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»

Виконала:
ст. групи КН-408
Бурцьо Ольга

ЛЬВІВ – 2022

ВАРІАНТ 5

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

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.

Препроцесинг даних

```
[26] (x_train, y_train), (x_test, y_test) = datasets.fashion_mnist.load_data()
     print("X_train shape:", x_train.shape)
     print("X_test shape:", x_test.shape)
```

```
X_train shape: (60000, 28, 28)
X_test shape: (10000, 28, 28)
```

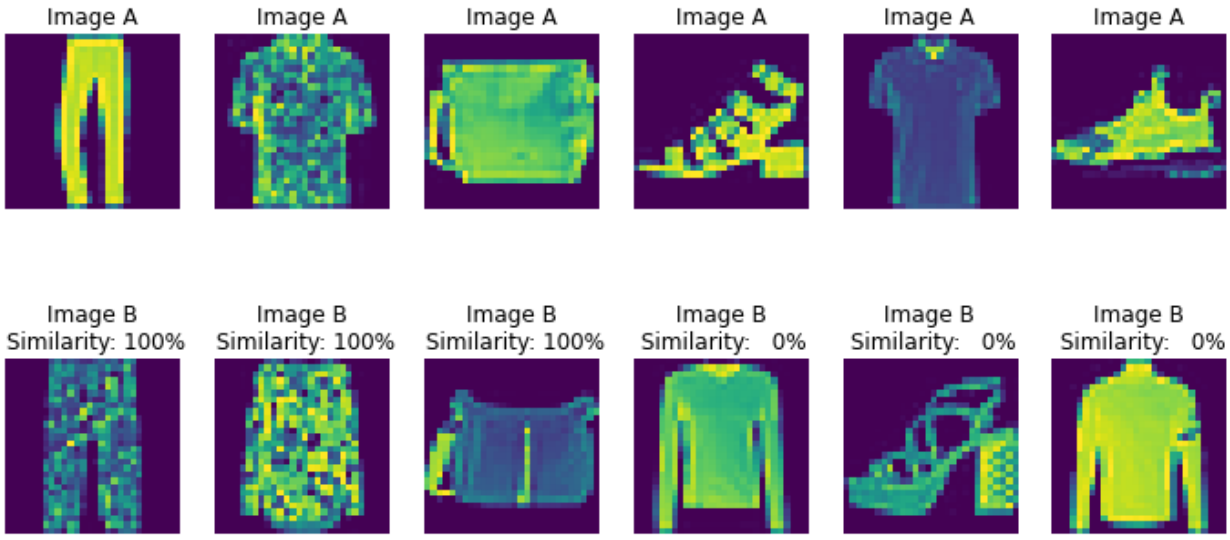
```
[27] x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.
     x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.

     print("X_train shape:", x_train.shape)
     print("X_test shape:", x_test.shape)
```

```
X_train shape: (60000, 28, 28, 1)
X_test shape: (10000, 28, 28, 1)
```

```
[28] # reorganize by groups
     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])
```

```
train groups: [6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000]
test groups: [1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000]
```

Дані датасету



Архітектура моделі

```
inputs = layers.Input(shape=x_train.shape[1:])

x = conv2D_bn_relu(inputs, filters=128, kernel_size=5, strides=1, padding='same', name='conv_1')
x = reduction_module_A(x, filters=256)
x = layers.SpatialDropout2D(0.3)(x)
x = inception_module_A(x, filters=256)
x = inception_module_A(x, filters=256)
x = reduction_module_A(x, filters=384)
x = layers.SpatialDropout2D(0.5)(x)
x = inception_module_C(x, filters=384)
x = inception_module_C(x, filters=384)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(32)(x)

model = keras.Model(inputs=inputs, outputs=x)
model.summary()
```

