Министерство науки высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

Факультет цифровых трансформаций

Образовательная программа Искусственный интеллект в промышленности

Направление подготовки (специальность) 09.04.02 - Информационные системы и технологии

О Т Ч Е Т

Лабораторной работе №3

Тема задания: Применение инструментов оптимизации моделей и создания сервисов

Обучающихся *Штыкина Ольга, Кузьмина Анна, группа J4151*

Преподаватель: *Старобыховская А.А.*

Санкт-Петербург
2024

СОДЕРЖАНИЕ

ЗАДАНИЕ

Цель задания: Сконвертировать модель в onnx и запустить в BentoML. Сравнить метрики.

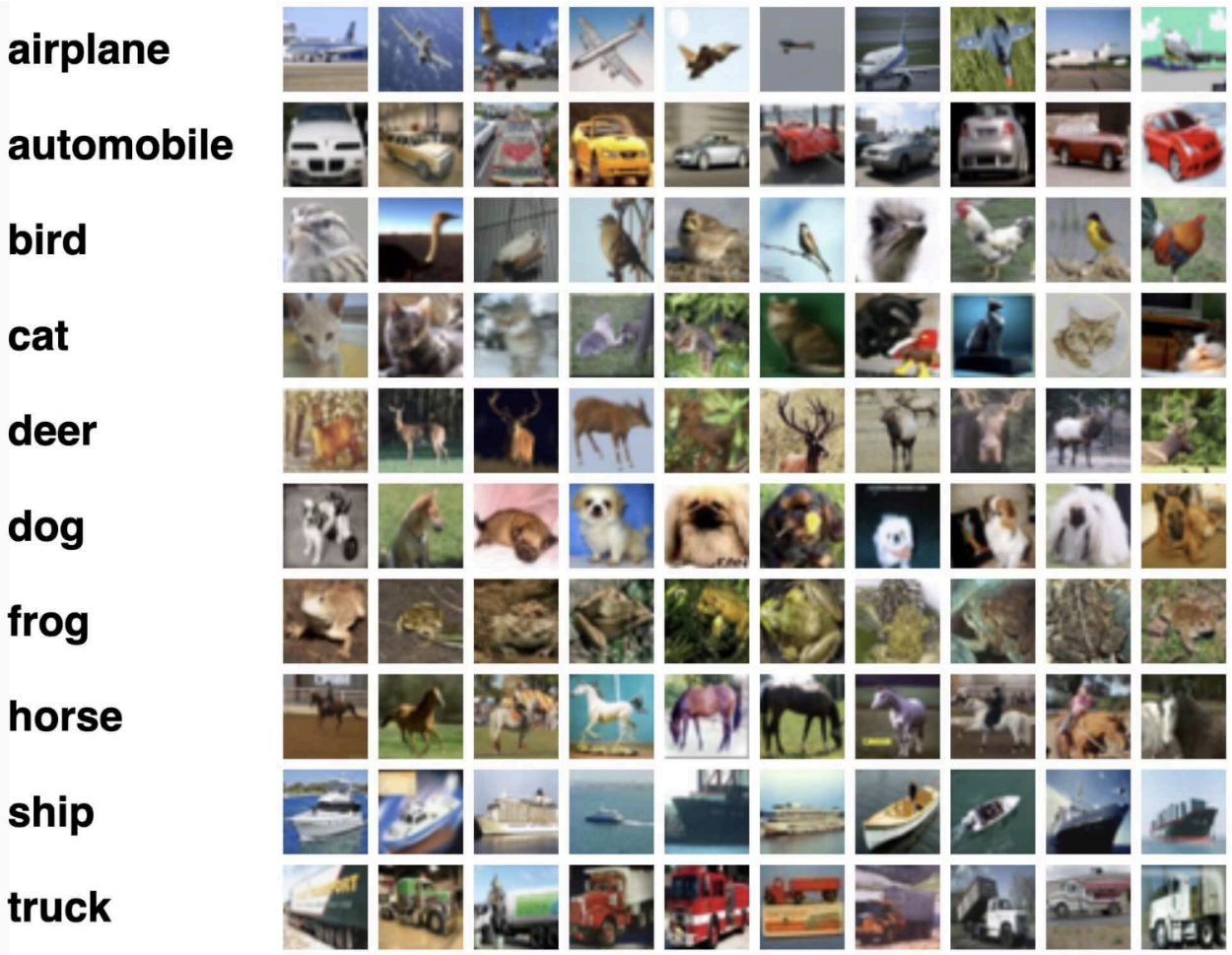
Описание предметной области: анализ параметров и метрик качества при обучении модели ИИ.

Исходные данные: датасет CIFAR100

ОСНОВНЫЕ ЭТАПЫ

1. Базовая модель

В качестве базовой модели была взята простая сверточная нейронная сеть с применением Batch Normalization, оптимизацией Adam и увеличенным количеством эпох до 10, которая показала лучший результат по итогам второй лабораторной.



