神经网络部署

8.1 程序语言运行时

Program Runtime

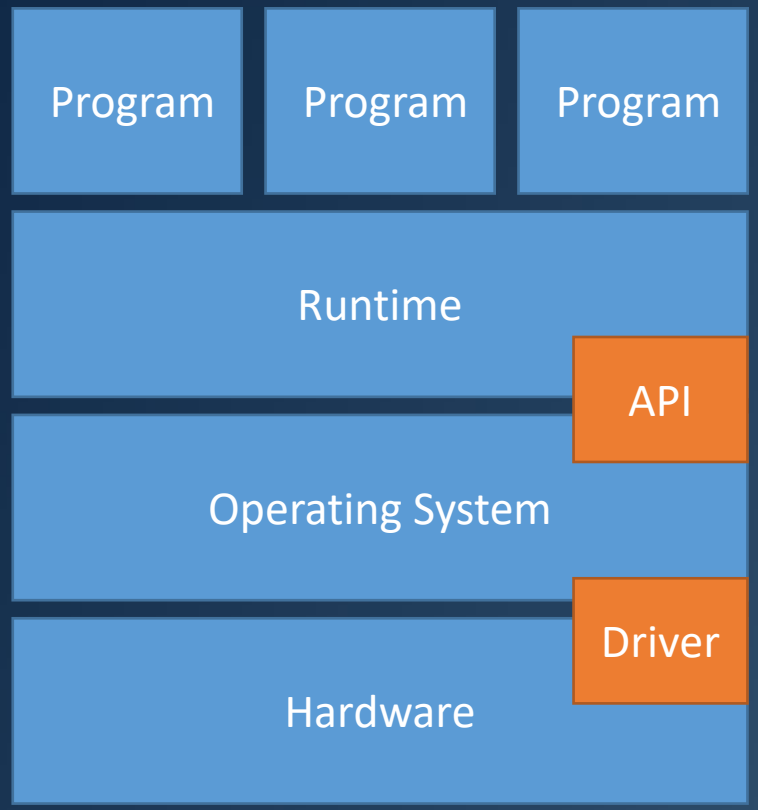
在很多程序语言中，你已经接触过运行时的概念：

- Runtime is a piece of code that implements portions of a programming language's execution model. In doing this, it allows the program to interact with the computing resources it needs to work. Runtimes are often integral parts of the programming language and don't need to be installed separately.

- 运行时是程序语言提供一个代码库，你的程序将调用运行时中的功能来访问系统资源，最典型的运行时是java虚拟机，c++ runtime，c# runtime等等。

8.1 程序语言运行时

Runtime



- 很容易理解的是，Runtime作为程序运行的基础，它决定了你的程序访问系统资源的方式。你写的程序最终总是要调用Runtime中相关的函数去执行，它会将你的程序翻译成操作系统调用，或更加底层的硬件汇编指令。
- 虽然我们在这里使用了“翻译”这个词，但不代表runtime是解释型语言特有的，事实上VC++ Runtime是目前最广泛使用的runtime库之一。

8.1.1 C++ 运行时与 C++ 标准

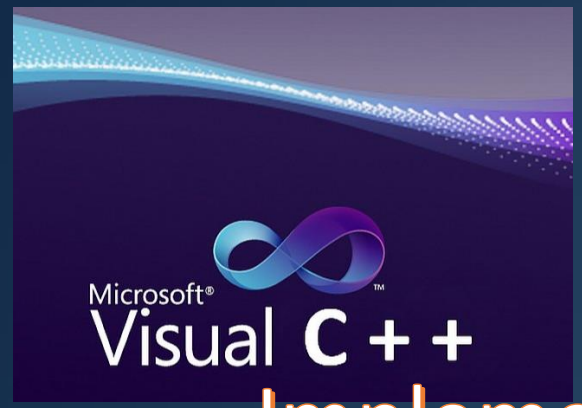
CRT & C++ Standard

- C++ 标准描述了 C++语言 所需要支持的基本数据类型、标准库所需包含的基本算法、对于内存和进程的访问控制方式等等 …
- 但简单来说，有了标准不等于可以写出程序，人们需要在 C++ 标准的基础上编写自己的编译器与运行时，甚至是集成开发环境。

C++ 语言必须包括 new，delete，int，float，double，+ - */，if，else，for，while，switch，还有STL库。

8.1.1 C++ 运行时与 C++ 标准

CRT & C++ Standard



VC Runtime

MSVC Compiler

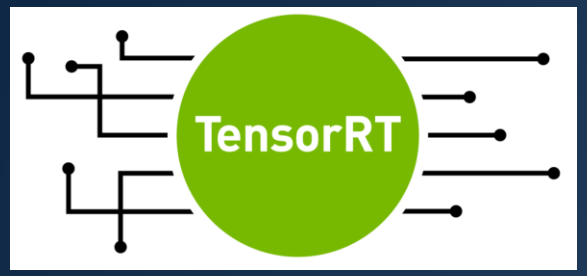
STL Implementation

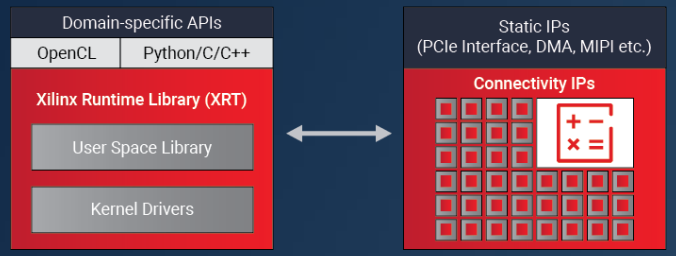
Implementation

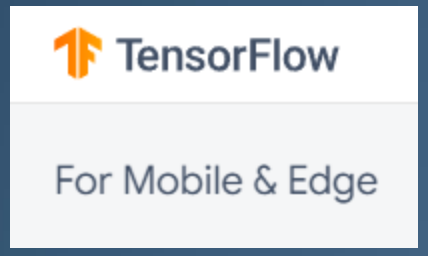
Standard

8.1.2 神经网络运行时

Nerual Network Runtime







8.1.2 神经网络运行时

Nerual Network Runtime

- 硬件厂商提供的：TensorRT，OpenVINO，VitisAI，SNPE，Tengine
- 软件厂商提供的：Onnxruntime，Tengine，Paddlelite，TFLite，TorchScript，MNN，ncnn，TNN，OpenPPL

大家静一静，我跟大家谈谈
神经网络标准的问题

8.1.2 神经网络运行时

Nerual Network Runtime

- 硬件厂商提供的：TensorRT，OpenVINO，VitisAI，SNPE，Tengine
- 软件厂商提供的：Onnxruntime，Tengine，Paddlelite，TFLite，TorchScript，MNN，ncnn，TNN，OpenPPL

