# *Лабораторная работа 2*

# *Классификация изображений с помощью сверточной сети*

**Задание 1.**

Необходимо построить CNN с топологией, аналогичной той, что была описана в лекции. Она должна состоять из двух сверточных слоев и одного полносвязного слоя.

Шаги описываются в двух измерениях, поскольку не требуется, чтобы шаги в каждом направлении были одинаковыми. Для двух сверточных слоев количество обучаемых параметров зависит не от количества нейронов в слое, а только от количества каналов и весов на нейрон. Для полносвязного слоя количество обучаемых параметров зависит от количества нейронов. Это приводит к тому, что, хотя первый слой имеет в четыре раза больше нейронов, чем второй, и в 1 638 раз больше нейронов, чем последний, он имеет только примерно на 10% больше обучаемых весов, чем каждый из двух последующих слоев.

1) Разберите фрагмент кода 1, дайте пояснения к происходящему в нем. Сделайте вывод о преимуществах и недостатках. Приведите графики для тестовой ошибки и ошибки обучения.
2) Попробуйте разные конфигурации, соответствующие таблице 1. Сведите полученные результаты также в таблицу. По каждому варианту добавьте вывод и сравнение. Подведите итог.

В таблице 1 сверточный слой обозначается, начиная с заглавной буквы C, за которой следуют три цифры, указывающие количество каналов, ширину и высоту. Полносвязный слой обозначается заглавной буквой F, за которой следует количество нейронов, и есть также третий тип слоя, MaxPool. Для сверточных слоев указаны размер ядра (K) и шага (S), где используется один и тот же размер в обоих направлениях; например, "K=5, S=2" означает ядро 5×5 и ширину шага 2×2. Для каждого слоя также указывается тип функции активации.

Таблица 1 – Конфигурации для экспериментов с CNN

| Конфигурация | Слои | Регуляризация |
|---|---|---|
| Conf1 | C64×16×16, K=5, S=2, ReLU C64×8×8, K=3, S=2, ReLU F10, softmax, cross-entropy loss | |
| Conf2 | C64×16×16, K=3, S=2, ReLU C16×8×8, K=2, S=2, ReLU F10, softmax, | |

| | | |
|---|---|---|
| | cross-entropy loss | |
| Conf3 | C64×16×16, K=3, S=2, ReLU C16×8×8, K=2, S=2, ReLU F10, softmax, cross-entropy loss | Dropout=0.2<br>Dropout=0.2 |
| Conf4 | C64×32×32, K=4, S=1, ReLU C64×16×16, K=2, S=2, ReLU C32×16×16, K=3 S=1, ReLU MaxPool, K=2, S=2 F64, ReLU F10, softmax, cross-entropy loss | Dropout=0.2<br>Dropout=0.2<br>Dropout=0.2<br>Dropout=0.2 |
| Conf5 | C64×32×32, K=4, S=1, ReLU C64×16×16, K=2, S=2, ReLU C32×16×16, K=3 S=1, ReLU C32×16×16, K=3 S=1, ReLU MaxPool, K=2, S=2 F64, ReLU F64, ReLU F10, softmax, cross-entropy loss | Dropout=0.2<br>Dropout=0.2<br>Dropout=0.2<br>Dropout=0.2<br>Dropout=0.2<br>Dropout=0.2 |
| Conf6 | C64×32×32, K=4, S=1, tanh C64×16×16, K=2, S=2, tanh C32×16×16, K=3 S=1, tanh C32×16×16, K=3 S=1, tanh MaxPool, K=2, S=2 F64, tanh F64, tanh F10, softmax, MSE loss | |

