МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кафедра «Системи штучного інтелекту»

Звіт

до лабораторної роботи №3

З дисципліни «Обробка зображень методами штучного інтелекту»

Виконав:

студент групи КН-409

Слава Л.Л.

Прийняв:

Пелешко Д.Д.

Львів-2022

Лабораторна робота 3

Класифікація зображень. Застосування нейромереж для пошуку подібних зображень.

Варіант 1

Побудувати CNN на основі LeNet-5 для класифікації зображень на основі датасету fashion-mnist. Зробити налаштування моделі для досягнення необхідної точностіНа базі Siamese networks побудувати систему для пошуку подібних зображень в датасеті fashion-mnist. Візуалізувати отримані результати t-SNE.

Після скачування датасету нам потрібно привезти його до розмірів 32x32x1, а шкали приведемо до меж від 0 до 1.

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

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

x_train = np.pad(x_train, ((0, 0), (2, 2), (2, 2), (0, 0)))
x_test = np.pad(x_test, ((0, 0), (2, 2), (2, 2), (0, 0)))

print('Training', x_train.shape, x_train.max())
print('Testing', x_test.shape, x_test.max())
```

Після ділення вибірок на класи пишемо функцію випадкового вибору зображень

```
def gen_random(in_groups, batch_halfsize=8):
    out_img_a, out_img_b, out_score = [], [], []
    all_groups = list(range(len(in_groups)))
    for match_group in [True, False]:
        group_idx = np.random.choice(all_groups, size=batch_halfsize)
        out_img_a += [in_groups[c_idx][np.random.choice(range(in_groups[c_idx].shape[0]))] for c_idx in group_idx]

        if match_group:
            b_group_idx = group_idx
            out_score += [1] * batch_halfsize
        else:
            non_group_idx = [np.random.choice([i for i in all_groups if i != c_idx]) for c_idx in group_idx]
            b_group_idx = non_group_idx
            out_score += [0] * batch_halfsize
        out_img_b += [in_groups[c_idx][np.random.choice(range(in_groups[c_idx].shape[0]))] for c_idx in b_group_idx]
    return np.stack(out_img_a, 0), np.stack(out_img_b, 0), np.stack(out_score, 0)
```

Реалізуємо згорткову мережу, використаємо активацію elu:

```
img_in = Input(shape = x_train.shape[1:], name = 'FeatureNet_ImageInput')
n_layer = img_in

n_layer = Conv2D(filters=6, kernel_size=5, strides=1, activation='elu', input_shape=(32, 32, 1))(n_layer)
n_layer = MaxPool2D(pool_size=2, strides=2)(n_layer)
n_layer = Conv2D(filters=16, kernel_size=5, strides=1, activation='elu', input_shape=(14, 14, 6))(n_layer)
n_layer = MaxPool2D(pool_size=2, strides=2)(n_layer)
n_layer = Flatten()(n_layer)
n_layer = Dense(units=120, activation='elu')(n_layer)
n_layer = Dense(units=84, activation='elu')(n_layer)
n_layer = Dropout(0.5)(n_layer)
n_layer = Dense(units=10, activation='softmax')(n_layer)

