

深度学习系统第六次作业

ZY2206117 黄海浪

1. 实验介绍

实验内容

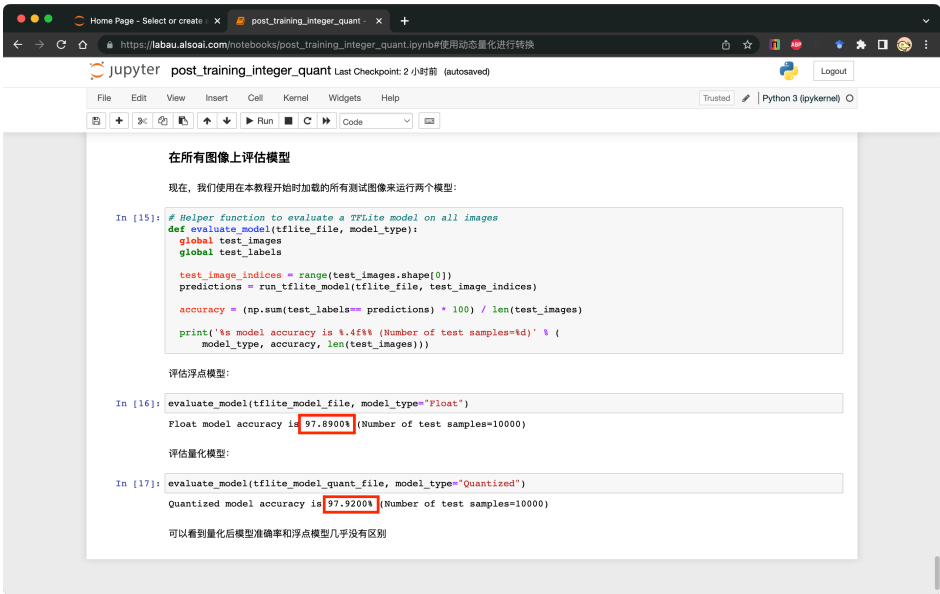
本次实验体验常用的模型轻量化方法（量化、剪枝、蒸馏），感受各方法之间的区别与联系。

实验步骤

1. 量化部分使用 MNIST 数据集从头开始训练一个模型、将其转换为 TensorFlow Lite 文件，并使用训练后量化的方法对其进行量化。最后检查转换后模型的准确率并将其与原始浮点模型进行比较。
2. 剪枝部分使用 MNIST 数据集从头开始训练一个模型，使用剪枝 API 微调模型并查看模型准确率，最后结合剪枝和后训练量化，生成一个是原模型 10 倍小的 TFLite 模型。
3. 蒸馏部分包含教师模型和学生模型，首先使用 MNIST 数据集训练教师模型，接着使用训练后的教师模型指导学生模型的训练过程，最后比较学生模型有无指导情况下的准确率。

2. 模型量化

使用全整数量化



在所有图像上评估模型

现在，我们使用在本教程开始时加载的所有测试图像来运行两个模型：

```
In [15]: # Helper function to evaluate a TFlite model on all images
def evaluate_model(tflite_file, model_type):
    global test_images
    global test_labels

    test_image_indices = range(test_images.shape[0])
    predictions = run_tflite_model(tflite_file, test_image_indices)
    accuracy = (np.sum(test_labels == predictions) * 100) / len(test_images)

    print('%s model accuracy is %.4f%% (Number of test samples=%d)' % (
        model_type, accuracy, len(test_images)))
```

评估浮点模型：

```
In [16]: evaluate_model(tflite_model_file, model_type="Float")
Float model accuracy is 97.8900% (Number of test samples=10000)
```

评估量化模型：

```
In [17]: evaluate_model(tflite_model_quant_file, model_type="Quantized")
Quantized model accuracy is 97.9200% (Number of test samples=10000)
```