Siamese модель

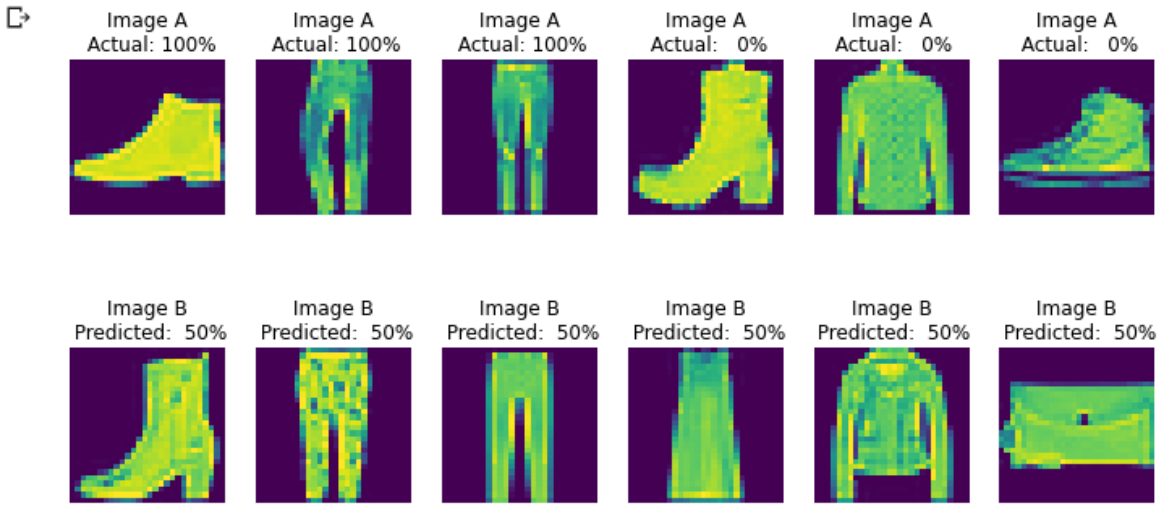
```
img_a_in = Input(shape = x_train.shape[1:], name = 'ImageA_Input')
img_b_in = Input(shape = x_train.shape[1:], name = 'ImageB_Input')

img_a_feat = model(img_a_in)
img_b_feat = model(img_b_in)

combined_features = concatenate([img_a_feat, img_b_feat], name ='merge_features')
combined_features = Dense(16, activation = 'linear')(combined_features)
combined_features = BatchNormalization()(combined_features)
combined_features = Activation('relu')(combined_features)
combined_features = Dense(4, activation = 'linear')(combined_features)
combined_features = BatchNormalization()(combined_features)
combined_features = Activation('relu')(combined_features)
combined_features = Dense(1, activation = 'sigmoid')(combined_features)
similarity_model = keras.Model(inputs = [img_a_in, img_b_in], outputs = [combined_features], name = 'Similarity_Model')
similarity_model.summary()
```

Використання RMSprop оптимізатора

```
[34] similarity_model.compile(optimizer='RMSprop', loss = 'binary_crossentropy', metrics = ['mae'])
```

Візуалізація результатів

```
def show_model_output(nb_examples = 3):
  pv_a, pv_b, pv_sim = gen_random_batch(test_groups, nb_examples)
  pred_sim = similarity_model.predict([pv_a, pv_b])
  fig, m_axs = plt.subplots(2, pv_a.shape[0], figsize = (12, 6))
  for c_a, c_b, c_d, p_d, (ax1, ax2) in zip(pv_a, pv_b, pv_sim, pred_sim, m_axs.T):
    ax1.imshow(c_a[:,:,0])
    ax1.set_title('Image A\n Actual: %3.0f%%' % (100*c_d))
    ax1.axis('off')
    ax2.imshow(c_b[:,:,0])
    ax2.set_title('Image B\n Predicted: %3.0f%%' % (100*p_d))
    ax2.axis('off')
  return fig
  #completely untrained model
_ = show_model_output()
```



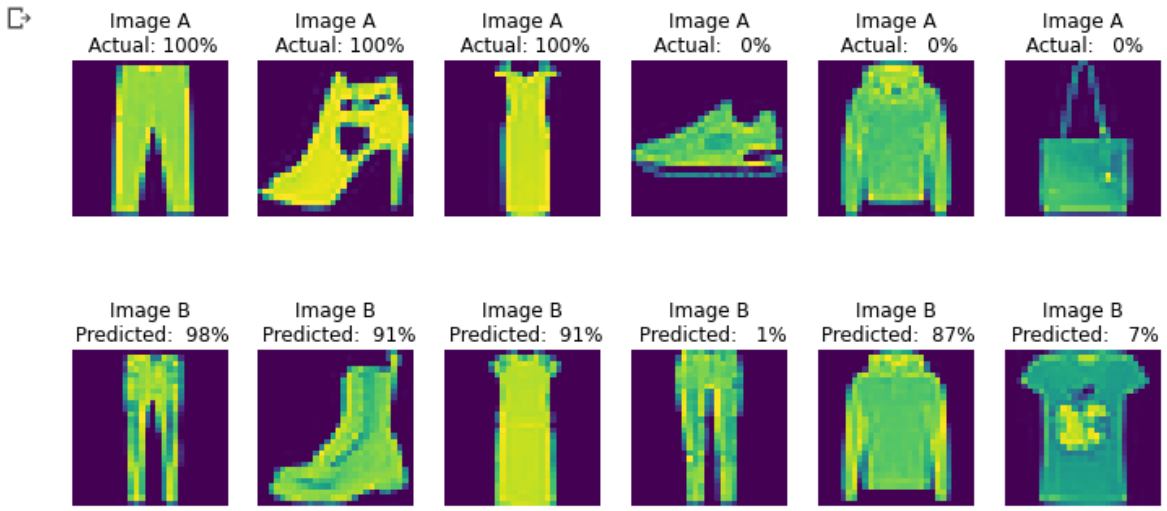
Тренування моделі Siamese на основі нейромережі

