

Лабораторная работа №3

Использование Python-библиотек для работы с ИИ

Выполнил: Галкин Климентий. 4851003/10002

Цель работы

Цель работы - получение навыков работы с методами ИИ для решения задачи классификации данных с использованием языка программирования Python. Задание на работу

- Изучить исходный код из условия лабораторной работы
- Проверить работоспособность кода на основе тестовых датасетов
- Сформировать датасеты в соответствии с вариантом
- Модифицировать скрипт для работы с аргументами командной строки
- Протестировать работу программы на разных изображениях
- Ответить на контрольные вопросы

Ход работы

Изначально исходный код формирует классы картинок на основе картинок, которые лежат в директориях `train` и `valid`. Все картинки делятся на так называемые `batches`, то есть пакет или партия. Такое разделение делается потому, что нельзя за один раз через нейронную сеть прогнать весь датасет, поэтому мы делим это все на блоки. Более того, все картинки подгоняются под размер 200x200 пикселей для однообразия данных, чтобы модели было проще анализировать.

После инициализируется сама модель, в которой указывается размерность текущих данных (у нас будет 200 списков длиной 200, указывается в `input_shape` параметре), создается два слоя – основной полносвязный `tf.nn.relu` и сигмоида `tf.nn.sigmoid`, некий слой усиления выходных результатов.



Проверяемая картинка масштабируется в соотношении 1/255, переводится в массив данных и отдается модели для классификации.

Для работы с аргументами командной строки был добавлен список аргументов от модуля `argparse`. В этих аргументах указывается путь до тренировочного датасета и датасета валидации данных, а так же путь до самой картинки, которую надо классифицировать. На выходе – результат классификации и средняя точность на всех пройденных эпохах. Теперь для проверки были сформированы и скачаны два датасета: картинки людей и картинки акул, количество картинок вышло следующее:

Из-за такого количества картинок в сете нужно изменить параметры `batch_size`. Для тренировочного генератора сделан `batch_size=190`, для валидации – 15. Размеры слоев

остались те же, количество проходимых эпох для валидации – 15. Количество шагов на эпохе – 8. Таким образом, результаты тестирования картинок представлены в таблице ниже:

Изменение количества эпох, размера сета(batch_size), конфигурации модели (например, основной слой будет размерность 256 вместо 128), могут привести к увеличению точности классификации картинки. Так, например, увеличение количества эпох привело к тому, что первая картинка из таблицы определилась, как акула, с точностью примерно 0.967, что больше на 0.014, чем предыдущий показатель. В рамках обучения моделей это достаточно весомый прогресс. Результат этой классификации ниже:

Картинка	Результаты
	Shark 95%
	Human 96%

Картинка**Результаты**

Human 94%

Однако при сильном изменении параметров модель может переобучиться и выдавать ложный результат. Новые изменения: увеличение размерности слоя до 256, увеличение количества эпох до 25 и количества шагов в эпохе до 15 вместо 8 – привели к ложным результатам:

Вывод

В ходе лабораторной работы были изучены некоторые методы из модулей языка Python для работы с ИИ. Обученная на сформированных датасетах модель показала высокую точность классификации картинок из разных наборов: случайная картинка из сети, картинка из набора для обучения и картинка для набора из валидации. Была изучена зависимость точности предсказания от параметров модели и валидации.

Ответы на вопросы

1. Какие классы задач могут быть решены с помощью методов искусственного интеллекта?

Спектр задач действительно большой: прогнозирование событий, классификация и кластеризация данных, обработка языка, речи и другой медиаинформации, автоматизация рутинных процессов и так далее.

2. Чем отличается обучающий набор данных от тестового?

На обучающем наборе данных модель обучается и развивается, обновляя свои параметры. Этот набор содержит метки классов (в нашем случае это "человек" или "акула") и целевые значения, которые модель должна предсказать. Тестовый набор данных – набор, приближенный по своим значениям и распределениям к обучающему. Такой набор используется для оценки производительности и эффективности модели и ее параметров.

3. Что такое признак в контексте методов искусственного интеллекта? Что такое метка в контексте методов искусственного интеллекта? В чем их разница?

Воспринимается все так же, как и в реальной жизни. Признак – характеристика объекта. Например, машина определенного цвета или размера. Метка – категория объекта. Например, если рассматривать все классы машин, то метки будут: "седан", "купе" и т. д. Разница между ними в том, что признаки описывают объект, а метка указывает возможный класс объекта. Признак используется при обучении модели, а метка – для классификации, определения объекта.