可以看到量化后模型准确率和浮点模型几乎没有区别

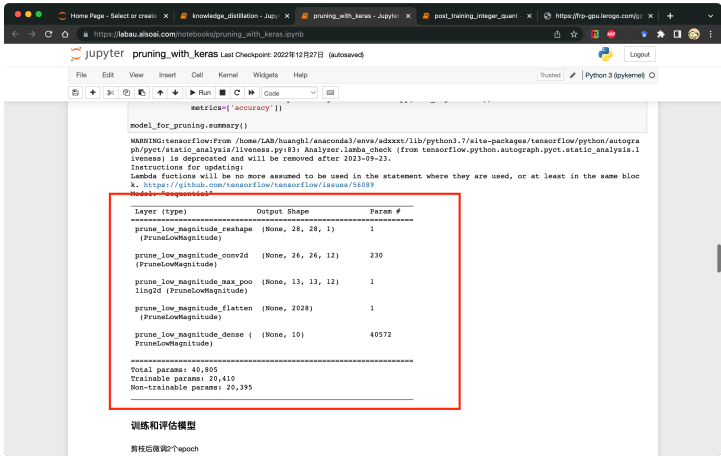
转化前准确率：97.8900%

转化后准确率：97.9200%

从结果可以看到，量化后模型的准确率并没有下降，而是和 float 模型保持相对稳定的准确率。

3. 模型剪枝

剪枝 API 微调模型



```
model_for_pruning.summary()
WARNING:tensorflow:From /home/LAB/huanghl/anaconda/envs/edxxt/lib/python3.7/site-packages/tensorflow/python/autograph/
ph/pyct/static_analysis/liveness.py:83: Analyzer.lamba_check (from tensorflow.python.autograph.pyct.static_analysis.l
iveness) is deprecated and will be removed after 2023-09-23.
Instructions for updating:
Lambda functions will be no more assumed to be used in the statement where they are used, or at least in the same bloc
k. https://github.com/tensorflow/tensorflow/issues/56089
Model: "Sequential"
Layer (type) Output Shape Param #
-----
prune_low_magnitude_reshape (None, 28, 28, 1) 1
prune_low_magnitude_conv2d (None, 26, 26, 12) 230
prune_low_magnitude_max_pooling2d (None, 13, 13, 12) 1
prune_low_magnitude_flatten (None, 2028) 1
prune_low_magnitude_dense (None, 10) 40572
Total params: 40,400
Trainable params: 20,410
Non-trainable params: 20,395
```

训练和评估模型

剪枝后微调20epoch

```
422/422 [=====] - 9s 12ms/step - loss: 0.0915 - accuracy: 0.9761 - val_loss: 0.1125 - val_ac
curacy: 0.9742
Epoch 2/2
422/422 [=====] - 5s 12ms/step - loss: 0.1096 - accuracy: 0.9696 - val_loss: 0.0946 - val_ac
curacy: 0.9745

Out[6]: <keras.callbacks.History at 0x7f6f08e4bd10>

对比剪枝后的准确率

In [7]: _, model_for_pruning_accuracy = model_for_pruning.evaluate(
        test_images, test_labels, verbose=0)

        print('Baseline test accuracy:', baseline_model_accuracy)
        print('Pruned test accuracy:', model_for_pruning_accuracy)

        Baseline test accuracy: 0.9786999821662903
        Pruned test accuracy: 0.9708999991416931

从日志中可以观察每层的稀疏度

In [8]: #docs_infra: no_execute
        # $tensorboard --logdir={logdir}
        print(logdir)

        ../tmp/tmp54zof5jq

在命令行中输入以下命令打开tensorboard可观察模型信息（其中路径请使用上文中打印的log_dir）
...
tensorboard --logdir=../tmp/tmpn1m5a6yl
...

使用剪枝生成一个3倍小的模型

首先，为TensorFlow创建一个可压缩的模型
```