8.1.2 神经网络运行时

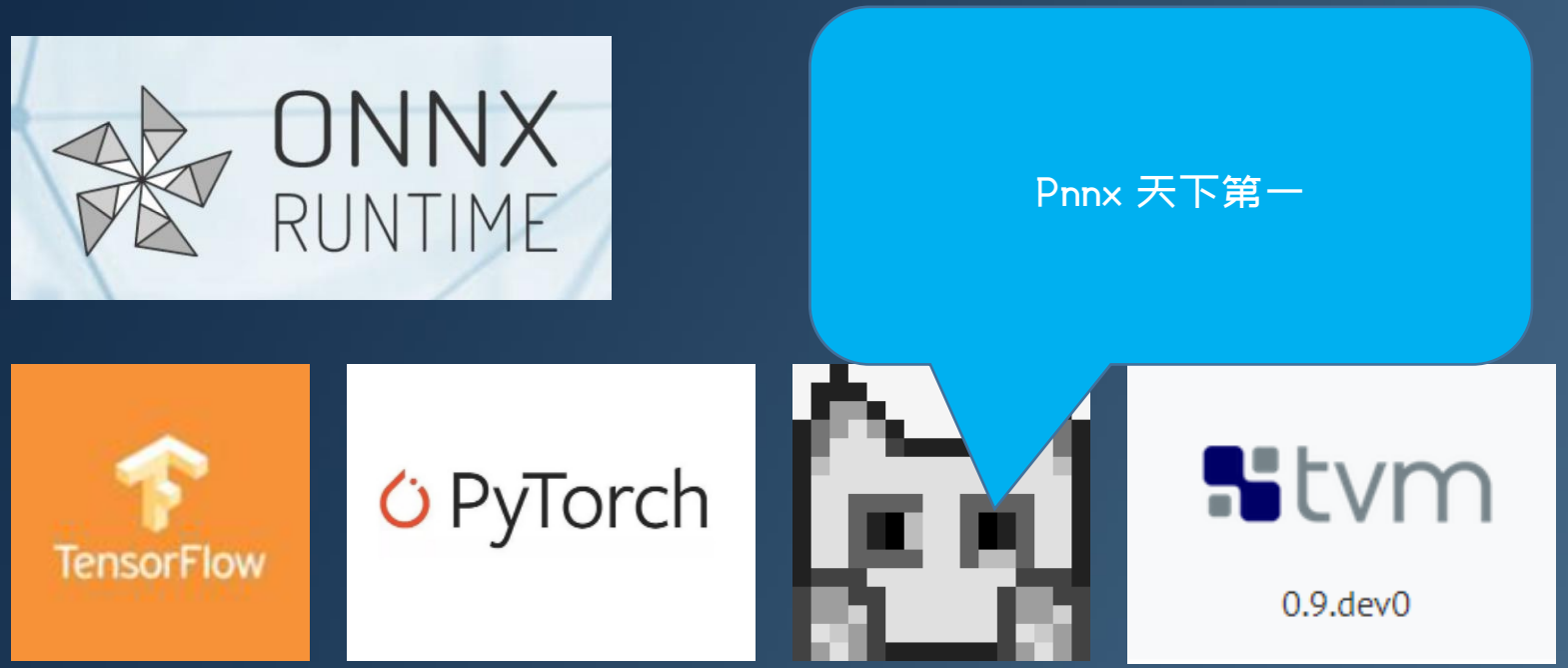
Nerual Network Runtime

- 硬件厂商提供的：TensorRT，OpenVINO，VitisAI，SNPE，Tengine
- 软件厂商提供的：Onnxruntime，Tengine，Paddlelite，TFLite，TorchScript，MNN，ncnn，TNN，OpenPPL

8.1.2 神经网络运行时

Nerual Network Runtime

- 硬件厂商提供的：TensorRT，OpenVINO，VitisAI，SNPE，Tengine
- 软件厂商提供的：Onnxruntime，Tengine，Paddlelite，TFLite，TorchScript，MNN，ncnn，TNN，OpenPPL



8.1.2 神经网络运行时

Nerual Network Runtime

- 硬件厂商提供的：TensorRT，OpenVINO，VitisAI，SNPE，Tengine
- 软件厂商提供的：Onnxruntime，Tengine，Paddlelite，TFLite，TorchScript，MNN，ncnn，TNN，OpenPPL

8.1.2 神经网络表示

Nerual Network Representation

- 截至2022年为止，神经网络的表示问题仍然没有得到完善解决，其中的原因包括：
  - 算法工程师总是喜欢在现有研究基础上进行排列组合，一顿操作猛如虎，一看涨点0.5%（直接原因）。
  - 神经网络领域的快速发展，技术的快速迭代使得原有表示不再适用。
  - 制定标准意味着利益，各大厂商为此蠢蠢欲动（根本原因）。

Oh, it Sucks!

8.1.2 神经网络表示

Nerual Network Representation

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

PyTorch

```
scores = torch.matmul(query, key.transpose(-2, -1)), math.sqrt(query.size(-1))

if mask is not None:
    scores = scores.masked_fill(mask == 0, -1e9)

p_attn = F.softmax(scores, dim=-1)

if dropout is not None:
    p_attn = dropout(p_attn)
```

8.1.2 神经网络表示

Nerual Network Representation

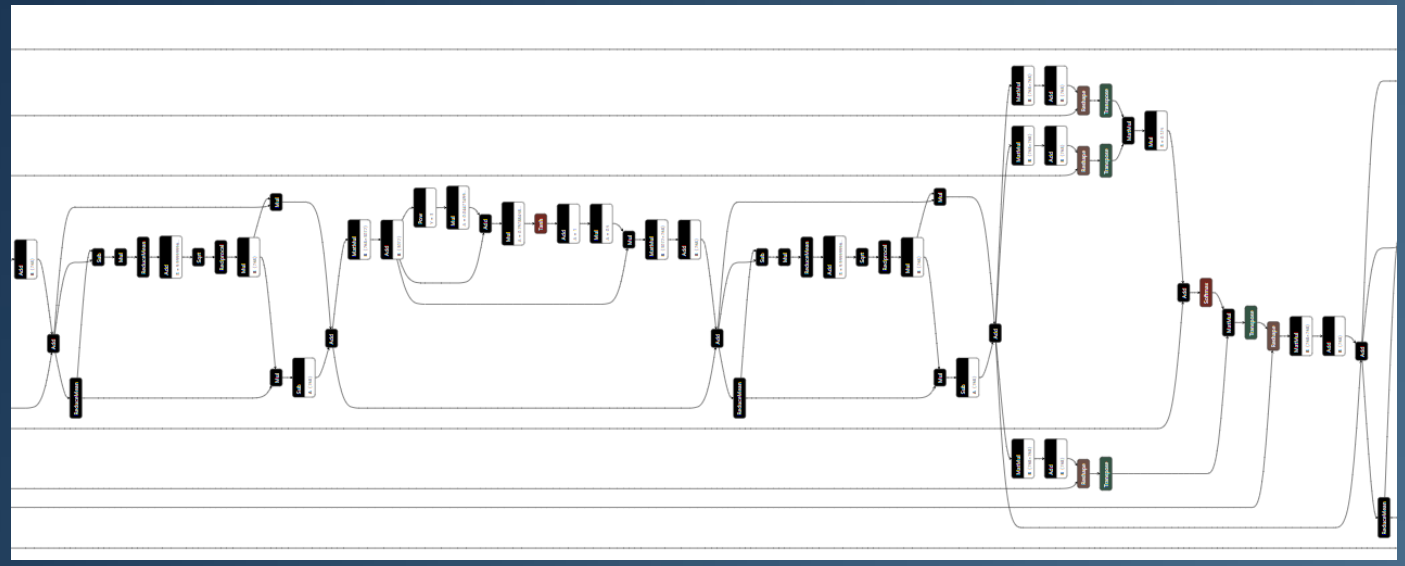
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$


```
attention_scores = tf.matmul(query_layer, key_layer, transpose_b=True)
 attention_scores = tf.multiply(attention_scores,
                                1.0 / math.sqrt(float(size_per_head)))

 if attention_mask is not None:
  attention_mask = tf.expand_dims(attention_mask, axis=[1])
  adder = (1.0 - tf.cast(attention_mask, tf.float32)) * -10000.0
  attention_scores += adder

 attention_probs = tf.nn.softmax(attention_scores)
 attention_probs = dropout(attention_probs, attention_probs_dropout_prob)
```

