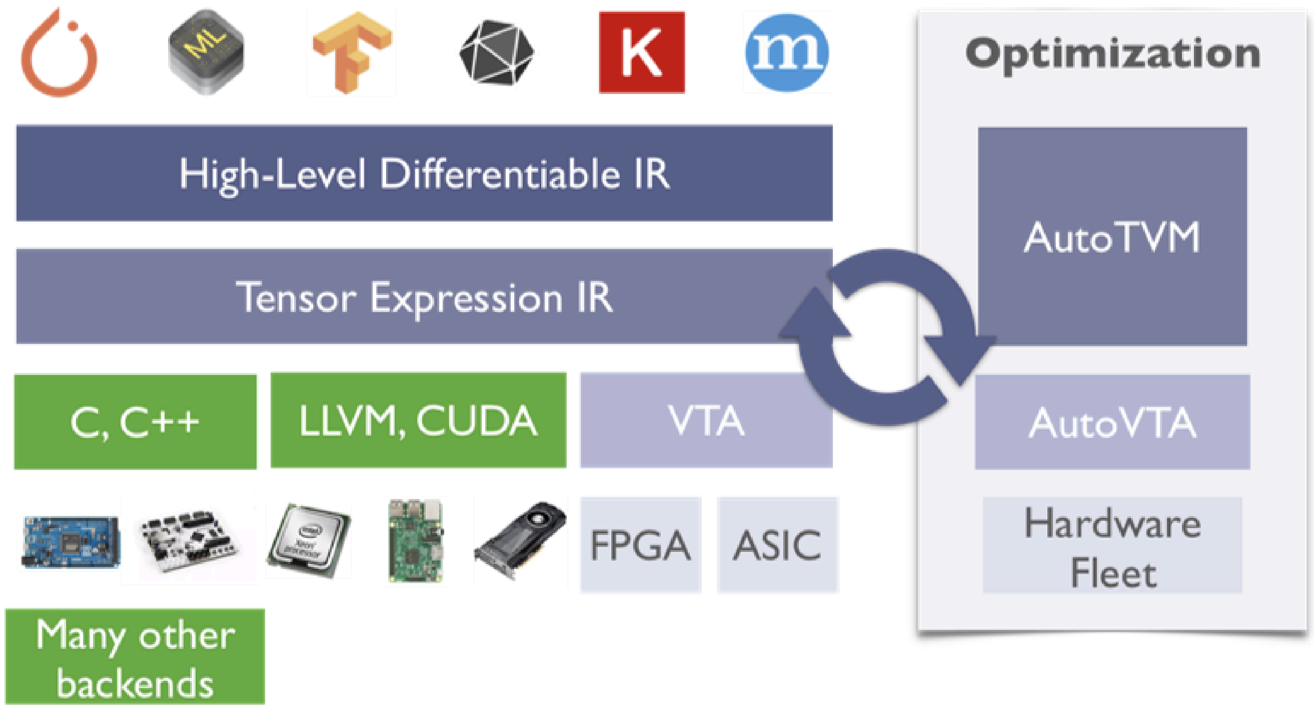
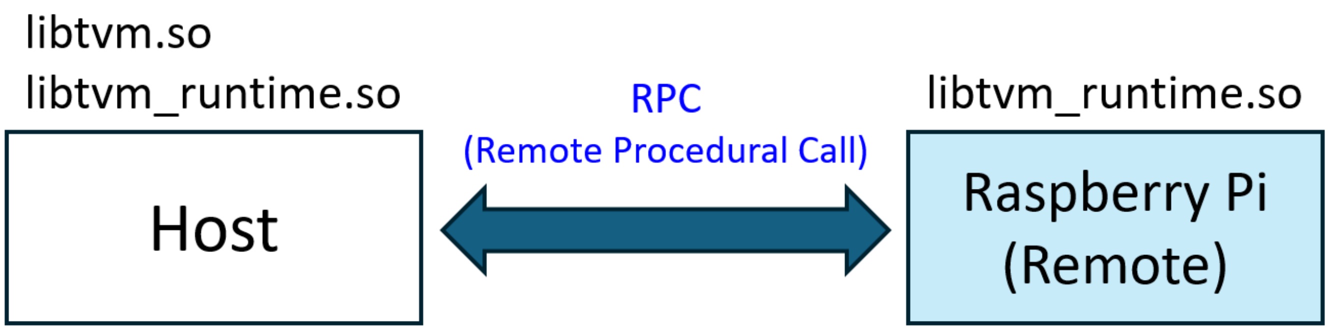
1. 從原始碼安裝 TVM

