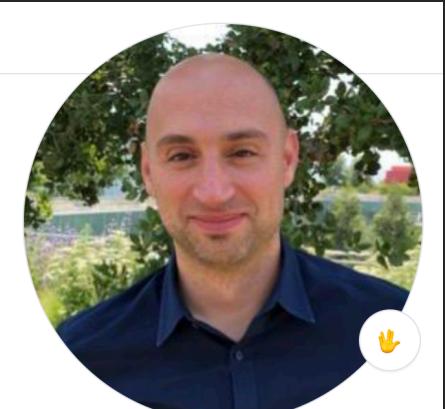
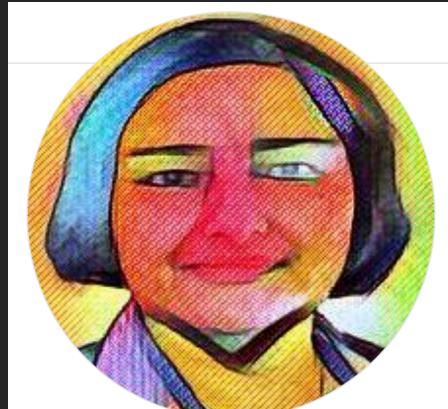




## AGENDA



**Joseph Spisak**  
jspisak



**Geeta Chauhan**  
chauhang

01

### WHAT IS PYTORCH?

02

### RECOMMENDER SYSTEMS USING DLRM

03

### INTERPRETABILITY & RECOMMENDER SYSTEMS

04

### USING MMF FOR KNOWLEDGE BASED RECOMMENDER SYSTEMS

05

### CLOSING THOUGHTS & MATERIALS



## PYTORCH MISSION

RESEARCH  
PROTOTYPING



PRODUCTION  
DEPLOYMENT



# OPEN SOURCE, COMMUNITY BASED DEEP LEARNING PLATFORM

The screenshot shows the GitHub repository page for PyTorch. The repository name is `pytorch / pytorch`. Key metrics displayed include 26,759 commits, 3,791 branches, 0 packages, 38 releases, and 1,399 contributors. Recent activity includes a commit by Nirav Mehta and a PR by facebook-github-bot. Pull requests listed include Reverting a Docker-related PR, adding a .ctags.d directory, and reverting another Docker-related PR.

Tensors and Dynamic neural networks in Python with strong GPU acceleration <https://pytorch.org>

neural-network autograd gpu numpy deep-learning tensor python machine-learning

26,759 commits 3,791 branches 0 packages 38 releases 1,399 contributors View license

Branch: master ▾ New pull request Find file Clone or download

Nirav Mehta and facebook-github-bot Adding support for manifold files in DBReader (#37727) ... Latest commit acacad2 1 hour ago

.circleci Revert D21585458: [pytorch][PR] [RELAND] .circleci: Improve docker im... 14 hours ago

.ctags.d Add a .ctags.d/ toplevel directory (#18827) 14 months ago

.github Revert D21585458: [pytorch][PR] [RELAND] .circleci: Improve docker im... 14 hours ago



`torch.nn`

NEURAL NETWORKS



`torch.optim`

OPTIMIZERS



`torch.data`

DATASETS &  
DATA LOADERS



`torch.autograd`

AUTO  
DIFFERENTIATION



`torch.vision`

MODELS, DATA



`torch.jit`

TORCH SCRIPT  
DEPLOYMENT



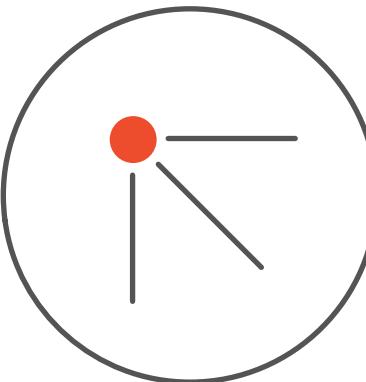
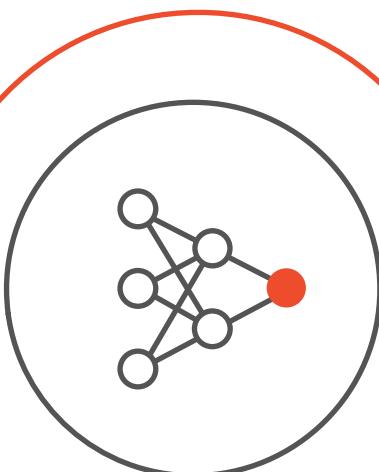