baseline: 0.9786999821662903

剪枝后：0.9708999991416931

从结果可以看到，模型剪枝后，比起 baseline 结果并无较大差距，性能几乎一致。

通过剪枝将 TF 和 TFLite 模型大小缩减为原来的 3 倍

```
In [11]: def get_gzipped_model_size(file):
        import os
        import zipfile

        _, zipped_file = tempfile.mktemp('.zip', dir='../tmp/')
        with zipfile.ZipFile(zipped_file, 'w', compression=zipfile.ZIP_DEFLATED) as f:
            f.write(file)

        return os.path.getsize(zipped_file)

对比模型大小，可以看到小了3倍

In [12]: print("Size of gzipped baseline Keras model: %.2f bytes" % (get_gzipped_model_size(keras_file)))
        print("Size of gzipped pruned Keras model: %.2f bytes" % (get_gzipped_model_size(pruned_keras_file)))
        print("Size of gzipped pruned TFLite model: %.2f bytes" % (get_gzipped_model_size(pruned_tflite_file)))

        Size of gzipped baseline Keras model: 78195.00 bytes
        Size of gzipped pruned Keras model: 25877.00 bytes
        Size of gzipped pruned TFLite model: 25217.00 bytes

使用剪枝和量化创建一个10倍小的模型

In [13]: converter = tf.lite.TFLiteConverter.from_keras_model(model_for_export)
        converter.optimizations = [tf.lite.Optimize.DEFAULT]
        quantized_and_pruned_tflite_model = converter.convert()

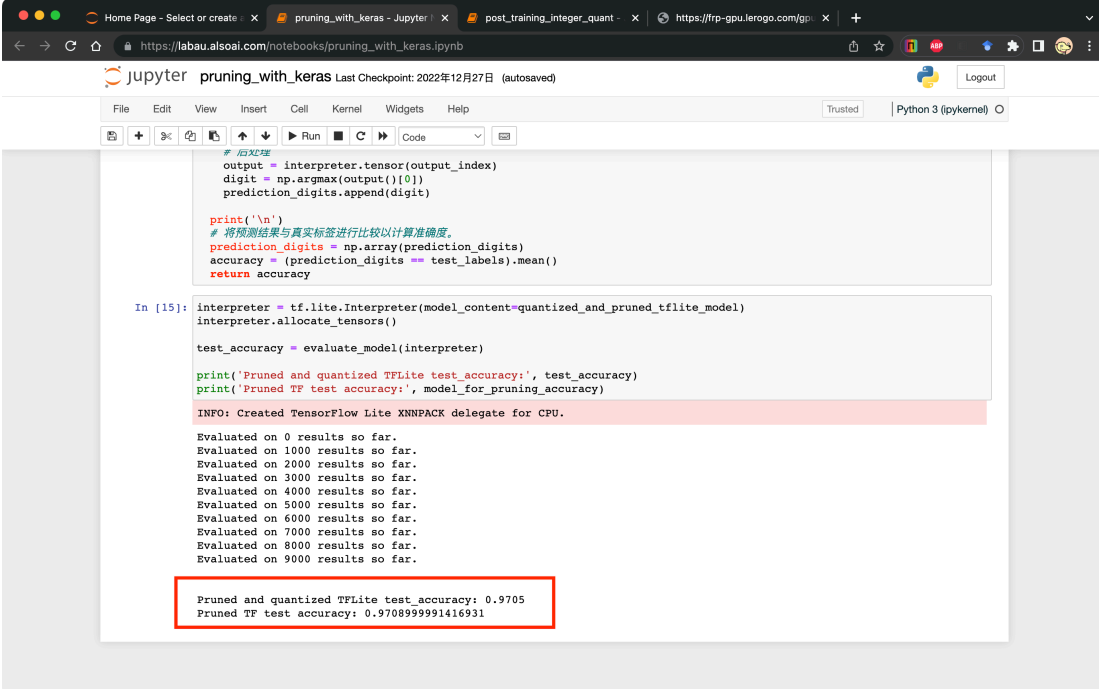
        _, quantized_and_pruned_tflite_file = tempfile.mktemp('.tflite', dir='../tmp/')

        with open(quantized_and_pruned_tflite_file, 'wb') as f:
            f.write(quantized_and_pruned_tflite_model)

        print('Saved quantized and pruned TFLite model to:', quantized_and_pruned_tflite_file)

        print("Size of gzipped baseline Keras model: %.2f bytes" % (get_gzipped_model_size(keras_file)))
        print("Size of gzipped pruned and quantized TFLite model: %.2f bytes" % (get_gzipped_model_size(quantized_and_pruned_tflite_file)))
```