8.1.2 神经网络表示

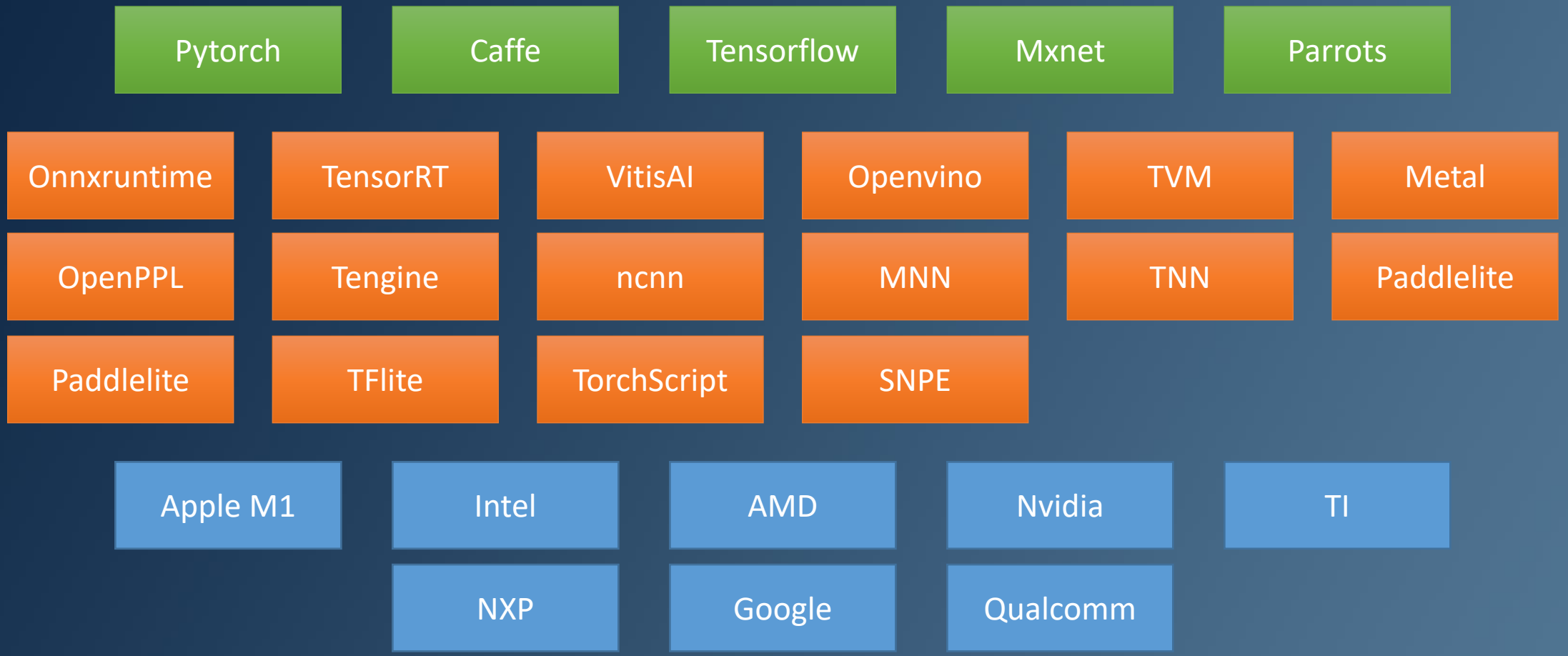
Nerual Network Representation

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



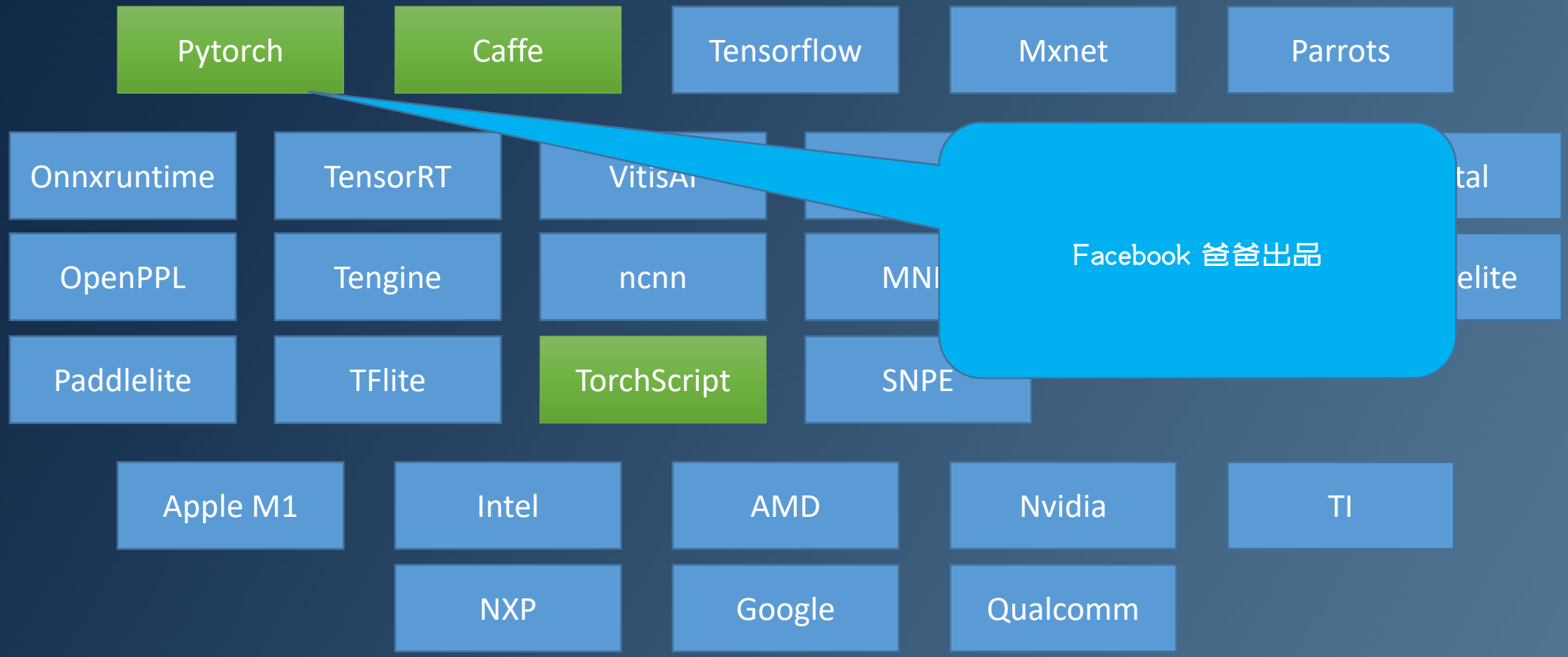
8.1.3 神经网络部署

Nerual Network Depoly



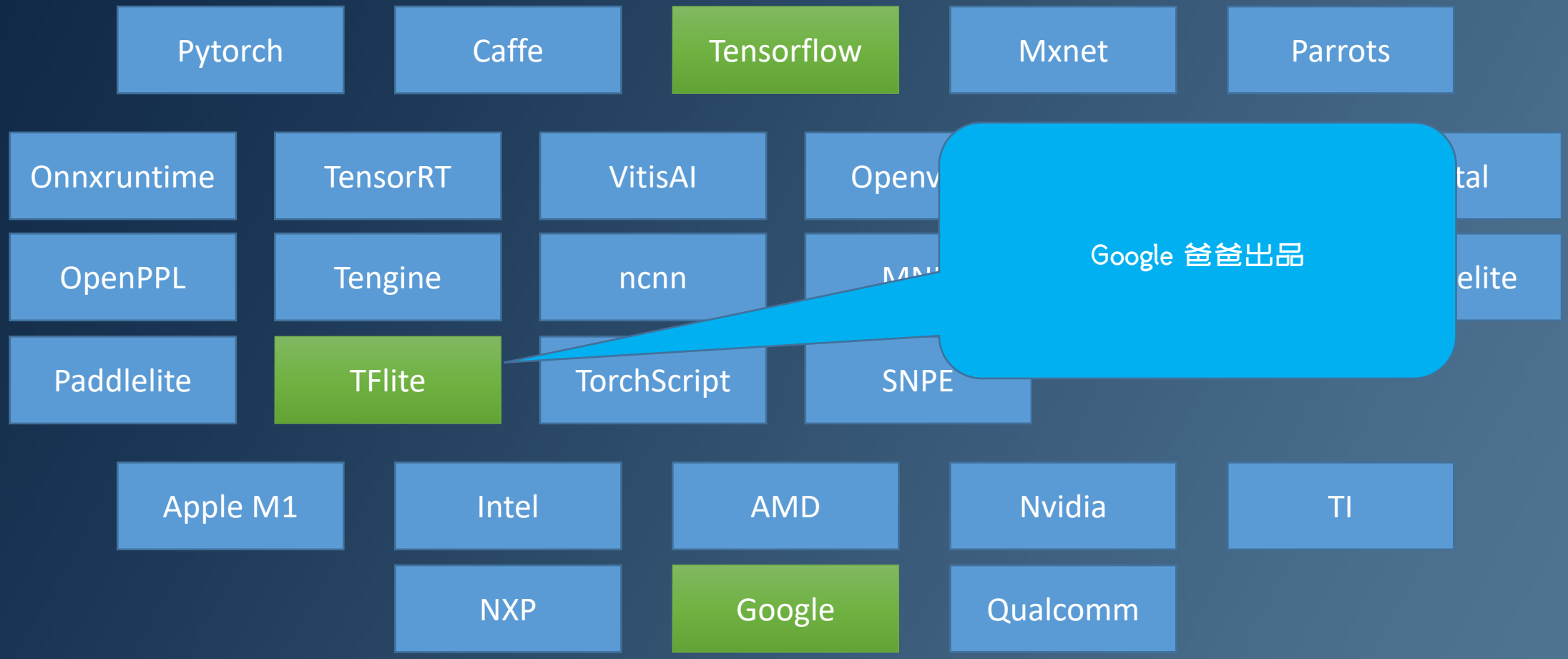
8.1.3 神经网络部署

Nerual Network Depoly



