Origins of PyTorch

- Interview with Soumith Chintala.
- Primarily developed by Facebook's AI Research lab (FAIR).
- Free and open-source software.

Debugging and Designing PyTorch

- PyTorch should be imperative, usable and Pythonic but at the same time as fast as other frameworks.
- Large parts of PyTorch are written in C++, but user end is Pythonic.

From Research to Production

- Researcher-friendly, doing loops of updates by taking feedback from researchers.
- Production looks only at the final goal. PyTorch devs wanted it to be user-friendly.
- API looks very similar to numpy and other data science packages.
- **Main Plus Point: Designed with users and their feedback in mind.**
- To make the code production-ready, just add function annotations (one-liners on top of the function). Then, PyTorch will parse this code into its own internal format. This model can then be treated as a black box. To experiment with it again, just comment out these function annotations.
- If a user wants to run the same code on a 8-bit machine rather than a 32-bit machine, can translate code to an intermediate representation (IR) which can run on C++.

Hybrid Frontend

- Parts of the model can be compiled (e.g. LSTM cells which never change much), and parts of them model can be experimented on (the arrangement of these cells).
- JIT compiler: exporting the trained model, making training phase faster and more optimized.
- ONNX standard: before PyTorch 1.0, this allowed deep learning frameworks like Caffe 2, TensorFlow, PyTorch to talk to each other. Allows us to take a model trained in one to be exported to another framework.
- But models that were more complicated were not ONNX exportable, so PyTorch was upgraded to allow itself to be made shipment-ready.

Cutting-edge Applications in PyTorch

- SMASH: One-Shot Model Architecture Search through HyperNetworks
  Andrew Brock, Theodore Lim, J.M. Ritchie, Nick Weston:
  https://arxiv.org/abs/1708.05344
- Hierarchical Neural Story Generation, Angela Fan, Mike Lewis, Yann Dauphin
  https://arxiv.org/abs/1805.04833

User Needs and Adding Features

a) Researchers:

- Want to be able to add features but no drop in performance.
- E.g. LSTM written with GPU library cuDNN, which is really faster than writing your own LSTM with loops.

- Problem: While trying a new paper idea, this architecture is limiting. E.g. trying out recurrent dropout. Handwriting the loops shows a 10x slowdown.
- Can be solved by stitching high-performance GPU kernels on the fly in the backend.

b) Startups and Online Courses

- Want more interactive tutorials based on iPython Notebooks.
- Integration with Colab, free GPU.
- Support for Google TPUs.

PyTorch and the Facebook Product

- Facebook AI Research.
- Requirements at FB like camera enhancements, accessibility interfaces, integrity filtering, machine translation. Tools like PyTorch supplement this research.
- Good to make it an Open Source tool - advance humanity through AI.

The Future of PyTorch

- DL is becoming a necessary component in many new fields: healthcare+DS, chemistry, particle physics, neurobiology, neuroscience.
- They may use it in a rudimentary sense: using DL repos as black boxes.
- Go into these fields, understand what they need and build packages that lower their DL burdens and that they can relate to.

Learning More in AI

- Some advice!
- Be hands-on from day 1.
- Passive stuff: listening to tutorials, reading blogs not enough, because being able to do things is important.
- Do projects.