МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

кафедра систем штучного інтелекту

ЗВІТ

про виконання лабораторної роботи №3

з курсу «Комп’ютерний зір»

Варіант №5

Виконав:

ст. групи КН-409

Чорній Юрій

Перевірив:

Пелешко Д. Д.

ЛЬВІВ – 2021

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.

Завдання

Побудувати CNN на основі Inception-v1 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.

Теоретичні відомості

В основі класифікації (для пошуку подібних) зображень пропонується використовувати Siamese networks. Ідея складається в тому щоб взяти випадково ініціалізовану мережу і застосувати її до зображень, щоб дізнатися наскільки вони схожі. Модель має значно полегшати виконання таких задач, як візуальний пошук по базі даних зображень, так як вона буде мати просту метрику подібності між 0 та 1 замість 2D масивів.

Хід роботи

1. Завантажую MNIST датасет

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

2. Нормалізую дані:

```
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.
y_train = y_train.astype('int')
y_test = y_test.astype('int')
print('Training', x_train.shape, x_train.max())
print('Testing', x_test.shape, x_test.max())
```

3. Створюю Inception v_3 модель

```
class Inception(keras.Model):

    def __init__(self, num_layers, num_classes, init_ch=16, **kwargs):
        super(Inception, self).__init__(**kwargs)

        self.in_channels = init_ch
        self.out_channels = init_ch
        self.num_layers = num_layers
        self.init_ch = init_ch

        self.conv1 = ConvBNRelu(init_ch)

        self.blocks = keras.models.Sequential(name='dynamic-blocks')

        for block_id in range(num_layers):

            for layer_id in range(2):

                if layer_id == 0:

                    block = InceptionBlk(self.out_channels, strides=2)

                else:
                    block = InceptionBlk(self.out_channels, strides=1)

                self.blocks.add(block)

            # enlarger out_channels per block
            self.out_channels *= 2

        self.avg_pool = GlobalAveragePooling2D()
        self.fc = Dense(num_classes)

    def call(self, x, training=None):
        out = self.conv1(x, training=training)

        out = self.blocks(out, training=training)

        out = self.avg_pool(out)
        out = self.fc(out)

        return out
```

4. Створюю сіамську модель та обєдную зі своєї

```
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 = Model(inputs = [img_a_in, img_b_in], outputs =
[combined_features], name = 'Similarity_Model')
similarity_model.summary()
```

5. Треную модель

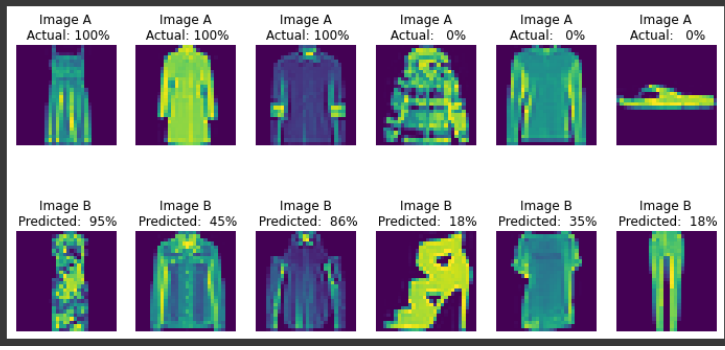
```
# make a generator out of the data
def siam_gen(in_groups, batch_size = 32):
  while True:
    pv_a, pv_b, pv_sim = gen_random_batch(train_groups, batch_size//2)
    yield [pv_a, pv_b], pv_sim
  # we want a constant validation group to have a frame of reference for model performance
valid_a, valid_b, valid_sim = gen_random_batch(test_groups, 1024)
loss_history = similarity_model.fit_generator(siam_gen(train_groups), steps_per_epoch = 500, validation_data=([valid_a, valid_b], valid_sim), epochs = 4, verbose = True)
```