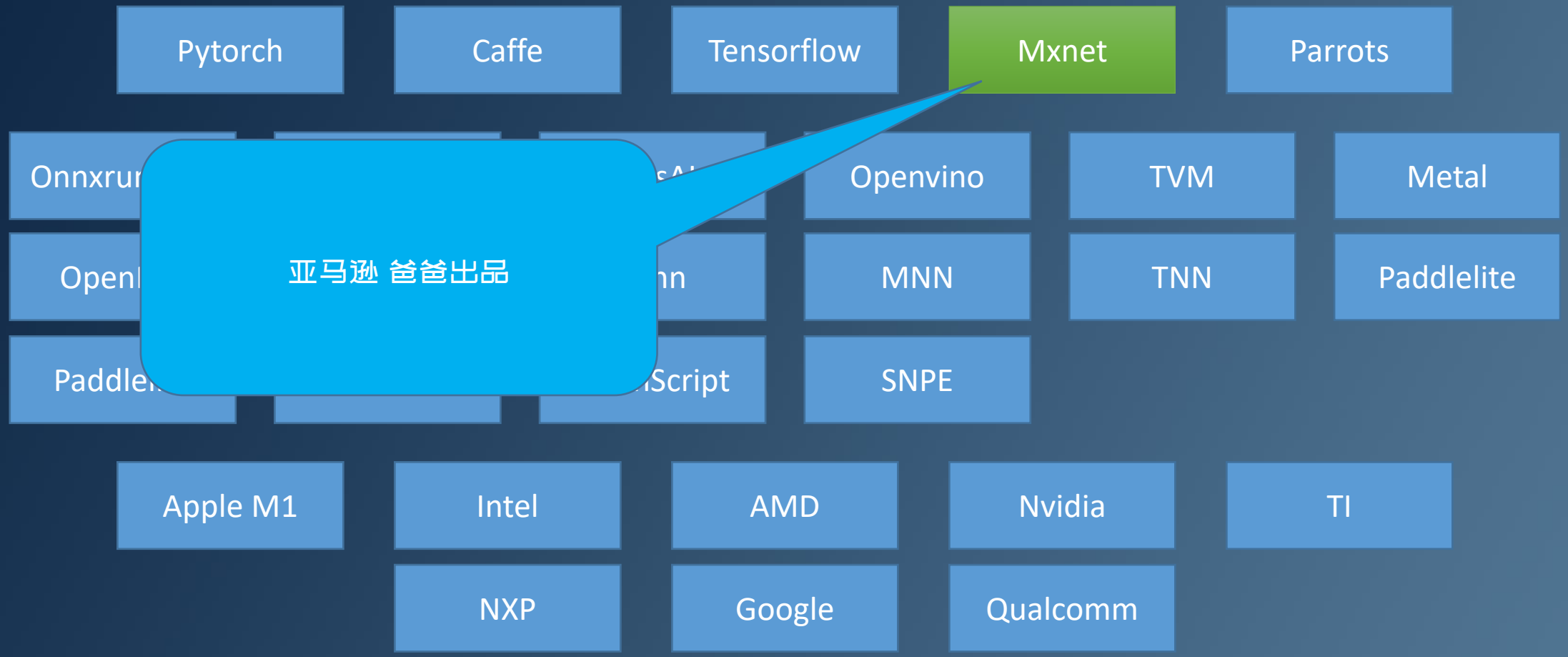
8.1.3 神经网络部署

Nerual Network Depoly



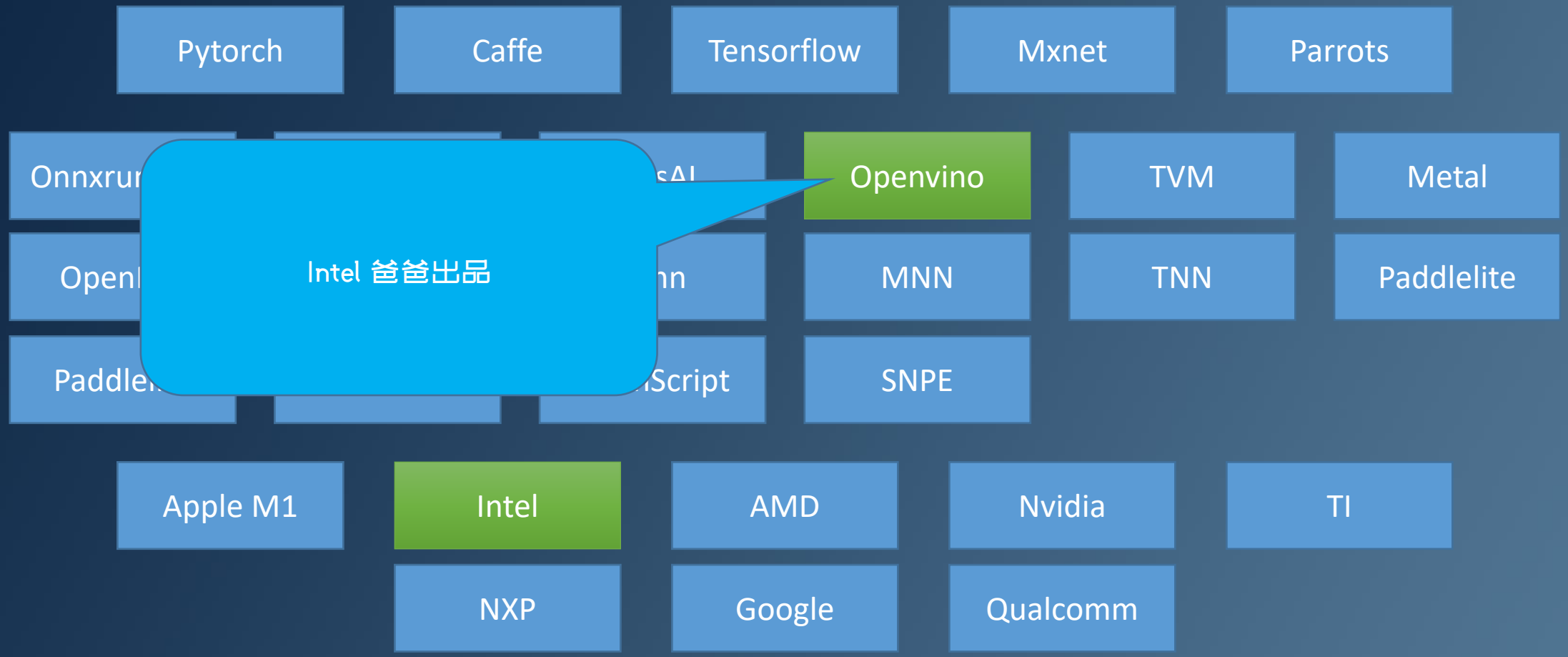
8.1.3 神经网络部署

Nerual Network Depoly



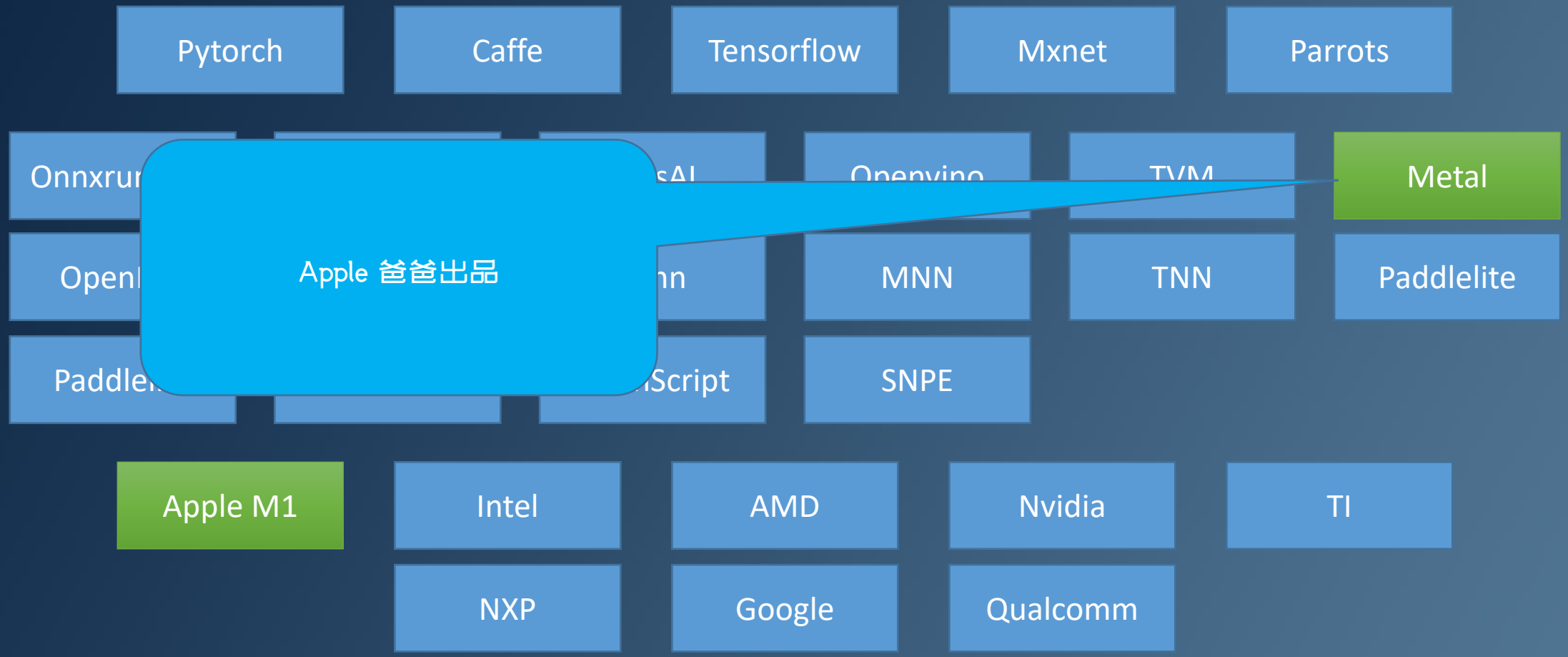
8.1.3 神经网络部署

Nerual Network Depoly



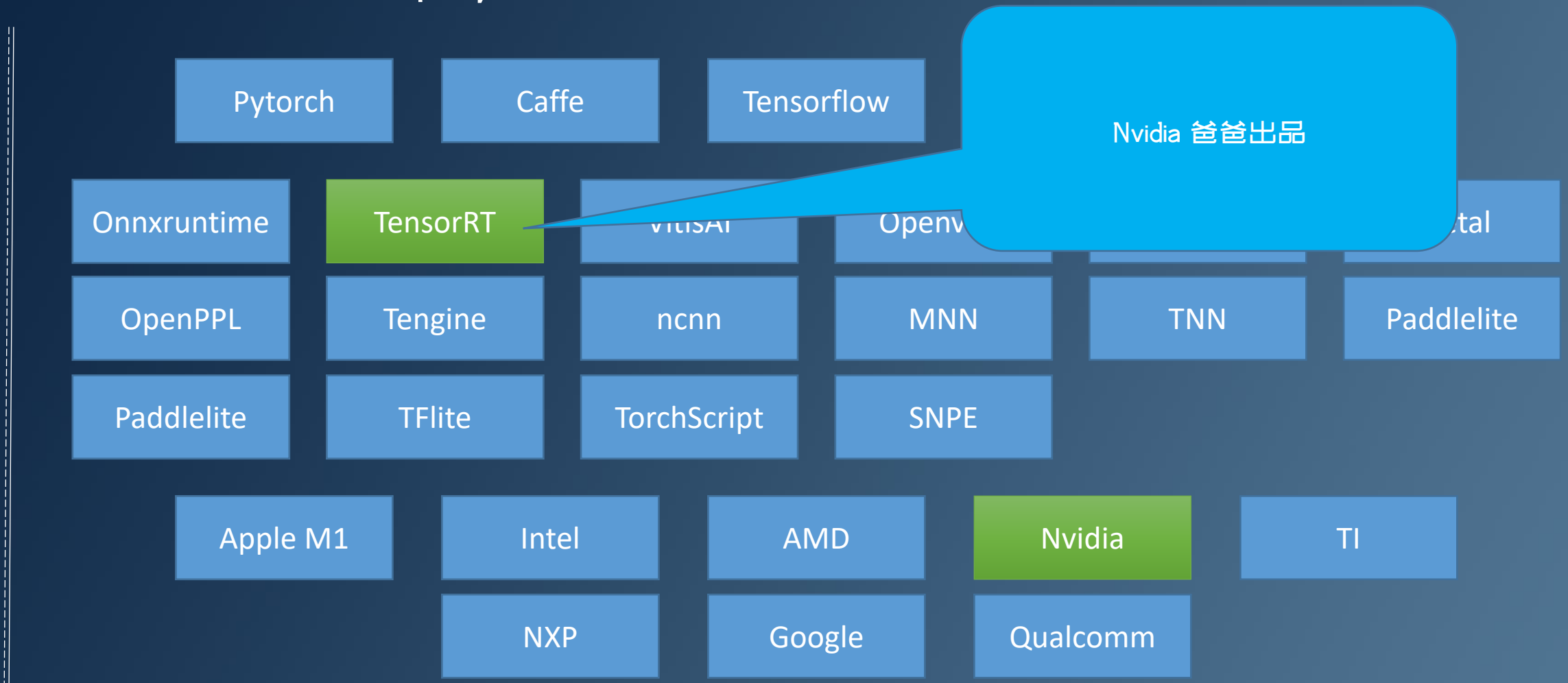