Примеры корректных кейсов

Примеры ошибочных кейсов

Точность: 42%

2. Конвертация в ONNX

Граф модели, конвертированной в onnx формат:

```
graph main_graph (
  %input[FLOAT, batch_sizex3x32x32]
) initializers (
  %fc1.weight[FLOAT, 512x4096]
  %fc1.bias[FLOAT, 512]
  %fc2.weight[FLOAT, 100x512]
  %fc2.bias[FLOAT, 100]
  %onnx::Conv_33[FLOAT, 32x3x3x3]
  %onnx::Conv_34[FLOAT, 32]
  %onnx::Conv_36[FLOAT, 64x32x3x3]
  %onnx::Conv_37[FLOAT, 64]
) {
  %/conv1/Conv_output_0 = Conv[dilations = [1, 1], group = 1, kernel_shape = [3, 3], pads = [1, 1, 1, 1], strides = [1, 1]](%input, %onnx::Conv_33, %onnx::Conv_34)
  %/Relu_output_0 = Relu(%/conv1/Conv_output_0)
  %/pool/MaxPool_output_0 = MaxPool[ceil_mode = 0, dilations = [1, 1], kernel_shape = [2, 2], pads = [0, 0, 0, 0], strides = [2, 2]](%/Relu_output_0)
  %/conv2/Conv_output_0 = Conv[dilations = [1, 1], group = 1, kernel_shape = [3, 3], pads = [1, 1, 1, 1], strides = [1, 1]](%/pool/MaxPool_output_0, %onnx::Conv_36,
  %/Relu_1_output_0 = Relu(%/conv2/Conv_output_0)
  %/pool_1/MaxPool_output_0 = MaxPool[ceil_mode = 0, dilations = [1, 1], kernel_shape = [2, 2], pads = [0, 0, 0, 0], strides = [2, 2]](%/Relu_1_output_0)
  %/Constant_output_0 = Constant[value = <Tensor>]()
  %/Reshape_output_0 = Reshape(%/pool_1/MaxPool_output_0, %/Constant_output_0)
  %/fc1/Gemm_output_0 = Gemm[alpha = 1, beta = 1, transB = 1](%/Reshape_output_0, %fc1.weight, %fc1.bias)
  %/Relu_2_output_0 = Relu(%/fc1/Gemm_output_0)
  %output = Gemm[alpha = 1, beta = 1, transB = 1](%/Relu_2_output_0, %fc2.weight, %fc2.bias)
  return %output
}
```

Сравнение исходной модели и сконвертированной:

```
Батч 1: Выходные тензоры близки: True
Сравнение предсказаний PyTorch и ONNX:
  Количество совпадающих предсказаний: 1024
  Количество несовпадающих предсказаний: 0
  Процент совпавших предсказаний: 100.00%
----------
Батч 2: Выходные тензоры близки: True
Сравнение предсказаний PyTorch и ONNX:
  Количество совпадающих предсказаний: 1024
  Количество несовпадающих предсказаний: 0
  Процент совпавших предсказаний: 100.00%
----------
```

Качество предсказаний сконвертированной модели такое же, как у исходной, а скорость ответов увеличилась почти в 2 раза.

3. BentoML

Сервис файл, включающий в себя название сервиса, все рабочие файлы и необходимые зависимости:

Локальный bento сервер:

```python
from torchvision import transforms
from PIL import Image
import bentoml
from bentoml.io import Image as BentoImage, JSON
import numpy as np
import asyncio
import nest_asyncio
nest_asyncio.apply()

# Инициализация модели
cifar100_runner = bentoml.onnx.get("model:latest").to_runner()
svc = bentoml.Service("model", runners=[cifar100_runner])

# API для предсказаний
GigaCode: explain | explain step by step | doc | test
@svc.api(input=BentoImage(), output=JSON())
async def predict(img: Image.Image):
    classes = ['apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle',

    # Преобразование изображения
    transform = transforms.Compose([
        transforms.Resize((32, 32)),
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Нормализация как в обучении
    ])
    img_tensor = transform(img).unsqueeze(0).numpy()  # Добавляем batch dimension

    loop = asyncio.get_event_loop()
    predictions = loop.run_until_complete(cifar100_runner.run.async_run(img_tensor))

    print(f"Предсказания модели: {predictions}")
    predicted_class = int(np.argmax(predictions[0]))
    print({"class_id": predicted_class, "class_name": classes[predicted_class]})
    return {"class_id": predicted_class, "class_name": classes[predicted_class]}
```

Сформированный и сохраненный сервис BentoML:

```
INFO: Locking PyPI package versions.
INFO: Locking packages for x86_64-unknown-linux-gnu. Pass `--platform` option to specify the platform.
```