4. Чем методы глубокого обучения отличаются от других методов искусственного интеллекта?

Если говорить коротко, то глубокое обучение – имитация человеческого сознания. Для создания такой имитации требуется большой объем данных, высокопроизводительные системы и большое количество времени для обучения, так как в процессе развития модели она сама создает новые признаки, в отличие от машинного обучения, где признаки заранее задаются. Так же выходные данные могут принимать разный формат: текст, оценка, звук и др.

5. Из чего состоит слой в нейронной сети? Какие слои бывают? Что такое нейрон?

Слой в нейронной сети состоит из набора нейронов, которые обрабатывают входные данные и генерируют выходные данные. Каждый нейрон принимает на вход некоторое количество значений (входных данных), выполняет некоторые операции с этими значениями и генерирует выходные данные. Существует несколько типов слоев в нейронной сети, включая полносвязный слой, сверточный слой, пулинговый слой и рекуррентный слой. Полносвязный слой соединяет каждый нейрон входного слоя с каждым нейроном выходного слоя. Сверточный слой используется для анализа изображений и имеет специальную архитектуру, которая позволяет выделять важные признаки изображения. Пулинговый слой используется для уменьшения размерности данных, что позволяет упростить модель и ускорить ее обучение. Рекуррентный слой используется для обработки последовательностей данных, таких как тексты или звуковые файлы.

Приложение

```
import os
import argparse as ap
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import load_img, img_to_array

# Parse arguments
parser = ap.ArgumentParser(prog="Image classification model")
```

```

parser.add_argument("--train-ds", help="Destination of train datasets
folder")
parser.add_argument("--valid-ds", help="Destination of valid datasets
folder")
parser.add_argument("--image", help="Path to image to be tested")
args = vars(parser.parse_args())
image_path = args['image']

# Config paths to datasets
train_dir = os.listdir(args['train_ds'])
valid_dir = os.listdir(args['valid_ds'])
train_human_dir = os.path.join(f"{args['train_ds']}/{train_dir[1]}")
train_animal_dir = os.path.join(f"{args['train_ds']}/{train_dir[0]}")
valid_human_dir = os.path.join(f"{args['valid_ds']}/{valid_dir[1]}")
valid_animal_dir = os.path.join(f"{args['valid_ds']}/{valid_dir[0]}")
train_human_names = os.listdir(train_human_dir)
train_animals_names = os.listdir(train_animal_dir)
validation_animal_names = os.listdir(valid_animal_dir)

print('total training human images:',
len(os.listdir(train_human_dir)))
print('total training shark images:',
len(os.listdir(train_animal_dir)))
print('total validation human images:',
len(os.listdir(valid_human_dir)))
print('total validation shark images:',
len(os.listdir(valid_animal_dir)))

# Generate new data
train_datagen = ImageDataGenerator(rescale=1/255)
validation_datagen = ImageDataGenerator(rescale=1/255)

train_generator = train_datagen.flow_from_directory(
    args['train_ds'],
    classes=['human', 'shark'],
    target_size=(200, 200),
    batch_size=190,
    class_mode='binary')

validation_generator = validation_datagen.flow_from_directory(
    args['valid_ds'],
    classes=['human', 'shark'],
    target_size=(200, 200),
    batch_size=15,
    class_mode='binary',
    shuffle=False)

# Init new model
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(200, 200, 3)),

```

```
tf.keras.layers.Dense(256, activation=tf.nn.relu),
tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])

# Train model
model.summary()
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='binary_crossentropy',
              metrics=['accuracy'])

history = model.fit(train_generator,
                    steps_per_epoch=15,
                    epochs=25,
                    verbose=1,
                    validation_data=validation_generator,
                    validation_steps=15)

img = load_img(image_path, target_size=(200, 200))
x = img_to_array(img)
plt.imshow(x / 255.)
x = np.expand_dims(x, axis=0)
images = np.vstack([x])
classes = model.predict(images, batch_size=19)
print(classes[0])
if classes[0] < 0.5:
    print(os.path.basename(image_path) + " is a Human")
else:
    print(os.path.basename(image_path) + " is a Shark")
acc_res = history.history['accuracy']
print(
    f"ACC result: {sum(acc_res) / len(acc_res)}")
```