

# TensorRT & AIoT

Edge AI Acceleration and Edge Device Communication

【110上】嵌入式系統技術實驗課程

TA: 陳翰群 [hanz1211.ee09@nycu.edu.tw](mailto:hanz1211.ee09@nycu.edu.tw)

# Before We Start



- Some packages we introduce later may already be installed in the system image TA provided, if not, don't forget to use **pip3** to install the package instead of pip
- Do not train DL model directly on Jetson Nano, the DL framework I installed is only for inference and optimize model
- To make your project stand out from other teams, there are some approaches:
  - Performance Optimization
    - Multi-thread/process programming
    - Include more experiment and performance comparison in your final report
  - Nice UI/UX
    - PyQt, Pygame, etc.
    - StyleSheet
  - Heterogeneous Platform
    - UART & GPIO
    - Socket & HTTP

- An SDK for high-performance deep learning inference
- There are 6 technique in TensorRT can optimize your model:
  1. Reduce Precision: quantizing models to FP16 or INT8
  2. Layer and Tensor Fusion: merge nodes to optimize GPU memory bandwidth
  3. Kernel Auto-Tuning: select best algorithms based on GPU platform
  4. Dynamic Tensor Memory: reuse memory for tensors efficiency
  5. Multi-Stream Execution: process multiple input streams in parallel
  6. Time Fusion: optimize RNN over time step
- Support TensorFlow, ONNX, MATLAB model out of the box
- PyTorch model requires more steps to convert
- You need to convert model directly on Jetson Nano

- All models in [Jetson Zoo](#) are optimized with TensorRT already
  - The sample code here are mostly from [Hello AI World \(Jetson Inference\)](#)
  - You may need to check API Reference to use this with your project
  - For example, [poseNet](#), from API you can learn how map keypoint ID to there meaning by GetKeypointName() method.

## API Reference

Below are links to reference documentation for the C++ and Python libraries from the repo:

### jetson-inference

	C++	Python
Image Recognition	<a href="#">imageNet</a>	<a href="#">imageNet</a>
Object Detection	<a href="#">detectNet</a>	<a href="#">detectNet</a>
Segmentation	<a href="#">segNet</a>	<a href="#">segNet</a>
Pose Estimation	<a href="#">poseNet</a>	<a href="#">poseNet</a>
Monocular Depth	<a href="#">depthNet</a>	<a href="#">depthNet</a>

```
class poseNet(tensorNet)
    Pose Estimation DNN - detects the poses of objects in an image

    Examples (jetson-inference/python/examples)
    posenet.py
    init(...)
    Loads an pose estimation model.

    Parameters:
        network (string) -- name of a built-in network to use
                           see below for available options.

        argv (strings) -- command line arguments passed to poseNet,
                           see below for available options.

        threshold (float) -- minimum detection threshold.
                           default value is 0.15
```

```
FindKeypointID(...)
    Return the keypoint ID for the given keypoint name.

    Parameters:
        (str) -- name of the keypoint

    Returns:
        (int) -- the ID of the keypoint

GetKeypointName(...)
    Return the keypoint name for the given keypoint ID.

    Parameters:
        (int) -- index of the keypoint, between [0, GetNumKeypoints\(\)]

    Returns:
        (string) -- the text description of the keypoint

GetKeypointScale(...)
    Get the scale used to calculate the radius of keypoints based on image d

    Parameters: (none)

    Returns:
        (float) -- the scale used to calculate the radius of keypoints based o

GetLinkScale(...)
    Get the scale used to calculate the width of link lines based on image d

    Parameters: (none)

    Returns:
        (float) -- the scale used to calculate the width of link lines based o

GetNumKeypoints(...)
    Return the number of keypoints in the model's pose topology.

    Parameters: (none)

    Returns:
        (int) -- number of keypoints in the model's pose topology
```

- ONNX is a universal ML **serialize model format**, only store and exchange the trained model, reduce the dependency during deployment.
- PyTorch, MXNet, Caffe2, TensorRT officially support export to ONNX
  - PyTorch has a simpler way to use TensorRT, you can jump to “**torch2trt**” page
- For example, in PyTorch, `torch.onnx.export()` can export model with given input size
  - When exporting your model as ONNX, if your original model can accept dynamic input, the **converted model input will be fixed** by the dummy input size you provided.
- Examples:
  - This PyTorch official [example](#) convert AlexNet to ONNX for further deployment
  - And this [advanced guide](#) show how to export model ONNX and run with ONNX Runtime (I do not recommend using ONNX Runtime on Nano, this is just for reference)
  - This [tutorials](#) also shows you can use Neutron to visualize the ONNX model