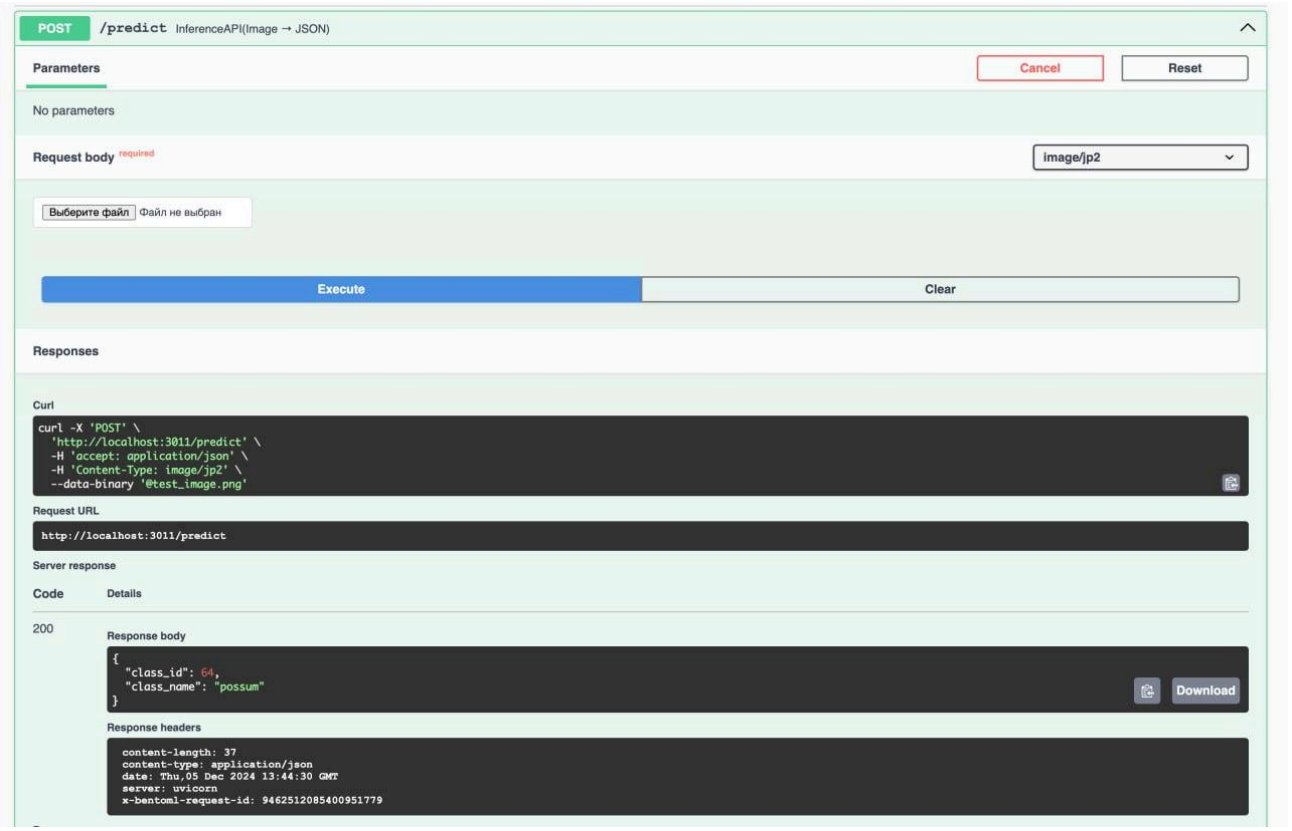
```
BENTOML
```

```
Successfully built Bento(tag="model:aaouxvftb6cjwsuv").

Next steps:

* Deploy to BentoCloud:
    $ bentoml deploy model:aaouxvftb6cjwsuv -n ${DEPLOYMENT_NAME}

* Update an existing deployment on BentoCloud:
    $ bentoml deployment update --bento model:aaouxvftb6cjwsuv ${DEPLOYMENT_NAME}

* Containerize your Bento with `bentoml containerize`:
    $ bentoml containerize model:aaouxvftb6cjwsuv

* Push to BentoCloud with `bentoml push`:
    $ bentoml push model:aaouxvftb6cjwsuv
```

Запущенный Bento сервис:

```
2024-12-05T16:44:14+0300 [INFO] [cli] Environ for worker 0: set CPU thread count to 8
2024-12-05T16:44:14+0300 [INFO] [cli] Prometheus metrics for HTTP BentoServer from "service.py" can be accessed at http://localhost:3011/metrics.
2024-12-05T16:44:14+0300 [INFO] [cli] Starting production HTTP BentoServer from "service.py" listening on http://0.0.0.0:3011 (Press CTRL+C to quit)
```

С помощью сервиса можно удобно загрузить фотографию и посмотреть, как модель ее классифицирует.



ВЫВОДЫ

Конвертация модели в onnx удобный способ передачи или переноса модели в другое место.

BentoML тоже может быть полезным если нужно автоматизировать работу уже готовой модели.