# Các lệnh linux

* Chuyển sang thư mục d:

cd /mnt/d

* Lệnh biên dịch:

nvcc reduceNeighbored.cu -o a

* Chạy profiling với nsys:

nsys profile ./a

# Lệnh conda

# Các lệnh linux

* Chuyển sang thư mục d:

cd /mnt/d

* Lệnh biên dịch:

nvcc reduceNeighbored.cu -o a

* Chạy profiling với nsys:

nsys profile ./a

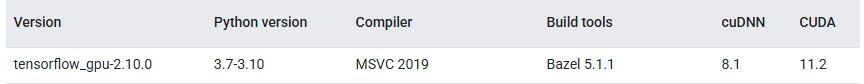
# Lệnh conda

* List env:

Conda env list

* Check version tương thích giữa tensorflow và python và cuda và cudnn trên window

https://www.tensorflow.org/install/source\_windows#gpu



* Install tensorflow tương thích với gpu:

conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0

pip install "tensorflow<2.11"

pip install "numpy<2.0"

* Check tf và active gpu thành công:

python -c "import tensorflow as tf; print(tf.config.list\_physical\_devices('GPU'))"

export môi trường để dùng sau:

conda env export > yolov5-tflite-env.yaml

Import lại môi trường:

conda env create -f yolov5-tflite-env.yaml

<https://fpt-software.udemy.com/course/tensorflow-developer-certificate-machine-learning-zero-to-mastery/learn/lecture/24874232#questions>

* <https://github.com/mrdbourke/tensorflow-deep-learning/blob/main/02_neural_network_classification_in_tensorflow.ipynb>

# MH

* Download:

git clone <https://github.com/hunglc007/tensorflow-yolov4-tflite.git>

* Tải trọng số (Pre-trained Weights)

# Tải trọng số gốc của YOLOv4 (định dạng Darknet)

wget [https://github.com/AlexeyAB/darknet/releases/download/darknet\_yolo\_v3\_optimal/yolov4.weights -O data/yolov4.weights](https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights%20-O%20data/yolov4.weights)

* Chuyển đổi trọng số sang định dạng TensorFlow

Repo này cung cấp script để chuyển đổi file .weights sang định dạng SavedModel của TensorFlow, giúp việc load model nhanh hơn.

python save\_model.py --weights ./data/yolov4.weights --output ./checkpoints/yolov4-416 --input\_size 416 --model yolov4

* + Lệnh này sẽ tạo một thư mục checkpoints/yolov4-416 chứa mô hình đã chuyển đổi.
  + Nếu bạn muốn dùng YOLOv4-tiny hoặc mô hình khác, thay đổi tham số --model và đường dẫn file --weights tương ứng.

# Note

tách bỏ các vòng lặp trong mã nguồn C chính là bước đầu tiên để chuyển đổi thành mã kernel CUDA C.

* Khi cấp phát bộ nhớ trên GPU, device pointer không truy xuất dữ liệu được trong host => nếu cố gắng truy xuất (dereference) thì bị segmentation fault hoặc undefind behavior. => Cách xử lý:
  + Copy về host
  + Truy cập data bằng kernel
  + Dùng unified memory\
* Trong c có loop qua array, thì trong kernel, array index được thay bằng thread coordinate variable.
* Khi cộng số n vào 1 con trỏ int\*, ta đang hiểu dịch con trỏ đi n vị trí trong mảng
* clang-format -i reduceNeighbored.cu
* <https://fpt-software.udemy.com/course/tensorflow-developer-certificate-machine-learning-zero-to-mastery/learn/lecture/24874232#questions>
* <https://github.com/mrdbourke/tensorflow-deep-learning/blob/main/02_neural_network_classification_in_tensorflow.ipynb>

# MH

* Download:

git clone <https://github.com/hunglc007/tensorflow-yolov4-tflite.git>

* Tải trọng số (Pre-trained Weights)

# Tải trọng số gốc của YOLOv4 (định dạng Darknet)

wget [https://github.com/AlexeyAB/darknet/releases/download/darknet\_yolo\_v3\_optimal/yolov4.weights -O data/yolov4.weights](https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights%20-O%20data/yolov4.weights)

* Chuyển đổi trọng số sang định dạng TensorFlow

Repo này cung cấp script để chuyển đổi file .weights sang định dạng SavedModel của TensorFlow, giúp việc load model nhanh hơn.

python save\_model.py --weights ./data/yolov4.weights --output ./checkpoints/yolov4-416 --input\_size 416 --model yolov4

* + Lệnh này sẽ tạo một thư mục checkpoints/yolov4-416 chứa mô hình đã chuyển đổi.
  + Nếu bạn muốn dùng YOLOv4-tiny hoặc mô hình khác, thay đổi tham số --model và đường dẫn file --weights tương ứng.

# Note

tách bỏ các vòng lặp trong mã nguồn C chính là bước đầu tiên để chuyển đổi thành mã kernel CUDA C.

* Khi cấp phát bộ nhớ trên GPU, device pointer không truy xuất dữ liệu được trong host => nếu cố gắng truy xuất (dereference) thì bị segmentation fault hoặc undefind behavior. => Cách xử lý:
  + Copy về host
  + Truy cập data bằng kernel
  + Dùng unified memory\
* Trong c có loop qua array, thì trong kernel, array index được thay bằng thread coordinate variable.
* Khi cộng số n vào 1 con trỏ int\*, ta đang hiểu dịch con trỏ đi n vị trí trong mảng
* clang-format -i reduceNeighbored.cu

# Conda và vscode ko đồng nhất môi trường

**Ctrl shift P => Python: Select Interpreter => Chọn env mong muốn**

# Khởi động vscode, cursor từ anaconda terminal

**Vscode:**

**Chuyển từ ổ C sang ổ D:**

**D:**

**Cd đến thư mục cần khởi động.**

**Khởi động vscode:**

**code .**

**Cursor:**

**Chuyển từ ổ C sang ổ D:**

**D:**

**Cd đến thư mục cần khởi động**

**Khởi động cursor:**

**"C:\Users\PC\AppData\Local\Programs\cursor\Cursor.exe"**

**Laptop : “C:\Users\xungv\AppData\Local\Programs\cursor\ Cursor.exe”**