feature_model = Model(inputs = [img_in], outputs = [n_layer], name='FeatureGenerationModel')
tf.keras.utils.plot_model(feature_model, show_shapes=True, show_layer_names=True)
```

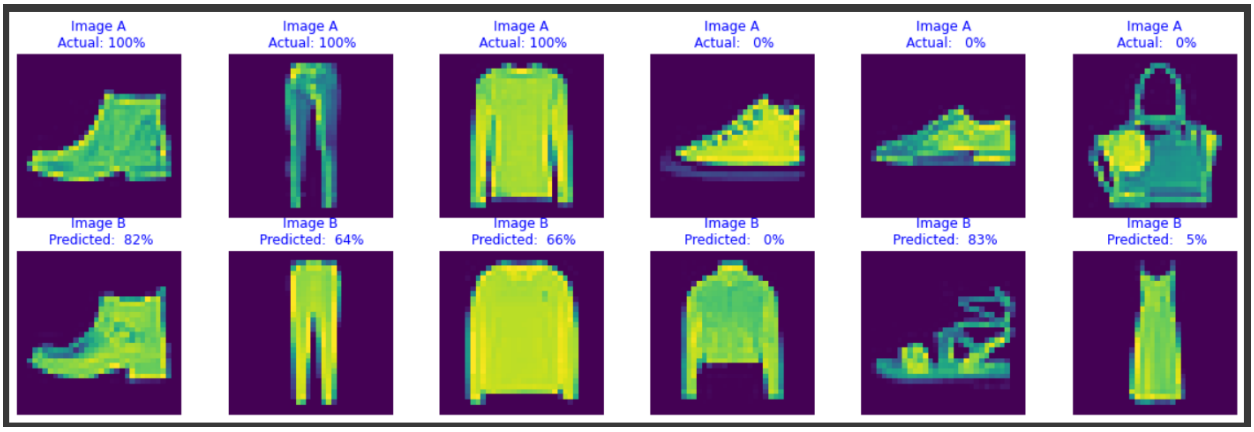
Створимо сіамську модель:

```
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


similarity_model.compile(optimizer='adamax', loss = 'binary_crossentropy', metrics = ['accuracy', 'mae'])

early_stop = tf.keras.callbacks.EarlyStopping(
    monitor="val_accuracy",
    min_delta=0,
    patience=5,
    verbose=0,
    mode="auto",
)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_accuracy',
    patience=3,
    verbose=1,
    factor=0.5,
    min_lr=1e-6,
)
valid_a, valid_b, valid_sim = gen_random_batch(test_groups, 1024)
loss_history = similarity_model.fit(siam_gen(train_groups),
    steps_per_epoch = 500,
    validation_data=([valid_a, valid_b],
    valid_sim),
    epochs = 40,
    verbose = True,
    callbacks = [early_stop,reduce_lr]
  )
```

```
Epoch 16/40
500/500 [==============================] - 44s 88ms/step - loss: 0.4078 - accuracy: 0.7954 - mae: 0.2759 - val_loss: 0.4123 - val_accuracy: 0.8008 - val_mae: 0.2748 - lr: 2.5000e-04
```



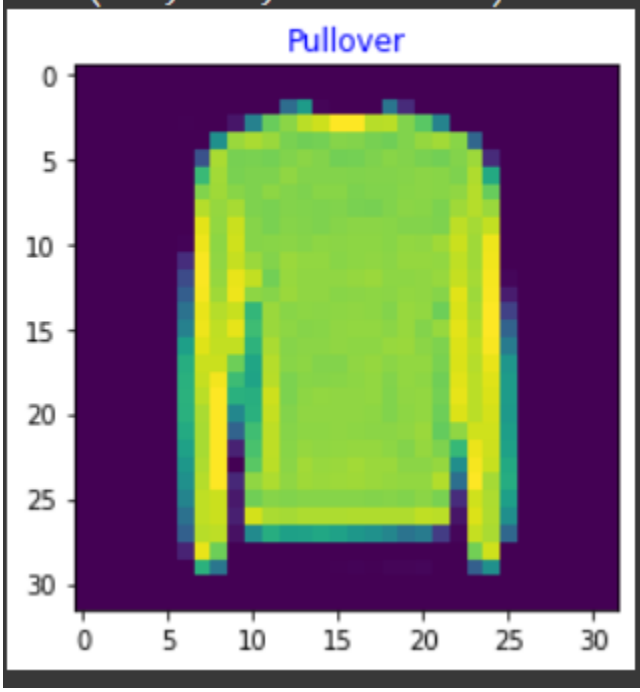
Виберемо рандомний елемент одягу:

