Котолевский Максим Николаевич, группа 19

Лабораторная работа № 2

Вариант № 9

Цель работы

Обучить сверточную нейронную сеть задаче классификации на небольшом объеме данных.

Задание

Решить задачу классификации исходного изображения с помощью глубокой сверточной нейронной сети (арифметические операции). Оценить точность полученной модели. Не использовать переобученную нейронную сеть

Код программы (внесённые изменения в шаблон кода выделены)

Ссылка на исходный dataset:
https://www.kaggle.com/datasets/sagyamthapa/handwritten-math-symbols
(использовались только классы add, sub, mul, div)

splitting.py: разбиение исходного набора данных на обучающую, тестовую и валидационную выборки.

```
import shutil
import os

data_dir = 'C:/Users/user/Desktop/project_2/data/train'
# Каталог с набором данных

train_dir = 'train_1'  # Каталог с данными для обучения
val_dir = 'val_1'  # Каталог с данными для проверки
test_dir = 'test_1'  # Каталог с данными для тестирования

test_data_portion = 0.15  # Часть набора данных для тестирования
val_data_portion = 0.15  # Часть набора данных для проверки

nb_images = 551  # Количество элементов данных в одном классе


def create_dir(dir_name):
    if os.path.exists(dir_name):
        shutil.rmtree(dir_name)

    os.makedirs(dir_name)
    os.makedirs(os.path.join(dir_name, "add"))
    os.makedirs(os.path.join(dir_name, "sub"))
    os.makedirs(os.path.join(dir_name, "mul"))
    os.makedirs(os.path.join(dir_name, "div"))


create_dir(train_dir)
create_dir(val_dir)
create_dir(test_dir)


def copy_img(start_index, end_index, source_dir, dest_dir):
    for i in range(start_index, end_index):
        shutil.copy2(os.path.join(source_dir, "add(" + str(i) + ").jpg"),
os.path.join(dest_dir, "add"))
        shutil.copy2(os.path.join(source_dir, "sub(" + str(i) + ").jpg"),
os.path.join(dest_dir, "sub"))
```

```
        shutil.copy2(os.path.join(source_dir, "mul(" + str(i) + ").jpg"),
os.path.join(dest_dir, "mul"))
        shutil.copy2(os.path.join(source_dir, "div(" + str(i) + ").jpg"),
os.path.join(dest_dir, "div"))


start_val_data_idx = int(nb_images * (1 - val_data_portion -
test_data_portion))
start_test_data_idx = int(nb_images * (1 - test_data_portion))
print("Стартовый индекс валидационной выборки: ", start_val_data_idx)
print("Стартовый индекс тестовой выборки: ", start_test_data_idx)


copy_img(0, start_val_data_idx, data_dir, train_dir)
copy_img(start_val_data_idx, start_test_data_idx, data_dir, val_dir)
copy_img(start_test_data_idx, nb_images, data_dir, test_dir)
```

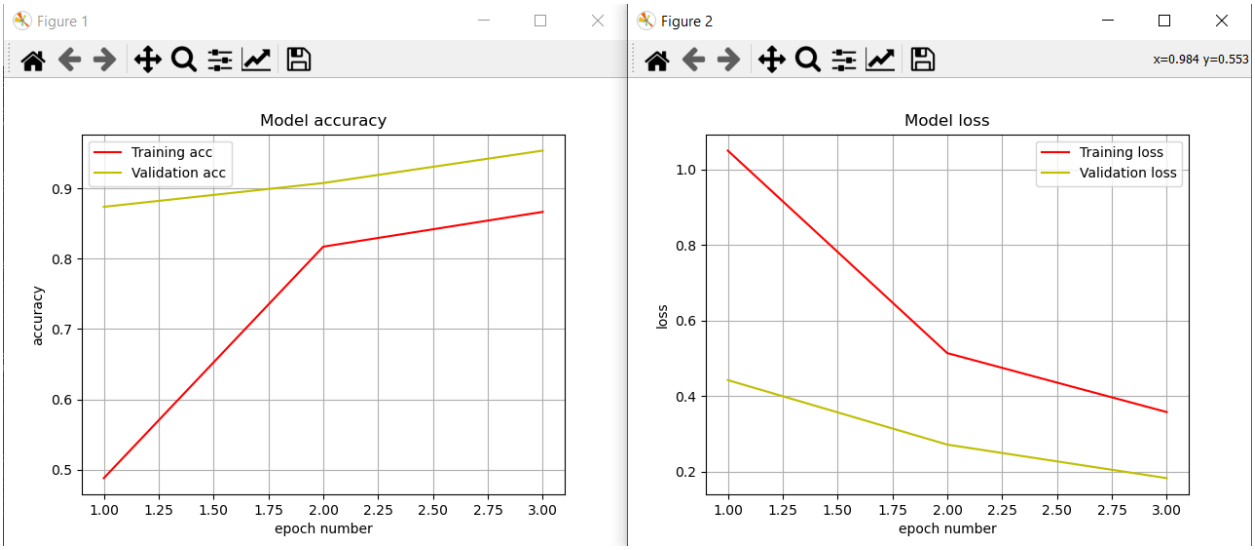
operations.py: обучение, тестирование и проверка сети.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

