ПРИЛОЖЕНИЕ А

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИИТ

Моделирование алгоритмов обучения и функционирования искусственных нейронных сетей на базе фреймфорка TensorFlow

ТЕКСТ ПРОГРАММЫ
(на оптическом носителе CD-R)

КР.ПО4.190333-03 12 00

Листов 8
Объём 6.6 MiB

Руководитель	Ю. В. Савицкий
Выполнил	П. И. Галанин
Консультант по ЕСПД	Ю. В. Савицкий

2021

Содержание

					КР.ПО4.190333-03 12 00		
Изм	Лист	№ докум.	Подп.	Дата			
Разраб.		Галанин			Моделирование алгоритмов обучения и функционирования искусственных нейронных сетей на базе фреймфорка TensorFlow Текст программы	Лит.	Лист 2 / Листов 8
Пров.		Савицкий				К	
Н. контр.		Савицкий				БрГТУ	
Утв.							

1 Распознование цифры

Файл, реализация которого распознаёт на картинке рукописную цифру.

Листинг: main.py

```
# = = = = = библиотеки = = = = =

# При запуске программы TensorFlow 2+ пытается запустить GPU
# Нам нужен CUDA для GPU TensorFlow
# Чтобы не выводились предупреждения при запуске программы
# пропишем две строчки:
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' #ignore

import numpy
import matplotlib.pyplot as plt
import tensorflow

print(f'TensorFlow verion : {tensorflow.__version__}')
print(f'Keras verion      : {tensorflow.keras.__version__}')

(x_train, y_train), (x_test, y_test) = tensorflow.keras.datasets.mnist.load_data()

# стандартизация входных данных
x_train = x_train / 255
x_test = x_test / 255

y_train_cat = tensorflow.keras.utils.to_categorical(y_train, 10)
y_test_cat = tensorflow.keras.utils.to_categorical(y_test, 10)

# отображение первых 20*24=480 изображений из обучающей выборки
plt.figure(figsize=(10,14)) # размер в дюймах
for i in range(600):
    plt.subplot(20,30,i+1)  # картинки 20 по строке и 24 по столбцу
    plt.xticks([])          # не печатать оси по x
    plt.yticks([])          # не печатать оси по y
    plt.title(y_train[i])   # печатать в заголовке картинки цифру
    plt.imshow(             # печатать в рамке картинку
        x_train[i],
        cmap=plt.cm.binary
    )
plt.show()                  # печатаем картинку в окно

# = = = = = Формируем модель НС и выводим структуру на консоль = = = = =

# Для распознования образов используют сверточную нейронную сеть
# Пока мы ничего не знаем о сверточной нс,
# поэтому будем использовать обычную нейронную сеть
# Создаем модель нейронной сети
model = tensorflow.keras.Sequential([
    # матрица 28 пикселей на 28 пикселей = 784 входных слоёв
    # 28x28 + 1 входной нейрон bias = 785
    # Данный слой преобразует 2D-изображение размером 28x28 пикселей
    #(на каждый пиксель приходится 1 байт для оттенков серого)
    # в 1D-массив состоящий из 784 пикселей.
    tensorflow.keras.layers.Flatten(input_shape=(28, 28, 1)),
    # 0 весов
```

```
    # Скрытый слой из 128 нейронов с функцией активации ReLu
    # Не обязательно брать 128 и один скрытый слой.
    # Скрытых слоев может быть хоть два
    # Нейронов может быть хоть 50
    tensorflow.keras.layers.Dense(128, activation='relu'),
    # (784 нейронов + 1 bias) * 128 нейронов = 100480 весов
    # Выходной слой из 10 нейронов с функцией активации softmax
    # Нейронны класифицируют [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    tensorflow.keras.layers.Dense(10, activation='softmax')
    # (128 нейронов + 1 bias) * 10 нейронов = 1290 весов
])

# Flatten - создаст слой, который будет брать картинку построчно
# Dense - создаст слой, который свяжет нейронны с предыдущим слоем

# Распечатаем сводку по модели, чтобы получить представление о модели в целом
print('Печатаем информацию о модели')
model.summary()       # вывод структуры НС в консоль
print('Информацию о модели распечатана\n')

# = = = = = Компиляция НС с оптимизацией

print('Компилируем модель нейронной сети')
model.compile(
    #optimizer='adam',                    # оптимизация по adam
    optimizer=tensorflow.keras.optimizers.Adam(0.001),
    loss='categorical_crossentropy',     # функция потерь - категориальная кросс-энтропия
    metrics=['accuracy']                 # метрика. Выводит в консоль процент правильно распознаных
    цифр
)
print('Модель нейронной сети скомпилированна\n')

# = = = = = Запуск процесса обучения 80% - обучающая выборка, 20% - выборка валидации = = = = =

# Обучает модель для фиксированного количества эпох (итераций в наборе данных)
model.fit(
    x_train,                # входное обучающее множество
    y_train_cat,            # требуемое значение на выходе
    batch_size=32,          # размер batch'a: после 32 изображений корректируем изображения
    epochs=5,               # колиечство эпох
    validation_split=0.2    # разбиение обучающей выборки и проверочной 0.2 = 20% обучающей выборки
     для валидации
)

# Возвращает значение потерь и значения показателей для модели
# Оцениваем качество обучения на тестовой выборке
model.evaluate(x_test, y_test_cat)

def print_info_about_image_by_index(n):
  x = numpy.expand_dims(
      x_test[n],
      axis=0      # новая ось axis со значением 0
  )

  # Метод model.predict возвращает список списков (массив массивов)
  res = model.predict(x)
  plt.title(f'Распознанная цифра : {numpy.argmax(res)}')
  plt.imshow(x_test[n], cmap=plt.cm.binary)
  plt.show()