說明如何從原始碼構建並安裝 TVM 套件。

實驗模組作者：王泓翔、高效能計算實驗室

若是做了有問題，可以email聯絡課程助教。





目錄

- Step 5. Validate Installation

Step 1. Install Dependencies

為了成功編譯 TVM，請先安裝以下：

- CMake (>= 3.18.0)
- LLVM (建議 >= 14.0.0)
- 支援 C++ 17 的現代 C++ 編譯器
  - GCC 7.1
  - Clang 5.0
- Git
- Python (>= 3.8)
- **建議避免使用 conda 建立虛擬環境，因為在安裝或編譯過程中，可能會遇到相依性版本不一致的問題。推薦使用 Python 內建的 venv。**

```
$ sudo apt-get install -y python3 python3-dev python3-setuptools gcc git \
libtinfo-dev zlib1g-dev build-essential cmake libedit-dev libxml2-dev
```

Step 2. Get the Source from the Official Site

下載 0.18.0 版本

```
$ wget https://dlcdn.apache.org/tvm/tvm-v0.18.0/apache-tvm-src-v0.18.0.tar.gz
Or
Use browser to https://downloads.apache.org/tvm/ to download the file.

$ tar zxvf apache-tvm-src-v0.18.0.tar.gz
```

Step 3. Configure and Build

```
cd tvm
rm -rf build && mkdir build && cd build
cp ../cmake/config.cmake .
```

修改 `config.cmake`：(可編輯檔案: config.cmake，搜尋下面的變數設定，直接修改，或透過下面方式修改)

```
$ echo "set(USE_LLVM ON)" >> config.cmake
$ echo "set(HIDE_PRIVATE_SYMBOLS ON)" >> config.cmake
$ echo "set(USE_CUDA   OFF)" >> config.cmake
$ echo "set(USE_METAL  OFF)" >> config.cmake
$ echo "set(USE_VULKAN OFF)" >> config.cmake
$ echo "set(USE_OPENCL OFF)" >> config.cmake
$ echo "set(USE_CUBLAS OFF)" >> config.cmake
$ echo "set(USE_CUDNN  OFF)" >> config.cmake
$ echo "set(USE_CUTLASS OFF)" >> config.cmake
$ echo "set(USE_GRAPH_EXECUTOR ON)" >> config.cmake
$ echo "set(USE_PROFILER ON)" >> config.cmake
$ echo "set(CMAKE_BUILD_TYPE Release)" >> config.cmake
```

開始編譯 TVM： (使用Niniga編譯，可以避免一些問題，縮短編譯時間)

```
$ cmake .. -G Ninja
$ ninja
```

成功Build TVM 後，需要在檔案中設定環境變數~/.bashrc。`path-to-tvm`是TVM source code的位置。

```
export TVM_HOME=/path-to-tvm
export PYTHONPATH=$TVM_HOME/python:$PYTHONPATH
```

Step 4. Extra Python Dependencies

- Install Python pip

```
$ sudo -s
$ apt install python3-pip
```

- Create a virtual environment that is located at the directory `/home/pschen/tvmenv`. Please modify it according to your requirements.

```
$ python3 -m venv /home/pschen/tvmenv
```

- Activate the virtual environment. Please properly modify for your virtual environment location.

```
$ . /home/pschen/tvmenv/bin/activate
```