8.1.3 神经网络部署

Nerual Network Depoly



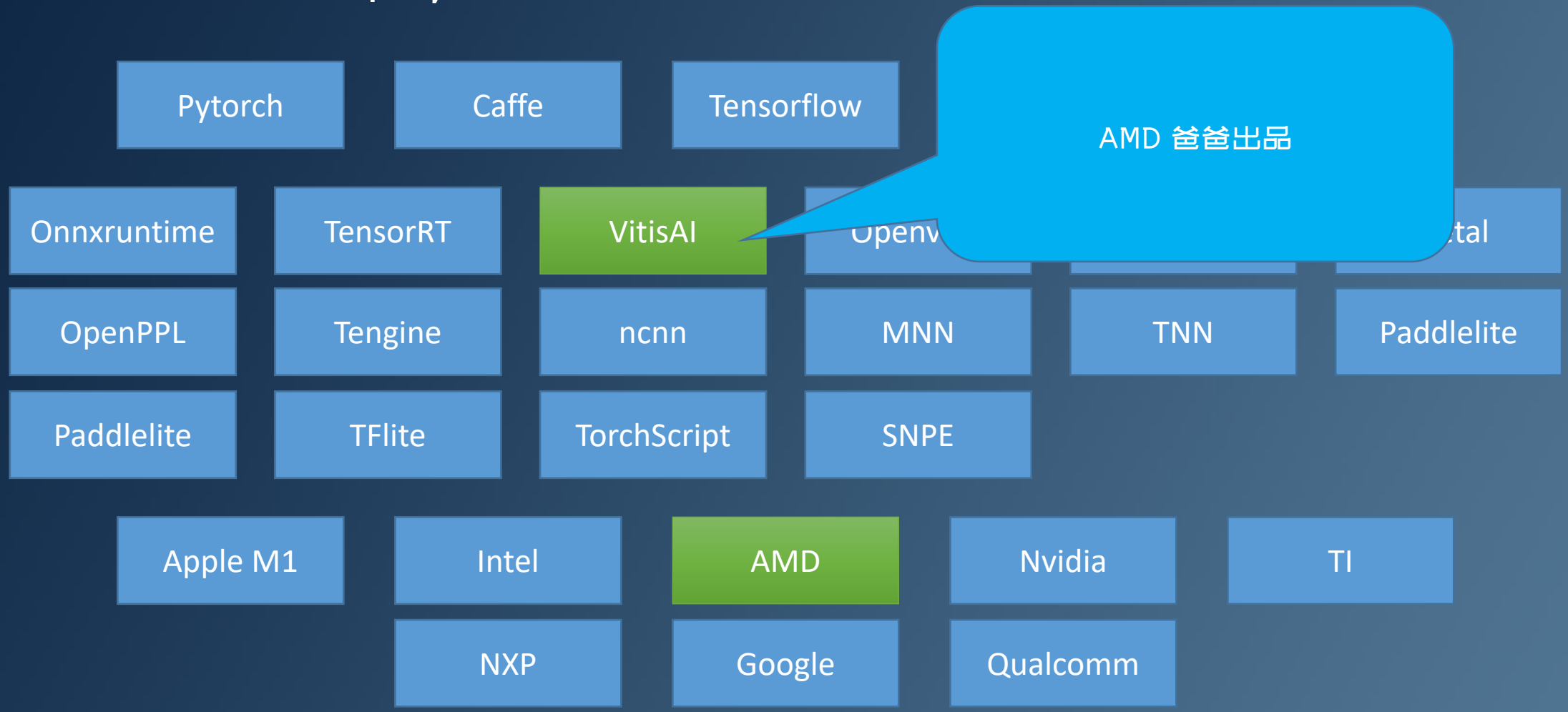
8.1.3 神经网络部署

Nerual Network Depoly



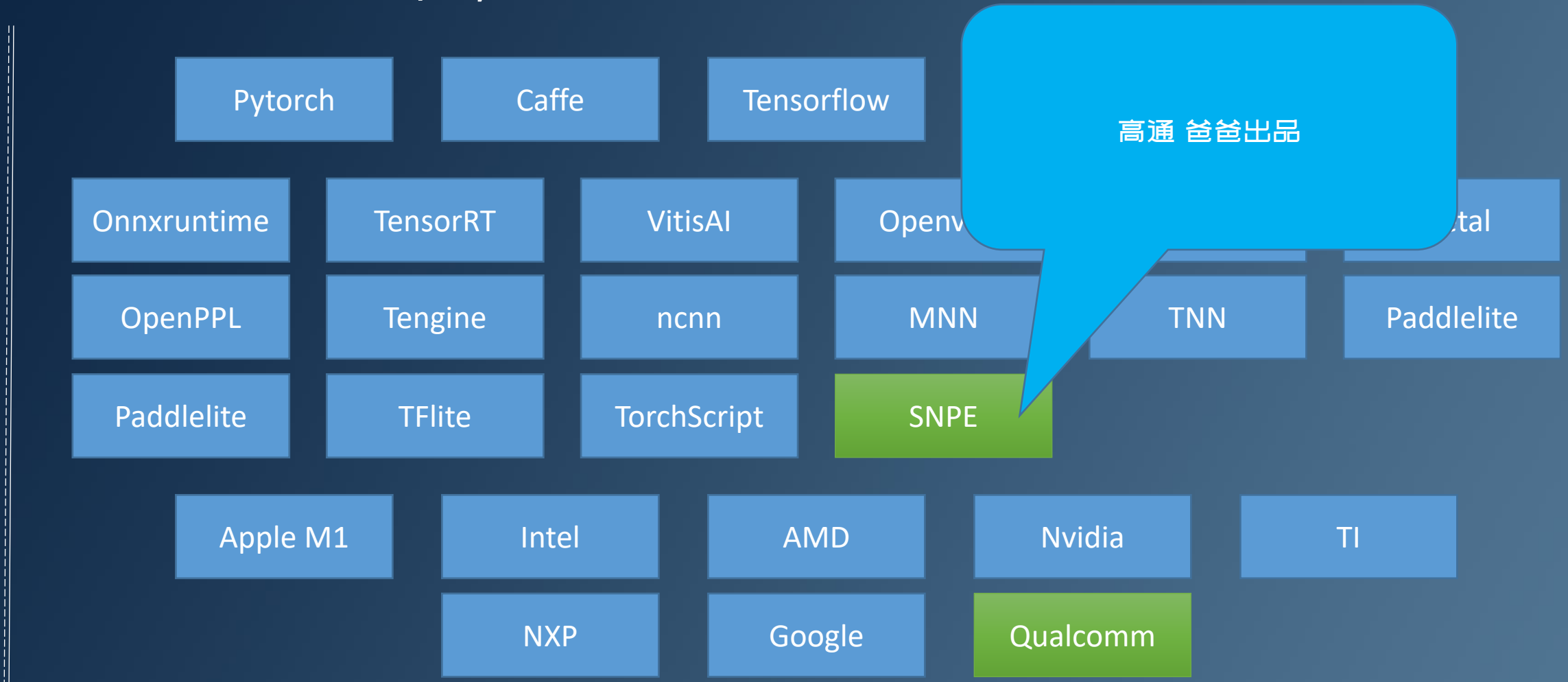
8.1.3 神经网络部署

Nerual Network Depoly



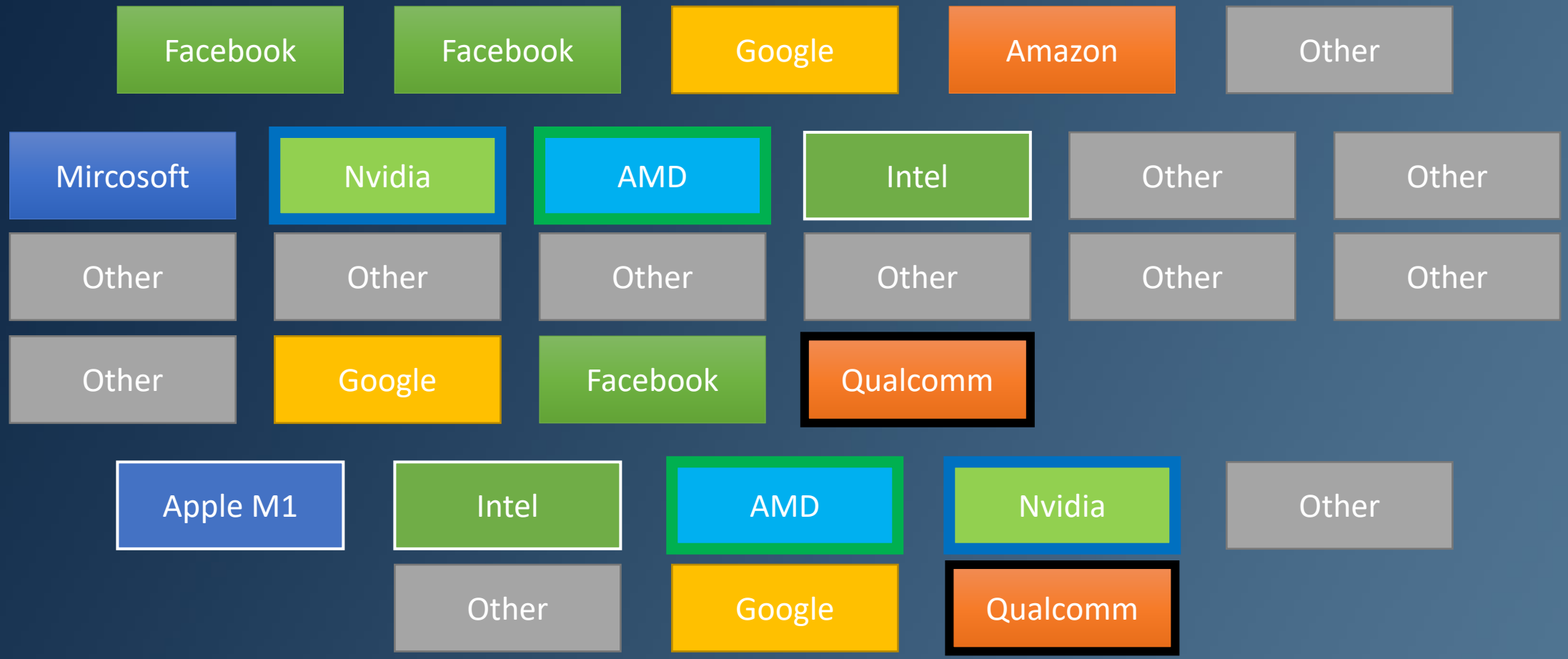
8.1.3 神经网络部署

Nerual Network Depoly



8.1.3 神经网络部署