```

```
# = = = = = Проверка распознания цифр = = = = =

# 10 раз вызвали функцию
for i in range(0, 10):
  print_info_about_image_by_index(i)

# метод предикт ожидает, что мы подадим несколько изображений
# а если передаем одно, то должны представить его в виде трехмерного тензора,
# и каждым элементом этого тезора будет это изображение

# Распознавание всей тестовой выборки
pred = model.predict(x_test)
pred = numpy.argmax(pred, axis=1)
print(f'Цифры_____________: {pred}')
print(f'Размерность_массива: {pred.shape}')
print(f'Что_предсказала_НС: {pred[:37]}')
print(f'Что_должно_быть____: {y_test[:37]}')

# Выделение неверных вариантов
mask = pred == y_test
x_false = x_test[~mask]
p_false = pred[~mask]
print(f'Размерность_x_false: {x_false.shape}')
print(f'Размерность_p_false: {p_false.shape}')

# Вывод неверных результатов
plt.figure(figsize=(10,10)) # размер в дюймах
for i in range(p_false.shape[0]):
    plt.subplot(13,20,i+1)  # расположить картинки в 13x20
    plt.xticks([])          # не выводить оси по x
    plt.yticks([])          # не выводить оси по y
    plt.title(p_false[i])   # печать в заголовок картинки цифру
    plt.imshow(x_false[i], cmap=plt.cm.binary) # печатает в рамку
plt.show()                  # напечатать картинку
```

2 Коты против собак

Файл, реализация которого распознаёт на картинке либо кошку, либо собаку.