- Install the necessary dependencies.

```
$ pip install numpy decorator attr typing-extensions psutil scipy packaging pil \
torchvision
```

- 我們需要使用 RPC Tracker.

```
$ pip install tornado
```

Step 5. Validate Installation

- 執行下面的python program，看是否可以正常輸出結果。

```
import tvm
print(tvm.__version__)
```

---

2. Deploy the Pretrained Model on Raspberry Pi

說明如何在Raspberry Pi 上 Build TVM Runtime 和啟動 RPC Server。

目錄

Step 1. Get the Source from the Official Site

下載 0.18.0 版本

```
$ wget https://dlcdn.apache.org/tvm/tvm-v0.18.0/apache-tvm-src-v0.18.0.tar.gz
Or
Use browser to https://downloads.apache.org/tvm/ to download the file.

$ tar zxvf apache-tvm-src-v0.18.0.tar.gz
```

Step 2. Build TVM Runtime on Device

在 Raspberry pi 上 Build TVM Runtime：

```
$ cd <tvm-source>
$ mkdir build
$ cp cmake/config.cmake build
$ cd build
$ cmake .. -G Ninja
$ ninja runtime
```

成功Build TVM Runtime後，需要在檔案中設定環境變數~/.bashrc

```
$ export PYTHONPATH=$PYTHONPATH:<tvm-source>/python
```

更新環境變數後，請執行。

```
$ source ~/.bashrc
```

Step 3. Set Up RPC Server on Device

在 Raspberry pi 上啟動 RPC Server

```
$ python -m tvm.exec.rpc_server --host 0.0.0.0 --port=9090
or
$ python3 -m tvm.exec.rpc_server --host 0.0.0.0 --port=9090
```

如果成功啟動 RPC Server 會看到以下訊息:

```
INFO:root:RPCServer: bind to 0.0.0.0:9090
```

3. Example Program Testing

範例程式：使用 TVM RPC 在遠端裝置上執行 ResNet-18 模型

本程式示範以下流程：

1. 使用 PyTorch 的 ResNet-18 前處理方法載入並處理一張貓的圖片。
2. 在本地電腦上使用 TVM cross-compile ResNet-18 模型，針對 Raspberry Pi 的架構產生可執行模組。
3. 透過 TVM RPC 將編譯後的模型 .tar 檔案傳送至 Raspberry Pi。
4. 在遠端裝置上執行 Inference，取得輸出結果並回傳至本地電腦。

