ИТМО

Лабораторная работа №3

по дисциплине: Технологии нейросетевых вычислений

вариант: Plant Segmentation

Выполнили: Миху Вадим

Чернова Анна

P34301

Преподаватель: Старобыховская Анастасия Александровна

Санкт-Петербург

2024

Задание

1. Выполнить Лабораторную работу 1/2
2. Экспортировать модель в представление onnx
3. Запустить через onnxruntime (на cpu/gpu с учетом доступных ресурсов)
4. Проанализировать изменение времени на инференс (сравнивая время инференс 1/2/3 лабораторных)
5. Получить оценки по точности, примеры ошибочных семплов, сделать выводы
6. Создать отчет, прикрепить к заданию
7. Защититься на 8+ баллов

Выполнение

I. Модель

Исходный код модели

```python
def create_unet_model(input_size=(IMG_HEIGHT, IMG_WIDTH, 3)):
    inputs = keras.layers.Input(input_size)

    c1 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
    c1 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(c1)
    p1 = keras.layers.MaxPooling2D((2, 2))(c1)

    c2 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(p1)
    c2 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(c2)
    p2 = keras.layers.MaxPooling2D((2, 2))(c2)

    c3 = keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(p2)
    c3 = keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(c3)

    u1 = keras.layers.UpSampling2D((2, 2))(c3)
    u1 = keras.layers.concatenate([u1, c2])
    c4 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(u1)
    c4 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(c4)

    u2 = keras.layers.UpSampling2D((2, 2))(c4)
    u2 = keras.layers.concatenate([u2, c1])
    c5 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(u2)
    c5 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(c5)

    outputs = keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c5)
    model = keras.Model(inputs, outputs)
    return model
```

II. Оптимизация

```
prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude
pruning_schedule = tfmot.sparsity.keras.PolynomialDecay(
    initial_sparsity=0.2,
    final_sparsity=0.8,
    begin_step=0,
    end_step=1000
)

pruned_model = prune_low_magnitude(base_model, pruning_schedule)

pruned_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

callbacks = [
    tfmot.sparsity.keras.UpdatePruningStep(),
    tfmot.sparsity.keras.PruningSummaries(log_dir='./logs')
]
```

III. Экспорт в ONNX Runtime

```
onnx_model, _ = tf2onnx.convert.from_keras(final_model, output_path="model.onnx")
```

IV. Исходные метрики

Время инференса оригинальной модели

```
import time

# Время предсказания для исходной модели
image = np.expand_dims(X_val[idx], axis=0)

start = time.time()
base_model.predict(image)
time_base = time.time() - start

# Время предсказания для обрезанной модели
start = time.time()
final_model.predict(image)
time_pruned = time.time() - start

print("Время предсказания исходной модели:", time_base, "секунд")
print("Время предсказания обрезанной модели:", time_pruned, "секунд")
```

```
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 160ms/step
Время предсказания исходной модели: 0.16750288009643555 секунд
Время предсказания обрезанной модели: 0.17182588577270508 секунд
```

V. Оптимизация ONNX Runtime

```
import onnx
from onnxruntime import InferenceSession
from onnxruntime_tools import optimizer

model_path = "model.onnx"
model = onnx.load(model_path)

optimized_model = optimizer.optimize_model(model_path)

onnx.save_model(optimized_model.model, "optimized_model.onnx")
```

VI. Сравнение

```
import onnxruntime as ort
import numpy as np
import time

session = ort.InferenceSession("model.onnx")

input_name = session.get_inputs()[0].name

start_time = time.time()
output = session.run(None, {input_name: input_image_expanded})
end_time = time.time()

print(f"Время инференса: {end_time - start_time} секунд")
```

```
Время инференса: 0.18994784355163574 секунд
```

```
session = InferenceSession("optimized_model.onnx")

start_time = time.time()
output = session.run(None, {input_name: input_image_expanded})
end_time = time.time()

print(f"Время инференса: {end_time - start_time} секунд")
```

```
2024-12-16 15:24:16.863682 [W:onnxruntime:Default, upsamplebase
this opset 13 model uses the deprecated attribute
2024-12-16 15:24:16.863696 [W:onnxruntime:Default, upsamplebase
this opset 13 model uses the deprecated attribute
Model producer not matched: Expect pytorch,  Got tf2onnx 1.16.1
2024-12-16 15:24:16.888518 [W:onnxruntime:Default, upsamplebase
this opset 13 model uses the deprecated attribute
2024-12-16 15:24:16.888541 [W:onnxruntime:Default, upsamplebase
this opset 13 model uses the deprecated attribute
Время инференса: 0.1924302577972412 секунд
```

VII. Вывод

Использование рантайма даже со встроенными утилитами оптимизации никак не повлияло на время инференса. Метрики точности соответственно также не изменились, так модель является лишь генерализацией над моделями конкретных фреймворков.