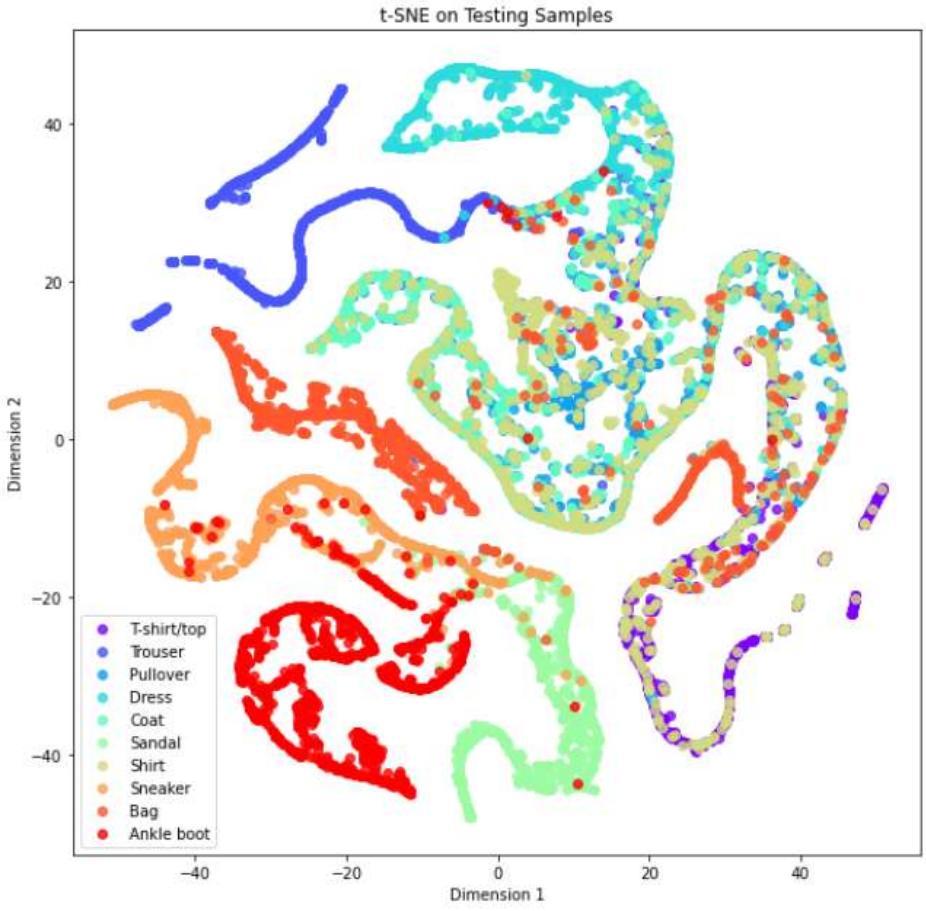


Рис. 1 Результат кластеризації властивостей, генерованих згортковою мережею.

Висновок: При виконанні даної лабораторної роботи було здійснено огляд практик класифікації зображень згортковими нейронними мережами, оглянуто вид сіамських нейронних мереж і метод кластеризації t-SNE.