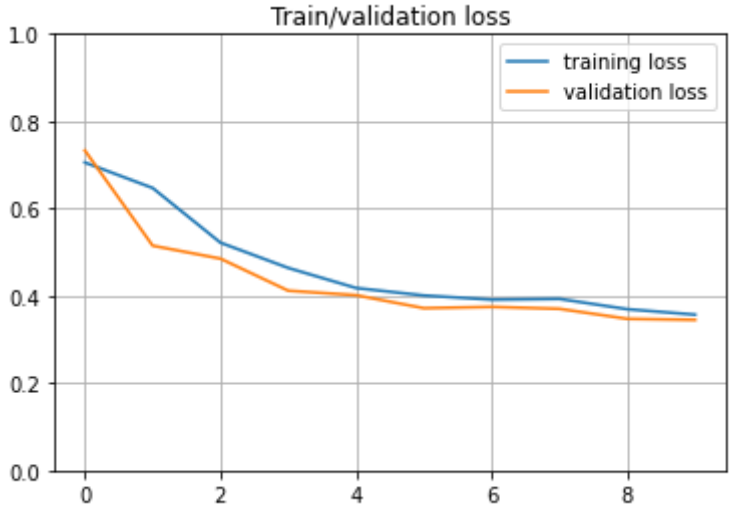
```
[36] def siam_gen(in_groups, batch_size = 32):
       while True:
         pv_a, pv_b, pv_sim = gen_random_batch(train_groups, batch_size//2)
         yield [pv_a, pv_b], pv_sim

     valid_a, valid_b, valid_sim = gen_random_batch(test_groups, 1024)
     loss_history = similarity_model.fit(siam_gen(train_groups), steps_per_epoch = 250, validation_data=([valid_a, valid_b], valid_sim), epochs = 10, verbose = True)
```

```
Epoch 1/10
250/250 [==============================] - 50s 159ms/step - loss: 0.7054 - mae: 0.5004 - val_loss: 0.7330 - val_mae: 0.5017
Epoch 2/10
250/250 [==============================] - 38s 152ms/step - loss: 0.6469 - mae: 0.4690 - val_loss: 0.5148 - val_mae: 0.3687
Epoch 3/10
250/250 [==============================] - 38s 153ms/step - loss: 0.5216 - mae: 0.3817 - val_loss: 0.4851 - val_mae: 0.3028
Epoch 4/10
250/250 [==============================] - 38s 153ms/step - loss: 0.4638 - mae: 0.3242 - val_loss: 0.4116 - val_mae: 0.2615
Epoch 5/10
250/250 [==============================] - 38s 151ms/step - loss: 0.4178 - mae: 0.2849 - val_loss: 0.4013 - val_mae: 0.2542
Epoch 6/10
250/250 [==============================] - 38s 152ms/step - loss: 0.4007 - mae: 0.2653 - val_loss: 0.3716 - val_mae: 0.2334
Epoch 7/10
250/250 [==============================] - 38s 153ms/step - loss: 0.3912 - mae: 0.2575 - val_loss: 0.3746 - val_mae: 0.2401
Epoch 8/10
250/250 [==============================] - 38s 152ms/step - loss: 0.3929 - mae: 0.2556 - val_loss: 0.3702 - val_mae: 0.2314
Epoch 9/10
250/250 [==============================] - 38s 152ms/step - loss: 0.3693 - mae: 0.2437 - val_loss: 0.3472 - val_mae: 0.2196
Epoch 10/10
250/250 [==============================] - 38s 152ms/step - loss: 0.3568 - mae: 0.2312 - val_loss: 0.3448 - val_mae: 0.2096
```