此範例展示如何在資源有限的嵌入式裝置上部署深度學習模型，並使用 TVM 的 RPC 機制遠端執行推論。

本流程中模型是在本機完成編譯的，遠端僅負責推論執行，可大幅減少嵌入式裝置的負擔。

```
import numpy as np
import tvm
from tvm import te, relay, rpc
from tvm.contrib import utils, graph_executor
import tvm.relay.testing
from tvm.contrib.download import download_testdata
from torchvision.models import resnet18, ResNet18_Weights
from PIL import Image

def get_cat_image():
    url =
"https://gist.githubusercontent.com/zhreshold/bcda4716699ac97ea44f791c24310193/raw/fa7ef0e9c
9a5daea686d6473a62aacd1a5885849/cat.png"
    dst = "cat.png"
    real_dst = download_testdata(url, dst, module="data")
    img = Image.open(real_dst).resize((224, 224))

    # Preprocess the image using PyTorch's ResNet-18 weights
    weights = ResNet18_Weights.DEFAULT
    preprocess = weights.transforms()
    img_tensor = preprocess(img).unsqueeze(0) #(3, 224, 224) -> (1, 3, 224, 224)

    # Convert to numpy array with the correct data type
    img_array = img_tensor.numpy()
    return np.asarray(img_array, dtype="float32")

#----------------------------------------
batch_size = 1
num_class = 1000
image_shape = (3, 224, 224)
data_shape = (batch_size,) + image_shape

#
# get_workload: Retrieves a pre-defined ResNet-18 model in Relay's intermediate
#               representation, along with its parameters.
# mod: Represents the computational graph of the model.
# params: Contains the model's parameters, such as weights and biases.
#
mod, params = relay.testing.resnet.get_workload(
    num_layers=18, batch_size=batch_size, image_shape=image_shape
)

#----------------------------------------
```

```python
# Compilation configuration.
#
local_demo = False

if local_demo:
    target = "llvm"
else:
    # target ARM-based devices, such as Raspberry Pi.
    target = "llvm -mtriple=aarch64-linux-gnu"

#----------------------------------------
# Compile the model using TVM.
#
with tvm.transform.PassContext(opt_level=3):
    lib = relay.build(mod, target, params=params)

#
# save the compiled library at a local temp folder.
#
temp = utils.tempdir()
path = temp.relpath("lib.tar")
lib.export_library(path)


#----------------------------------------
# Establish RPC connection.
#
if local_demo:
    remote = rpc.LocalSession()
else:
    # The following is my environment, change this to the IP address of your target device
    host = "Raspberry Pi IP address"
    port = 9090
    remote = rpc.connect(host, port)


# upload the library to the remote device.
remote.upload(path)

# Load the library on the remote device, making it ready for execution.
rlib = remote.load_module("lib.tar")


#----------------------------------------
# Execute the model.
#

# Specifie the CPU device on the remote machine.
dev = remote.cpu(0)

# Wrap the loaded module for execution.
module = graph_executor.GraphModule(rlib["default"](dev))
```

```
# Set input data and parameters.
module.set_input("data", get_cat_image(), **params)

# Execute the model.
module.run()

# Get output
out = module.get_output(0).numpy()
print("output", out)
```

Output:

```
[[0.00091911 0.00103146 0.00092205 0.00104711 0.00107227 0.00106975
  0.00104817 0.00095867 0.00108485 0.00115485 0.00089056 0.00100619
  0.00098092 0.00092844 0.00113992 0.00095253 0.00107877 0.00092991
  0.00105469 0.00093011 0.00106876 0.00113837 0.00102562 0.00105526
  0.00106317 0.00087647 0.00098837 0.00086262 0.00104102 0.00119332
  0.00100116 0.00097005 0.00098161 0.00096744 0.00110561 0.00097907
  0.00094669 0.00095704 0.00103643 0.00107123 0.00111971 0.00107182
  0.00100056 0.00108396 0.0012215  0.0009167  0.0008263  0.00100674
  0.00088927 0.00098282 0.00109451 0.00090901 0.00100885 0.00105487
  0.00089944 0.00106309 0.00102808 0.00097973 0.00099066 0.00113175
  0.001027   0.00095957 0.0008951  0.00099928 0.00106457 0.00112521
  0.00111821 0.00084777 0.0010425  0.00082684 0.00129191 0.00093949
]]
```

---

4. DEMO

完成以下問題，簡述你的實作方法，並繳交最終程式碼。**你需要閱讀程式碼並自行搜尋相關資料以完成此作業**。

1. 請截圖成功執行程式的結果。(15%)
2. 比較不同 TVM `opt_level` 的效能（opt_level = 0, 1, 2, 3），使用 Python所提供的`time.time` 分別計算 `relay.build(mod, target, params)` 的編譯時間與 `module.run()` 的執行時間。請嘗試說明每個 opt_level 對模型進行了哪些優化，並觀察 build 時間與 run 時間的差異。(30%)
3. 比較 TVM `opt_level` 為 0 與 3 時，在不同優化下的記憶體使用情況。請透過下列程式碼觀察記憶體變化，並簡要分析 build 階段與 run 階段的記憶體使用差異。(30%)

   ```
   build_rss = psutil.Process().memory_info().rss / (1024 * 1024)
   print(f"Build memory usage: {build_rss:.2f} MB")
   ```

4. 修改 `local_demo` 變數為 `True`，在本地執行推論，記錄並比較本地（local）與遠端（remote）執行的時間差異。 (15%)
5. 心得(10%)