***Фрагмент кода 1 – Код инициализации для сверточной сети***

```
import tensorflow as  tf
from tensorflow import keras
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Conv2D
import numpy as np
```

```
import logging
tf.get_logger().setLevel(logging.ERROR)
EPOCHS = 128
BATCH_SIZE = 32
# Загрузка набора данных
cifar_dataset = keras.datasets.cifar10
  (train_images, train_labels), (test_images,
       test_labels) = cifar_dataset.load_data()
# Стандартизация набора данных
mean = np.mean(train_images)
stddev = np.std(train_images)
train_images (train_images - mean) / stddev
test_images (test_images - mean) / stddev
print('mean: ', mean)
print('stddev: ', stddev)
train_labels = to_categorical(train_labels, num_classes=10)
test_labels = to_categorical(test_labels, num_classes=10)
# Модель с двумя сверточными и одним полносвязным слоем
model = Sequential()
model.add(Conv2D(64, (5, 5), strides=(2,2),
            activation='relu', padding='same',
            input_shape=(32, 32, 3),
            kernel_initializer='he_normal', 'zeros'))

model.add(Conv2D(64, (3, 3), strides=(2,2),
            activation='relu', padding='same',
            kernel_initializer='he_normal',
            bias_initializer='zeros'))
model.add(Flatten())
model.add(Dense(10, activation='softmax',
            kernel_initializer='glorot_uniform',
            bias_initializer='zeros'))

model.compile(loss='categorical_crossentropy',
                optimizer='adam', metrics =['accuracy'])
model. summary ()
history = model. fit(
       train_images, train_labels, validation_data =
```

```
    (test_images, test_labels), epochs=EPOCHS,
    batch_size=BATCH_SIZE, verbose=2, shuffle=True)
```

**Задание 2.**

Разберите фрагменты кода 2 и 3, дайте пояснения и комментарии к происходящему в них. Приведите результаты работы для каждого, сделайте выводы.

***Фрагмент кода 2 – Общее представление о свертке***

```
import numpy as np
from sklearn.datasets import load_sample_image


# Загрузка изображений
china = load_sample_image("china.jpg") / 255
flower = load_sample_image("flower.jpg") / 255
images = np.array([china, flower])
batch_size, height, width, channels = images.shape


# Создание 2 фильтров
filters = np.zeros(shape=(7, 7, channels, 2),
dtype=np.float32)
filters[:, 3, :, 0] = 1 # вертикальная линия
filters[3, :, :, 1] = 1  # горизонтальная линия


outputs = tf.nn.conv2d(images, filters, strides=1,
            padding="SAME")


plt.imshow(outputs[0, :, :, 1], cmap="gray") # plot 1st
            image's 2nd feature map
```

```
plt.axis("off")
plt.show()
for image_index in (0, 1):
    for feature_map_index in (0, 1):
        plt.subplot(2, 2, image_index * 2 + feature_map_index
+ 1)
        plot_image(outputs[image_index, :, :,
feature_map_index])

plt.show()
def crop(images):
    return images[150:220, 130:250]
plot_image(crop(images[0, :, :, 0]))
save_fig("china_original", tight_layout=False)
plt.show()

for feature_map_index, filename in
enumerate(["china_vertical", "china_horizontal"]):
    plot_image(crop(outputs[0, :, :, feature_map_index]))
    save_fig(filename, tight_layout=False)
    plt.show()
plot_image(filters[:, :, 0, 0])
plt.show()
plot_image(filters[:, :, 0, 1])
plt.show()
```

***Фрагмент кода 3 – Сверточный слой***

```
np.random.seed(42)
tf.random.set_seed(42)

conv = keras.layers.Conv2D(filters=2, kernel_size=7,
strides=1,
                           padding="SAME", activation="relu",
input_shape=outputs.shape)
conv_outputs = conv(images)
conv_outputs.shape
plt.figure(figsize=(10,6))
for image_index in (0, 1):
    for feature_map_index in (0, 1):
        plt.subplot(2, 2, image_index * 2 + feature_map_index
```

```
+ 1)
        plot_image(crop(conv_outputs[image_index, :, :,
feature_map_index]))
plt.show()
conv.set_weights([filters, np.zeros(2)])
conv_outputs = conv(images)
conv_outputs.shape
plt.figure(figsize=(10,6))
for image_index in (0, 1):
    for feature_map_index in (0, 1):
        plt.subplot(2, 2, image_index * 2 + feature_map_index
+ 1)
        plot_image(crop(conv_outputs[image_index, :, :,
feature_map_index]))
plt.show()
```