Результат

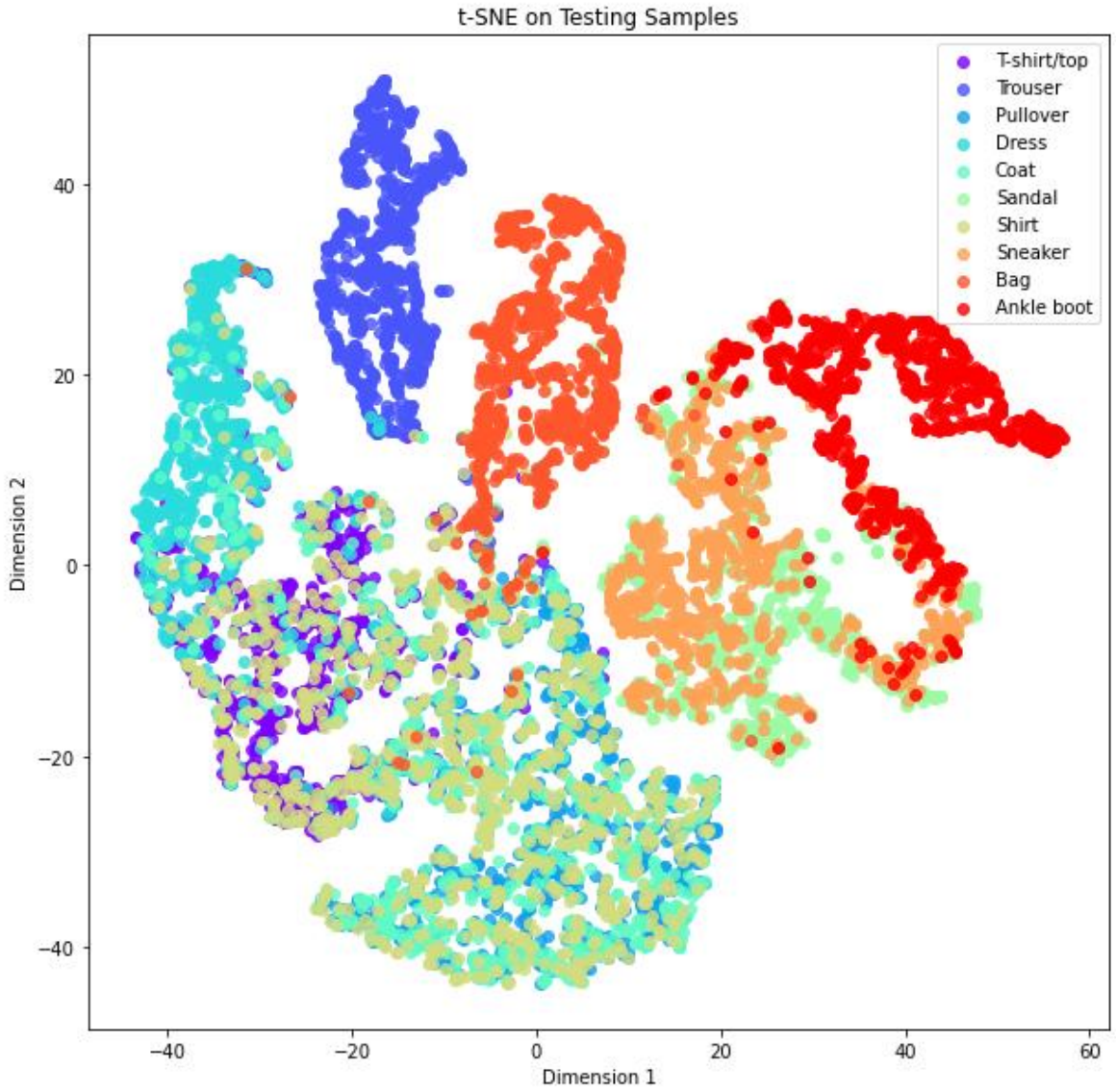
```
_ = show_model_output()
```





Тренування t-SNE на основі нейромережі

```
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=2)
tsne_features = tsne_obj.fit_transform(x_test_features)
```



Висновок: на цій лабораторної роботі я відтворила нейронну мережу inception-v3 та протестовано на Simemese. Також було досліджено t-sne модель на тестових даних.