结合剪枝和后训练量化，生成一个原模型 10 倍小的 TFLite 模型



```
# 后处理
output = interpreter.tensor(output_index)
digit = np.argmax(output[0])
prediction_digits.append(digit)

print('\n')
# 将预测结果与真实标签进行比较以计算准确度。
prediction_digits = np.array(prediction_digits)
accuracy = (prediction_digits == test_labels).mean()
return accuracy

In [15]: interpreter = tf.lite.Interpreter(model_content=quantized_and_pruned_tflite_model)
interpreter.allocate_tensors()

test_accuracy = evaluate_model(interpreter)

print('Pruned and quantized TFLite test accuracy:', test_accuracy)
print('Pruned TF test accuracy:', model_for_pruning_accuracy)

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.

Evaluated on 0 results so far.
Evaluated on 1000 results so far.
Evaluated on 2000 results so far.
Evaluated on 3000 results so far.
Evaluated on 4000 results so far.
Evaluated on 5000 results so far.
Evaluated on 6000 results so far.
Evaluated on 7000 results so far.
Evaluated on 8000 results so far.
Evaluated on 9000 results so far.

Pruned and quantized TFLite test accuracy: 0.9705
Pruned TF test accuracy: 0.9708999991416931
```

剪枝和量化后结果：0.9705

baseline Keras model: 78195.00 bytes

pruned and quantized TFLite model: 8130.00 bytes

从结果可以看到，生成原模型 10 倍小的 TFLite 模型性能与 baseline 相比并没有多大差别，但是模型大小可以大大减小，减少了资源占用和计算量。

4. 模型蒸馏

老师模型：

老师模型为了得到更高的准确率，实验将代码的 epochs 改为 10，进行训练，准确率为 0.9781

Home Page - Select or create x knowledge_distillation - Jup... pruning_with_keras - Jupyter x post_training_integer_quant x https://frp-gpu.lerogo.com/gp... x +

https://lab.au.alsoai.com/notebooks/knowledge_distillation.ipynb

jupyter knowledge_distillation Last Checkpoint: 6 分钟前 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

训练教师模型

在知识蒸馏中，我们假设教师模型是训练完成的。因此，我们首先以通常的方式在训练集上训练教师模型。

```
In [11]: # 正常训练教师模型
teacher.compile(
    optimizer=keras.optimizers.Adam(),
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

teacher.fit(x_train, y_train, epochs=10)
teacher.evaluate(x_test, y_test)
```

```
Epoch 1/10
1875/1875 [=====] - 17s 8ms/step - loss: 0.1416 - sparse_categorical_accuracy: 0.9560
Epoch 2/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0913 - sparse_categorical_accuracy: 0.9725
Epoch 3/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0837 - sparse_categorical_accuracy: 0.9756
Epoch 4/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0726 - sparse_categorical_accuracy: 0.9793
Epoch 5/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0746 - sparse_categorical_accuracy: 0.9794
Epoch 6/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0668 - sparse_categorical_accuracy: 0.9820
Epoch 7/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0689 - sparse_categorical_accuracy: 0.9825
Epoch 8/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0632 - sparse_categorical_accuracy: 0.9848
Epoch 9/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0596 - sparse_categorical_accuracy: 0.9855
Epoch 10/10
1875/1875 [=====] - 15s 8ms/step - loss: 0.0669 - sparse_categorical_accuracy: 0.9856
313/313 [=====] - 2s 6ms/step - loss: 0.1255 - sparse_categorical_accuracy: 0.9781

Out[11]: [0.1255345642566681, 0.978100018119812]
```

将教师模型的知识蒸馏给学生

蒸馏模型：

蒸馏模型训练了 3 个 epoch，准确率为 0.9798

Home Page - Select or create x knowledge_distillation - Jup... pruning_with_keras - Jupyter x post_training_integer_quant x https://frp-gpu.lerogo.com/gp... x +

https://lab.au.alsoai.com/notebooks/knowledge_distillation.ipynb

jupyter knowledge_distillation Last Checkpoint: 6 分钟前 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