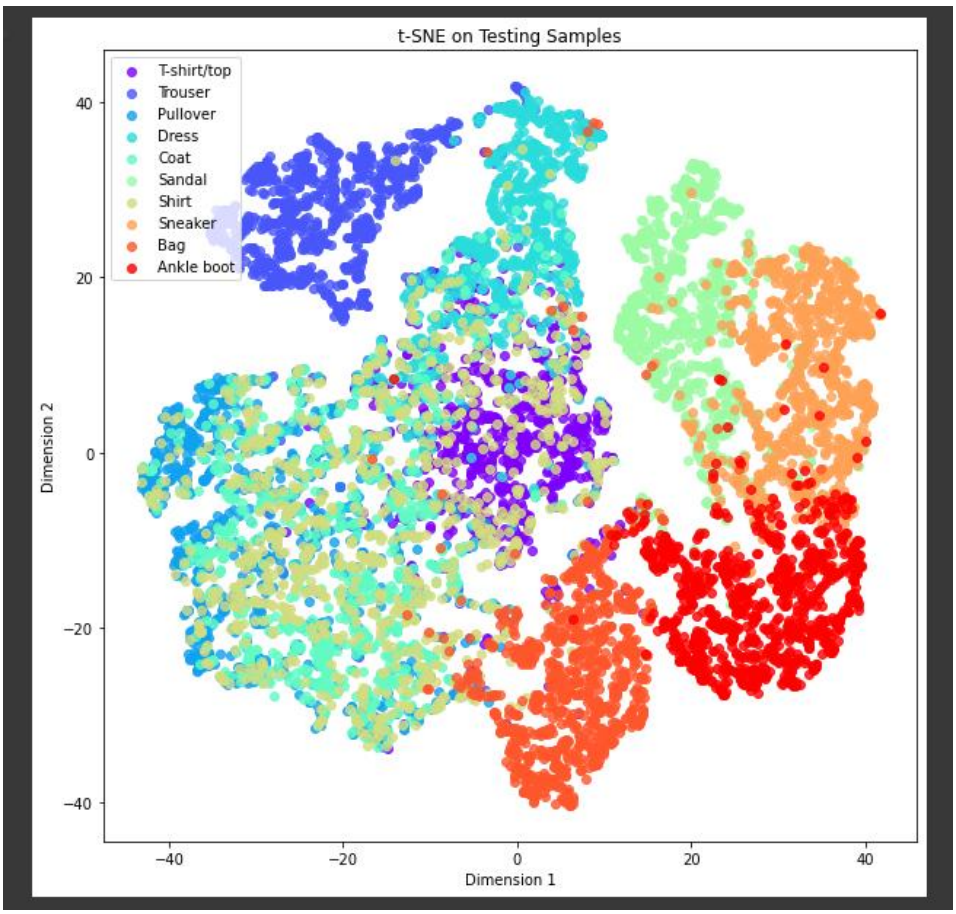
```
Epoch 1/4
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model

500/500 [==============================] - 35s 60ms/step - loss: 0.6012 - mae: 0.4230 - val_loss: 0.5985 - val_mae: 0.3940
Epoch 2/4
500/500 [==============================] - 29s 59ms/step - loss: 0.5037 - mae: 0.3608 - val_loss: 0.5016 - val_mae: 0.3431
Epoch 3/4
500/500 [==============================] - 30s 60ms/step - loss: 0.4483 - mae: 0.3167 - val_loss: 0.4781 - val_mae: 0.3104
Epoch 4/4
500/500 [==============================] - 29s 59ms/step - loss: 0.4031 - mae: 0.2810 - val_loss: 0.3967 - val_mae: 0.2787
```

6. Результати:

```
[13] 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
```

```
_ = show_model_output()
```





Висновок

Під час виконання лабораторної роботи я набув практичних навиків у розв’язанні задачі пошуку подібних зображень на прикладі організації CNN класифікації, створив мережу inception-v3 та об’єднав з мережею Simemese. Дослідив результати за допомогою t-sne.