```
from random import randint

obj_categories = ['T-shirt/top','Trouser','Pullover','Dress',
'Coat','Sandal','Shirt','Sneaker','Bag','Ankle boot']

idx = randint(0, 59999)
image = x_train[idx]
plt.imshow(image[:, :, 0])
plt.title(obj_categories[y_train[idx]])
```

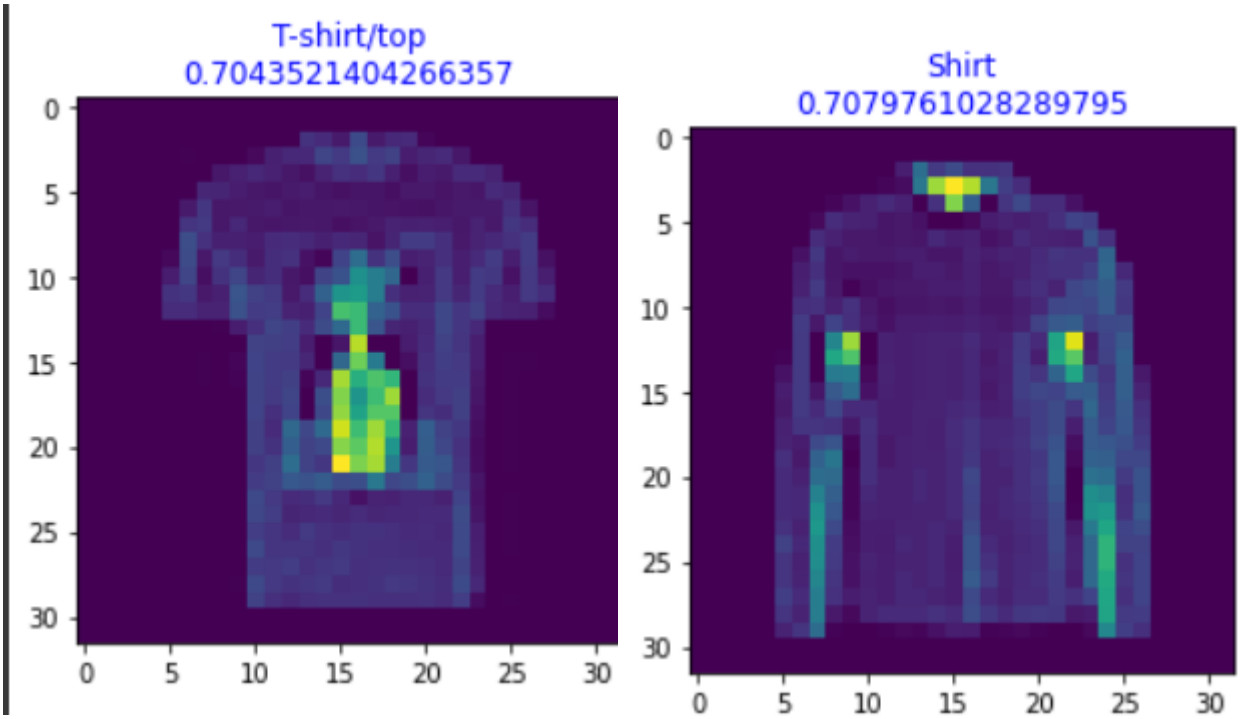
```
Text(0.5, 1.0, 'Pullover')
```



Та знайдемо схожий:

```
def find_simmilar(dataset, labels, image, res_len):
    compiled = [np.tile(image, (len(dataset), 1, 1, 1)), np.stack(dataset)]
    preds = similarity_model.predict(compiled).reshape(-1)
    top = np.argpartition(preds, -res_len)[-res_len:]
    for i in top:
        plt.imshow(dataset[i][:, :, 0])
        plt.title(f'{obj_categories[labels[i]]}\n{preds[i]}')
        plt.show()


find_simmilar(x_test, y_test, image, 10)
```



Останнім кроком буде побудова t-SNE діаграми:

```
colors = plt.cm.rainbow(np.linspace(0, 1, 10))
plt.figure(figsize=(15, 15))
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.legend()
plt.show(block=False)
```