将教师模型的知识蒸馏给学生

我们已经训练了教师模型，我们只需要初始化一个 `Distiller(student, teacher)` 实例，`compile()` 其带有的所需损失、超参数和优化器，将教师的知识蒸馏给学生。

```
In [13]: # 初始化并 compile distiller
distiller = Distiller(student=student, teacher=teacher)
distiller.compile(
    optimizer=keras.optimizers.Adam(),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
    student_loss_fn=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    distillation_loss_fn=keras.losses.KLDivergence(),
    alpha=0.1,
    temperature=10,
)

# 蒸馏教师模型知识给学生
distiller.fit(x_train, y_train, epochs=3)

# 在测试集上评估学生模型准确率
distiller.evaluate(x_test, y_test)
```

```
Epoch 1/3
1875/1875 [=====] - 19s 10ms/step - sparse_categorical_accuracy: 0.9771 - student_loss: 0.1285 - distillation_loss: 0.0608
Epoch 2/3
1875/1875 [=====] - 18s 10ms/step - sparse_categorical_accuracy: 0.9823 - student_loss: 0.0951 - distillation_loss: 0.0306
Epoch 3/3
1875/1875 [=====] - 18s 10ms/step - sparse_categorical_accuracy: 0.9837 - student_loss: 0.0851 - distillation_loss: 0.0267
313/313 [=====] - 2s 6ms/step - sparse_categorical_accuracy: 0.9798 - student_loss: 0.1085

Out[13]: [0.9797999858856201, 2.5468541934969835e-05]
```

从头开始训练学生模型用于对比

单独训练学生模型：

单独训练学生也为 3 个 epoch，准确率为 0.9710

The screenshot shows a Jupyter Notebook interface with the following content:

```
313/313 [=====] - 2s 6ms/step - sparse_categorical_accuracy: 0.9798 - student_loss: 0.1085
```

Out[13]: [0.9797999858856201, 2.5468541934969835e-05]

从头开始训练学生模型用于对比

我们可以训练一个没有教师模型指导的学生模型，用于对比蒸馏结果从而验证蒸馏方法的有效性

```
In [18]: # 正常训练学生模型
student_scratch.compile(
    optimizer=keras.optimizers.Adam(),
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

# 从头训练并评估学生模型
student_scratch.fit(x_train, y_train, epochs=3)
student_scratch.evaluate(x_test, y_test)
```

```
Epoch 1/3
1875/1875 [=====] - 17s 8ms/step - loss: 0.2426 - sparse_categorical_accuracy: 0.9292
Epoch 2/3
1875/1875 [=====] - 16s 8ms/step - loss: 0.0916 - sparse_categorical_accuracy: 0.9722
Epoch 3/3
1875/1875 [=====] - 16s 8ms/step - loss: 0.0736 - sparse_categorical_accuracy: 0.9774
313/313 [=====] - 2s 6ms/step - loss: 0.0873 - sparse_categorical_accuracy: 0.9710
```

Out[18]: [0.087277352809906, 0.9710000157356262]

在本教程中可以观察到蒸馏后的学生模型效果比不进行蒸馏的学生模型效果更好。（结果根据不同的权重初始化值将会有所波动）

从结果可以看到，通过教师蒸馏得到的模型性能比单独训练模型效果要好。但是由于数据集比较简单，差距并不是很大。不过该方法是很好且常用的方法。

5. 小结

网络剪枝的主要思想就是将权重矩阵中相对“不重要”的权值剔除，然后再重新 fine tune 网络进行微调。随机剪枝方法对硬件非常不友好，往往在硬件实现的过程中不一定能够很好地对网络起到加速和压缩的效果，而且一些标准化模块 conv2d 可能都不能用。后来就使用成块出现的结构化剪枝，包括 Filter Pruning，梯度 Pruning 等方法。量化则可以分为低比特量化、总体训练加速量化和分布式训练梯度量化，本次实验使用的是低比特量化。知识蒸馏将深度网络中所学到的知识转移到另一个相对简单的网络中去，这种方法对于没有标签的数据集也可以拿来训练。另外大模型的输出相较于 GTlabel 来说包含更多信息，如类间距和类内方差等，所以这也可以作为数据增广的一种手段。

网络量化和剪枝是属于后端压缩的方法，这种方法不可逆，而蒸馏的方法属于前端压缩的方法，该方法通过复杂网络去监督简单网络对模型进行简化，是可逆的。这些方法的目的都是最大程度的减小模型复杂度，减少模型存储需要的空间，也致力于加速模型的训练和推测。