train_dir = 'C:/Users/user/Desktop/project_2/train_1'
# Каталог с данными для обучения
val_dir = 'C:/Users/user/Desktop/project_2/val_1'
# Каталог с данными для проверки
test_dir = 'C:/Users/user/Desktop/project_2/test_1'
# Каталог с данными для тестирования

img_width, img_height = 150, 150  # Размеры изображений
# Размерность тензора на основе изображения для входных данных в нейронную
сеть backend Tensorflow, channels_last
input_shape = (img_width, img_height, 3)

epochs = 3  # Количество эпох
batch_size = 25  # Размер мини-выборки

nb_train_samples = 1540  # Количество изображений для обучения
nb_validation_samples = 332  # Количество изображений для проверки
nb_test_samples = 332  # Количество изображений для тестирования

num_classes = 4  # Количество классов изображений (+, -, *, /)

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.5))
```

```
model.add(Dense(num_classes))
model.add(Activation('sigmoid'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='sparse'
)

val_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='sparse'
)

test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='sparse'
)

history = model.fit(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=val_generator,
    validation_steps=nb_validation_samples // batch_size
)

val_scores = model.evaluate(val_generator)
test_scores = model.evaluate(test_generator)
print(f"Accuracy on validation data: {(val_scores[1]*100):2}")
print(f"Accuracy on test data: {(test_scores[1]*100):2}")
print("Number of images: ", nb_train_samples + nb_validation_samples +
nb_test_samples)
list_epochs = range(1, epochs + 1)

plt.title('Model accuracy')
plt.plot(list_epochs, history.history['accuracy'], 'r', label='Training acc')
plt.plot(list_epochs, history.history['val_accuracy'], 'y', label='Validation
acc')
plt.xlabel('epoch number')
plt.ylabel('accuracy')
plt.grid(visible='on')
plt.legend()
plt.figure()

plt.title('Model loss')
plt.plot(list_epochs, history.history['loss'], 'r', label='Training loss')
plt.plot(list_epochs, history.history['val_loss'], 'y', label='Validation
loss')
plt.xlabel('epoch number')
plt.ylabel('loss')
plt.grid(visible='on')
plt.legend()
plt.show()
```

Результаты выполнения задания

```
Found 1540 images belonging to 4 classes.
Found 332 images belonging to 4 classes.
Found 332 images belonging to 4 classes.
Epoch 1/3
61/61 [==============================] - 34s 524ms/step - loss: 1.0494 - accuracy: 0.4878 - val_loss: 0.4425 - val_accuracy: 0.8738
Epoch 2/3
61/61 [==============================] - 24s 396ms/step - loss: 0.5139 - accuracy: 0.8172 - val_loss: 0.2719 - val_accuracy: 0.9077
Epoch 3/3
61/61 [==============================] - 22s 357ms/step - loss: 0.3579 - accuracy: 0.8667 - val_loss: 0.1832 - val_accuracy: 0.9538
14/14 [==============================] - 1s 91ms/step - loss: 0.1830 - accuracy: 0.9548
14/14 [==============================] - 1s 97ms/step - loss: 0.2137 - accuracy: 0.9548
Accuracy on validation data: 95.48192620277405
Accuracy on test data: 95.48192620277405
Number of images:  2204
```



Точность полученной нейронной сети равна 95%. Было подано 3 эпохи, если подать большее количество эпох произойдет переобучение.