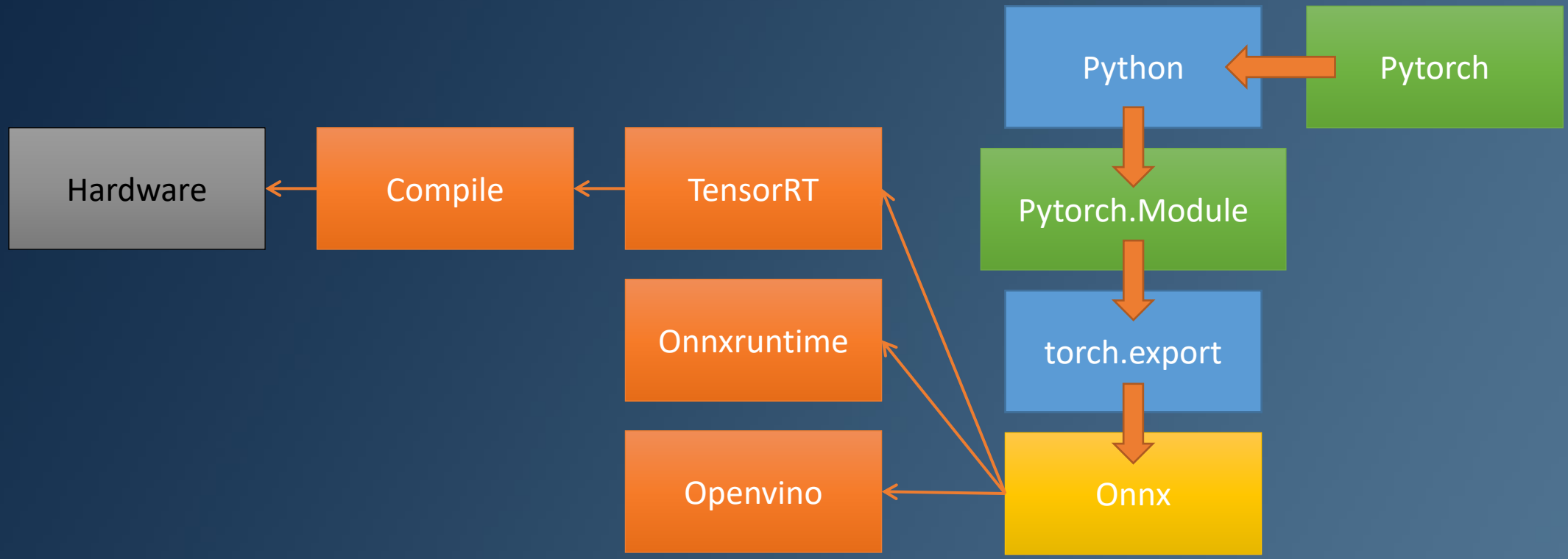
Nerual Network Depoly



小知识：这几家公司总市值约等于我国GDP的三分之一、约等于20个腾讯、40个阿里巴巴

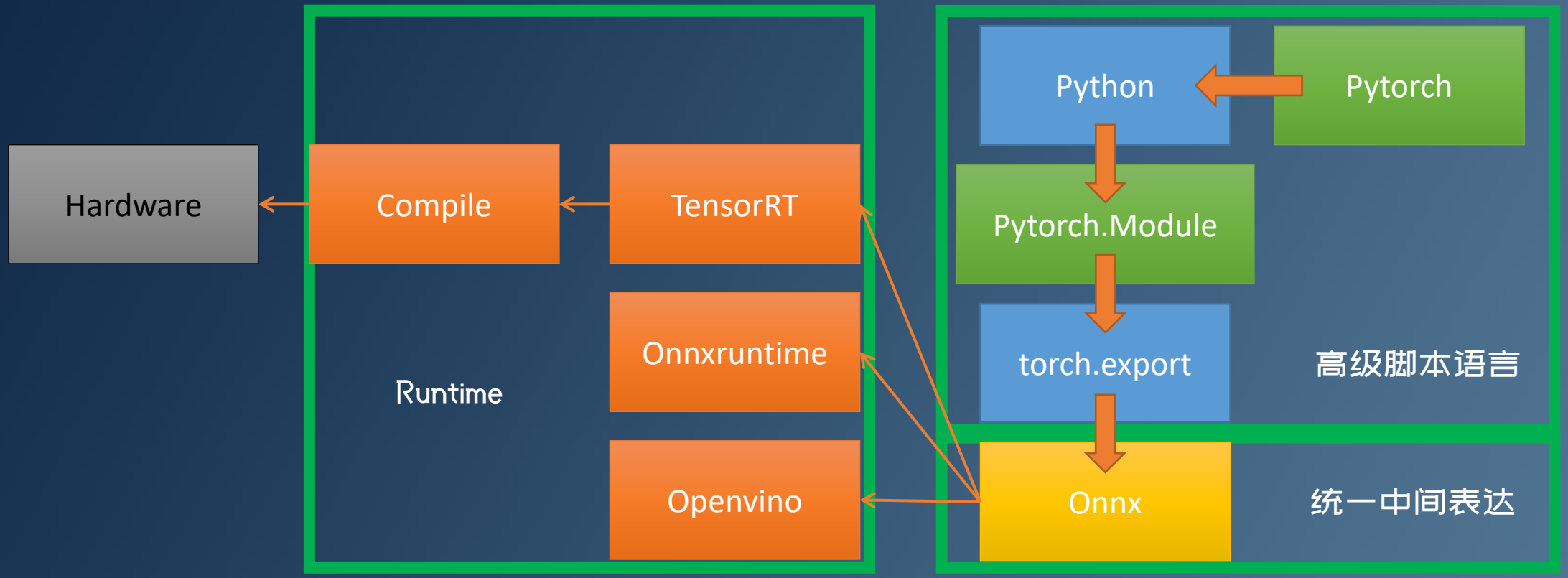
8.1.3 神经网络部署

Nerual Network Depoly



8.1.3 神经网络部署

Nerual Network Depoly



8.1.3 神经网络部署

Nerual Network Depoly

- 送给你的建议：