Листинг: main.py

```
# = = = = = библиотеки = = = = =

# При запуске программы TensorFlow 2+ пытается запустить GPU
# Нам нужен CUDA для GPU TensorFlow
# Чтобы не выводились предупреждения при запуске программы
# пропишем две строчки:
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' #ignore

import numpy
import tensorflow
import tensorflow_datasets
import matplotlib.pyplot
#from google.colab import files # для загрузки файлов в Google Colabs

train, _ = tensorflow_datasets.load(
    'cats_vs_dogs',
    split=['train[:1%]'],
    with_info=True,
    as_supervised=True
)

for img, label in train[0].take(10):
    matplotlib.pyplot.figure()
    matplotlib.pyplot.imshow(img)
    class_in_img = 'Dog' if label else 'Cat'
    matplotlib.pyplot.title(class_in_img)
matplotlib.pyplot.show()

SIZE = 224
def resize_image(img, label):
    img = tensorflow.cast(img, tensorflow.float32)
    img = tensorflow.image.resize(img, (SIZE, SIZE))
    img = img / 255.0
    return img, label

train_resized = train[0].map(resize_image)
train_batches = train_resized.shuffle(1000).batch(16)

base_layers = tensorflow.keras.applications.MobileNetV2(input_shape=(SIZE, SIZE, 3),
    include_top=False)
base_layers.trainable = False

model = tensorflow.keras.Sequential([
    base_layers,
    tensorflow.keras.layers.GlobalAveragePooling2D(),
    tensorflow.keras.layers.Dropout(0.2),
    tensorflow.keras.layers.Dense(1)
])
model.compile(
    optimizer='adam',
    loss=tensorflow.keras.losses.BinaryCrossentropy(from_logits=True),
```

```
    metrics=['accuracy']
)

model.fit(train_batches, epochs=1)

f = []
#f = files.upload()

list_img_path = []
for key in f:
  list_img_path.append(key)

for i in range(len(list_img_path)):
    path = list_img_path[i]
    img = tensorflow.keras.preprocessing.imag.load_img(path)
    img_array = tensorflow.keras.preprocessing.imag.img_to_array(img)
    img_resized, _ = resize_image(img_array, _)
    img_expended = numpy.expand_dims(img_resized, axis=0)
    prediction = model.predict(img_expended)[0][0]
    pred_label = 'Cat' if prediction < 0.5 else 'Dog'
    matplotlib.pyplot.figure()
    matplotlib.pyplot.imshow(img)
    matplotlib.pyplot.title(f'{path}\n{pred_label} {prediction}')
    matplotlib.pyplot.show()

model.fit(train_batches, epochs=1)

for i in range(len(list_img_path)):
    path = list_img_path[i]
    img = tensorflow.keras.preprocessing.imag.load_img(path)
    img_array = tensorflow.keras.preprocessing.imag.img_to_array(img)
    img_resized, _ = resize_image(img_array, _)
    img_expended = numpy.expand_dims(img_resized, axis=0)
    prediction = model.predict(img_expended)[0][0]
    pred_label = 'Cat' if prediction < 0.5 else 'Dog'
    matplotlib.pyplot.figure()
    matplotlib.pyplot.imshow(img)
    matplotlib.pyplot.title(f'{path}\n{pred_label} {prediction}')
    matplotlib.pyplot.show()

model.fit(train_batches, epochs=1)

for i in range(len(list_img_path)):
    path = list_img_path[i]
    img = tensorflow.keras.preprocessing.imag.load_img(path)
    img_array = tensorflow.keras.preprocessing.imag.img_to_array(img)
    img_resized, _ = resize_image(img_array, _)
    img_expended = numpy.expand_dims(img_resized, axis=0)
    prediction = model.predict(img_expended)[0][0]
    pred_label = 'Cat' if prediction < 0.5 else 'Dog'
    matplotlib.pyplot.figure()
    matplotlib.pyplot.imshow(img)
    matplotlib.pyplot.title(f'{path}\n{pred_label} {prediction}')
    matplotlib.pyplot.show()
```

3 Цельсий в Фаренгейт

Файл, реализация которого переводит градус Цельсий в Гградус Фаренгейт.

Листинг: main.py

```
# = = = = = библиотеки = = = = =

# При запуске программы TensorFlow 2+ пытается запустить GPU
# Нам нужен CUDA для GPU TensorFlow
# Чтобы не выводились предупреждения при запуске программы
# пропишем две строчки:
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' #ignore

import numpy
import matplotlib.pyplot
import tensorflow

c = numpy.array( [-40, -10,  0,  8, 15, 22,  38] )  # градусы цельсий
f = numpy.array( [-40,  14, 32, 46, 59, 72, 100] )  # градусы фаренгейт

model = tensorflow.keras.Sequential()

model.add(
    tensorflow.keras.layers.Dense(
        units=1,
        input_shape=(1,),
        activation='linear'
    )
)

model.compile(
    loss='mean_squared_error',
    optimizer=tensorflow.keras.optimizers.Adam(0.1)
)

history = model.fit(
    c,
    f,
    epochs=500,
    verbose=0
)
print("Обучение завершено")

mySample = [100, 101, 102]
print(f'Прогноз при выборке : {mySample}')
print(f'Результат           : {model.predict(mySample)}')
print()

print(f'Веса модели         : {model.get_weights()}')

matplotlib.pyplot.plot(history.history['loss'])
matplotlib.pyplot.grid(True)
matplotlib.pyplot.ylabel('Ошибка')
matplotlib.pyplot.xlabel('Итерации')
matplotlib.pyplot.title('Зависимость ошибки от итераций')
matplotlib.pyplot.show()
```