1. 实现方式：符号式编程vs命令式编程

tensorflow是纯符号式编程，而pytorch是命令式编程。

命令式编程优点是实现方便，缺点是运行效率低。

符号式编程通常是在计算流程完全定义好后才被执行，因此效率更高，但缺点是实现复杂。

2. 图的定义：动态定义vs静态定义

两个框架都是在张量上进行运算，但是却存在着很大的差别。

TensorFlow遵循"数据即代码，代码即数据"的理念，可以在运行之前静态的定义图，然后调用session来执行图。

pytorch中图的定义是动态化的，可以随时定义、随时更改、随时执行节点。

因此相对而言，pytorch更加灵活，更加方便调试。

3. 可视化：tensorboard vs nothing

我认为TensorFlow最吸引人的地方之一就是tensorboard，可以清晰的看出计算图、网络架构，而pytorch自己没有类似tensorboard的工具，但是pytorch可以导入tensorboardx或者matplotlib这类工具包用于数据可视化。

以上对比基于老版本TF和pytorch，在TF2.0、pytorch1.0之后都做了大幅的升级和改进，如有不同之处，请以自己使用的版本为主。

F强调工业布署，PyTorch强调简法易用

动态图 vs. 静态图

在 fast.ai，我们在选择框架时优先考虑程序员编程的便捷性（能更方便地进行调试和更直观地设计），而不是框架所能带来的模型加速能力。这也正是我们选择 PyTorch 的理由，因为它是一个**具有动态图机制的灵活框架。**

依据采用动态计算或是静态计算的不同，可以将这些众多的深度学习框架划分成两大阵营，当然也有些框架同时具有动态计算和静态计算两种机制（比如 MxNet 和最新的 TensorFlow）。动态计算意味着程序将按照我们编写命令的顺序进行执行。这种机制将使得调试更加容易，并

且也使得我们将大脑中的想法转化为实际代码变得更加容易。而静态计算则意味着程序在编译执行时将先生成神经网络的结构，然后再执行相应操作。从理论上讲，静态计算这样的机制允许编译器进行更大程度的优化，但是这也意味着你所期望的程序与编译器实际执行之间存在着更多的代沟。这也意味着，代码中的错误将更加难以发现（比如，如果计算图的结构出现问题，你可能只有在代码执行到相应操作的时候才能发现它）。**尽管理论上而言，静态计算图比动态计算图具有更好的性能，但是在实践中我们经常发现并不是这样的。**

谷歌的 TensorFlow 主要使用了静态计算图，而 Facebook 的 PyTorch 则使用了动态计算图机制。（注：TensorFlow 在两周前宣布了一个动态计算选项 Eager Execution（http://t.cn/RIZizQ2），不过该特性还比较新颖并且 TensorFlow 的文档和项目依然以静态计算为主）。在九月份，fast.ai 宣布将在今年的课程中采用 PyTorch 框架进行教学以及开发 fast.ai 自己的框架（实际上就是采用了更好的编码方式对 PyTorch 进行高级封装）。简而言之，以下是我们选择 PyTorch 的几个原因（更详细的原因请参见这里 http://t.cn/Rpqj6pu）：

- 更加容易调试
- 动态计算更适用于自然语言处理
- 传统的面向对象编程风格（这对我们来说更加自然）
- TensorFlow 中采用的诸如 scope 和 sessions 等不寻常的机制容易使人感到疑惑不解，而且需要花费更多时间学习

谷歌在推广 TensorFlow 上已经花费了大量的资源，其投入要远远大于任何其它公司或者团队，并且我想这也是为什么 TensorFlow 会如此出名的原因之一（对于很多深度学习的门外汉，TensorFlow 是他们唯一听说过的框架）。正如之前所述，TensorFlow 在几周前发布了动态计算选项，这将解决了一些上述提到的问题。然后许多人就向 fast.ai 提问说我们是否考虑迁移回 TensorFlow 框架。**但是目前 TensorFlow 提供的动态选项还比较新颖而且开发也不够完善，所以我们依然选择继续愉快地使用 PyTorch**。但是 TensorFlow 团队非

常乐意于接受我们的想法，我们也很高兴看到我们的 fastai 库（http://t.cn/RYyK6jC）被移植到 TensorFlow 中。