# ONNX to TensorRT

- Original TensorRT is using ONNX as default
- You can easily convert ONNX to TRT with official Jetson Nano tool: trtexec
  - `/usr/src/tensorrt/bin/trtexec`
- TensorRT runtime requires 4 steps:
  1. ONNX Parser
  2. Builder: Build TensorRT engine
    - You need to specify input size, workspace size (VRAM during inference), and precision
  3. Engine: Provide inference
    - Image should be Numpy format and RGB channel
  4. Logger: Logging inference information
- TL;DR
  - [Jetson Nano 運用TensorRT加速引擎 – 上篇](#)
  - [Jetson Nano 運用TensorRT加速引擎 – 下篇](#)
    - I did not install onnx and onnxruntime package for you since you can skip that section

# PyTorch to TensorRT: torch2trt



- [torch2trt](#) Github
  - The REAME.md shows some simple usage, for more detail check their [example](#)
- This is the official tool convert PyTorch nn.Module to TensorRT format
  - Behind the scene is still passing by ONNX format
- Has a high-level runtime class: **TRTModule**
  - No need to write TensorRT engine or builder yourself
- TL;DR
  - [Jetson Nano 運用TensorRT加速引擎 – 下篇](#)
    - There is a section using torch2trt you can reference
- You may want to dive into **source code** of [torch2trt class](#) for more configuration argument including INT8 mode
  - Some arguments you shall be aware are max\_workspace\_size, int8\_calib\_dataset



# TensorFlow to TensorRT



- TF-TRT: TensorFlow has a different way to convert to TensorRT, not using ONNX
- The model needs to be in [SavedModel format](#)
- You can follow this official [quickstart](#) guide, learn how to convert ResNet50 provided in Keras
  - On Jetson Nano, no **nvidia-smi** command available, use **jtop** to check GPU status
  - You should not use a large BATCH\_SIZE, usually during inference we only need **batch size of 1**
  - In the section “Reducing precision to INT8”, you need to provide a **calibration batch**(calibration are only required for INT8, not for FP16). The example use 8 dummy zero matrixes, the resulting model won't be correct, you need to use real images.
- Additional Examples
  - [Classification](#)
  - [Detection](#)
  - [Segmentation](#)



# YOLO to TensorRT



- Directly convert darknet model to TensorRT may delay your progress
- If you want to use YOLO v2/3/4 for object detection, use the same approach in the Jetson Zoo samples, which refer to this [GitHub repo](#)
  - Include detailed steps to convert ONNX format YOLOv3 & YOLOv4 to TensorRT and test on Jetson Nano
  - Also cover the step to convert GoogLeNet for classification, SSD for object detection, MTCNN for face detection
  - The section “Demo #6: Using INT8 and DLA core” is not viable for Jetson Nano

# Topic: Transfer Learning

- This PyTorch [tutorial](#) show how to use pretrained [torchvision model](#) and finetune it for your dataset, basically these steps:
  1. Load model from torchvision
  2. Drop the last FC layer (nn.Linear)
  3. [Freeze the model parameter](#) if you feel like it
  4. [Reshape network](#) based on your number of classes
    - Tutorial reshape output class from 1000 to 500
    - The in\_features size must match
- This TensorFlow [tutorial](#) use the same approach to achieve transfer learning too
  - The tutorial also introduce **data augmentation**, which is a powerful technique if your dataset is not large enough

# Topic: Pose Estimation



- The Jetson Zoo provides PoseNet with ResNet18 & DenseNet121 backbone
- If you want to know the details of converting these model or modify for your use, check [trt\\_pose](#) repo
  - This project use torch2trt
- Hand Gesture Recognition
  - The [trt\\_pose\\_hand](#) repo provides gesture recognition based on trt\_post

# Jetson Nano Forum



- <https://forums.developer.nvidia.com/c/agx-autonomous-machines/jetson-embedded-systems/jetson-nano/76>
- This is the official forum where you may find more resources
- If you felling the system image from TA is too bulky, you can use the official image, but remember, if you PyTorch or TensorFlow, you need to get it here

🚩 Debug tips when you hit any issue (fail to boot/ keep reboot ...etc)	  	2	1.8k
🔒 🚩 Jetson Nano FAQ		0	15.7k
🚩 PyTorch for Jetson - version 1.10 now available <span>965</span>	    	933	212k
🚩 Official TensorFlow for Jetson Nano!	    	225	84.3k
🚩 Links to Jetson Nano Resources & Wiki	    	74	25.6k
🚩 Power supply considerations for Jetson Nano Developer Kit	    	284	133k