1. **Постановка задачи.**

Настроить предобученную нейронную сеть для классификации рукописных цифр. Набор данных - MNIST. Результат оформить в виде отчета, включив в него

- описание выполненных действий;

- model.summary() полученной модели;

- графики обучения;

- полученную точность классификации изображений обучающего и проверочного множеств MNIST.

В качестве исходной берется VGG16.

2. **Описание выполненных действий.**

a. Загрузка набора данных.
b. Нормализация данных изображения, приведение к типу ‘float32’ и преобразование в массив numpy.
c. Преобразование меток в one-hot (категориальное) представление.
d. Изменение размера изображений для подачи на вход нейронной сети: для изображения добавление дополнительного измерения (np.expand_dims), увеличение количества каналов (np.repeat), изменение формы изображения (tf.image.resize) – форма на входе нейронной сети – (32, 32, 3) – минимальная допустимая входная форма.
e. Настройка встроенной модели: include_top = False, weights = “imagenet”, input_tensor = tf.keras.Input(shape=(32,32,3)). Отключение обучения слоёв свёрточной части сети.
```
nL = len(base_model.layers) # no learn
for layer in base_model.layers[:nL - 1]:
    layer.trainable = False
```
f. Добавление своих полносвязных слоёв:
```
x = base_model.output
x = Flatten()(x)
x = Dense(2048, activation = 'linear')(x)
x = Dense(1024, activation = 'linear')(x)
output = Dense(num_classes, activation = 'softmax')(x)
```
g. Сборка получившейся модели.
```
model = Model(inputs = base_model.input, outputs = output)
model.compile(optimizer = 'Adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```
h. Обучение получившейся модели
```
model.fit(x_train.numpy(), y_train, batch_size = 128, epochs=5,
                verbose=1, validation_data = (x_test.numpy(), y_test))
```
i. Вывод графиков обучения и точности классификации модели

3. **model.summary() полученной модели**

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 32, 32, 3)]	0
block1_conv1 (Conv2D)	(None, 32, 32, 64)	1792
block1_conv2 (Conv2D)	(None, 32, 32, 64)	36928
block1_pool (MaxPooling2D)	(None, 16, 16, 64)	0
block2_conv1 (Conv2D)	(None, 16, 16, 128)	73856
block2_conv2 (Conv2D)	(None, 16, 16, 128)	147584
block2_pool (MaxPooling2D)	(None, 8, 8, 128)	0
block3_conv1 (Conv2D)	(None, 8, 8, 256)	295168
block3_conv2 (Conv2D)	(None, 8, 8, 256)	590080
block3_conv3 (Conv2D)	(None, 8, 8, 256)	590080
block3_pool (MaxPooling2D)	(None, 4, 4, 256)	0
block4_conv1 (Conv2D)	(None, 4, 4, 512)	1180160
block4_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block4_conv3 (Conv2D)	(None, 4, 4, 512)	2359808

block4_pool (MaxPooling2D) (None, 2, 2, 512) 0

block5_conv1 (Conv2D) (None, 2, 2, 512) 2359808

block5_conv2 (Conv2D) (None, 2, 2, 512) 2359808

block5_conv3 (Conv2D) (None, 2, 2, 512) 2359808

block5_pool (MaxPooling2D) (None, 1, 1, 512) 0

flatten (Flatten) (None, 512) 0

dense (Dense) (None, 2048) 1050624

dense_1 (Dense) (None, 1024) 2098176

dense_2 (Dense) (None, 10) 10250

===

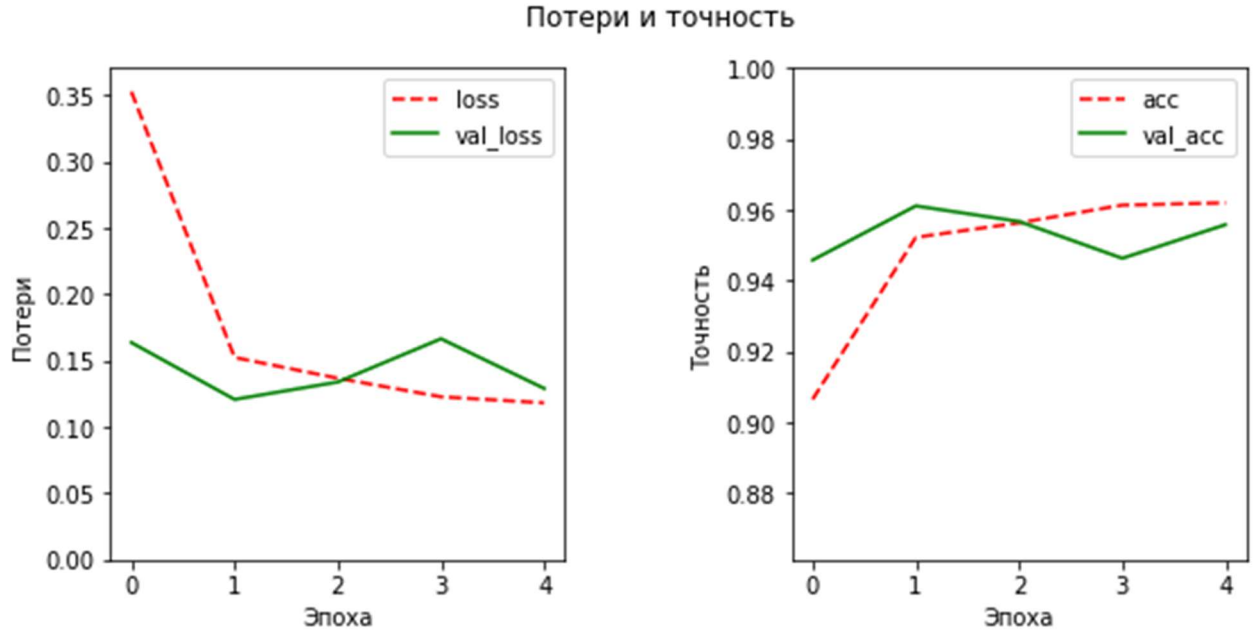
Total params: 17,873,738

Trainable params: 3,159,050

Non-trainable params: 14,714,688

_____

4. Графики обучения



5. Полученную точность классификации изображений обучающего и проверочного множеств MNIST

Точность на обучающем множестве: 95.93 %.

Точность на проверочном множестве: 95.58 %.

Точность по классам на обучающем множестве

Класс	Количество представителей класса	Точность, %
0	5923	96.02
1	6742	98.25
2	5958	87.11
3	6131	92.69
4	5842	95.69
5	5421	97.71
6	5918	99.1
7	6265	97.83
8	5851	97.25
9	5949	97.51

Точность по классам на проверочном множестве:

Класс	Количество представителей класса	Точность, %
0	980	95.51
1	1135	98.5
2	1032	87.5
3	1010	92.48
4	982	96.33
5	892	96.52
6	958	99.06
7	1028	96.5
8	974	97.74
9	1009	95.83