  - 确保你的网络可以被Onnx表示，避免其中出现复杂条件逻辑及循环逻辑。
  - 学会自定义算子，以备不时之需，（包括自定义算子的推理实现）。
  - 避免使用各种小Trick，额外加入的算子很可能会破坏图优化。
  - 神经网络能跑多快是Runtime决定的，神经网络加速应当根据runtime进行。
  - 用一下 Onnx Simplifier。
  - 写一个固定的 batchsize。

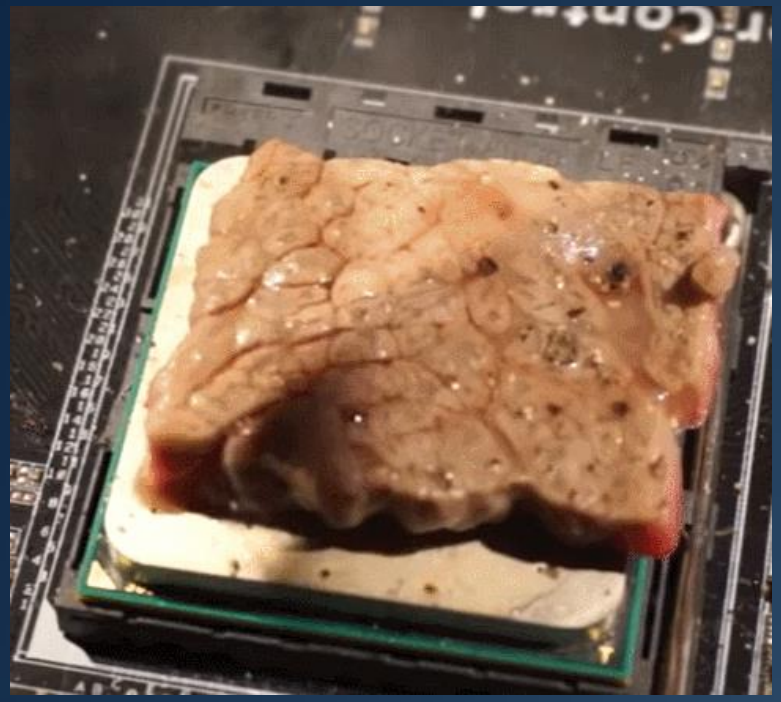
8.1.3 神经网络部署

Nerual Network Depoly

- 送给你的建议：
  - 确保你的网络可以被Onnx表示，避免其中出现复杂条件逻辑及循环逻辑。
  - 学会自定义算子，以备不时之需，（包括自定义算子的推理实现）。
  - 避免使用各种小Trick，额外加入的算子很可能会破坏图优化。
  - 神经网络能跑多快是Runtime决定的，神经网络加速应当根据runtime进行。
  - 用一下 Onnx Simplifier。
  - 写一个固定的 batchsize。

接下来让我们使用第一个神经网络推理框架：Onnxruntime

联系我们 https://github.com/openppl-public



广告位招租

微信群

QQ群（入群密令OpenPPL）