## DESIGN PRINCIPLES & TENETS



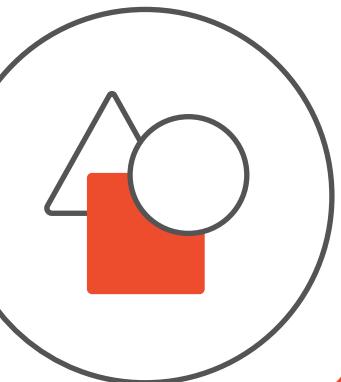
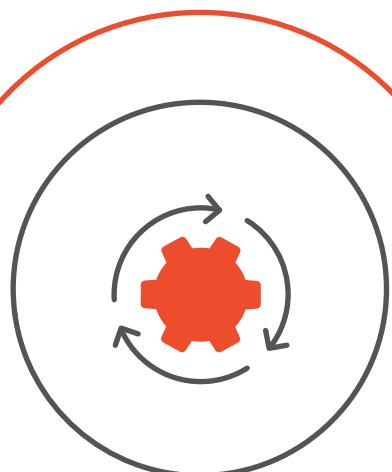
EAGER &  
GRAPH-BASED  
EXECUTION

DYNAMIC  
NEURAL  
NETWORKS



DISTRIBUTED  
TRAINING

HARDWARE  
ACCELERATION



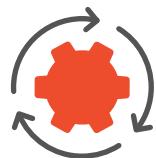
SIMPLICITY  
OVER  
COMPLEXITY



## DEVELOPER EXPERIENCE



FULL FREEDOM TO ITERATE AND EXPLORE THE DESIGN SPACE



CLEAN AND INTUITIVE APIs



A RICH ECOSYSTEM OF TOOLS AND LIBRARIES



```
import torch

class Net(torch.nn.Module):
    def __init__(self):
        self.fc1 = torch.nn.Linear(8, 64)
        self.fc2 = torch.nn.Linear(64, 1)

    def forward(self, x):
        x = torch.relu(self.fc1.forward(x))
        x = torch.dropout(x, p=0.5)
        x = torch.sigmoid(self.fc2.forward(x))
        return x
```

P Y T H O N

```
#include <torch/torch.h>

struct Net : torch::nn::Module {
    Net() : fc1(8, 64), fc2(64, 1) {
        register_module("fc1", fc1);
        register_module("fc2", fc2);
    }

    torch::Tensor forward(torch::Tensor x) {
        x = torch::relu(fc1->forward(x));
        x = torch::dropout(x, /*p=*/0.5);
        x = torch::sigmoid(fc2->forward(x));
        return x;
    }

    torch::nn::Linear fc1, fc2;
};
```

C + +



```
net = Net()

data_loader = torch.utils.data.DataLoader(
    torchvision.datasets.MNIST('./data'))

optimizer = torch.optim.SGD(net.parameters())

for epoch in range(1, 11):
    for data, target in data_loader:
        optimizer.zero_grad()
        prediction = net.forward(data)
        loss = F.nll_loss(prediction, target)
        loss.backward()
        optimizer.step()
    if epoch % 2 == 0:
        torch.save(net, "net.pt")
```

P Y T H O N

```
Net net;

auto data_loader = torch::data::data_loader(
    torch::data::datasets::MNIST("./data"));

torch::optim::SGD optimizer(net.parameters());

for (size_t epoch = 1; epoch <= 10; ++epoch) {
    for (auto batch : data_loader) {
        optimizer.zero_grad();
        auto prediction = net.forward(batch.data);
        auto loss = torch::nll_loss(prediction,
                                    batch.label);
        loss.backward();
        optimizer.step();
    }
    if (epoch % 2 == 0) {
        torch::save(net, "net.pt");
    }
}
```

C + +



## TOOLS & LIBRARIES



Ax



BoTorch



PyTorch3D



CrypTen



Captum

PyTorch

Get Started

Ecosystem

Mobile

Blog

Tutorials

Docs

Resources

Github



## ECOSYSTEM TOOLS

Tap into a rich ecosystem of tools, libraries, and more to support, accelerate, and explore AI development.

[Join the Ecosystem](#)

AdverTorch

A toolbox for adversarial robustness research. It contains modules for generating adversarial examples and defending against attacks.

Albumentations

Fast and extensible image augmentation library for different CV tasks like classification, segmentation, object detection and pose estimation.





Smerity  
@smerity

True to their mission, the [@PyTorch](#) community focused on solving the issues of eager mode w/o impacting interoperability?

Want  
it without

2 Oct 2018

Jeremy Howard  
@jeremyphoward

At the [@PyTorch](#) developer conference, I was part of a fascinating panel with [@clattner\\_llvm](#), Yangqing Jia, and Noah Goodman, Expertly moderated by [@soumithchintala](#). Here it is!

Comment Reply Heart

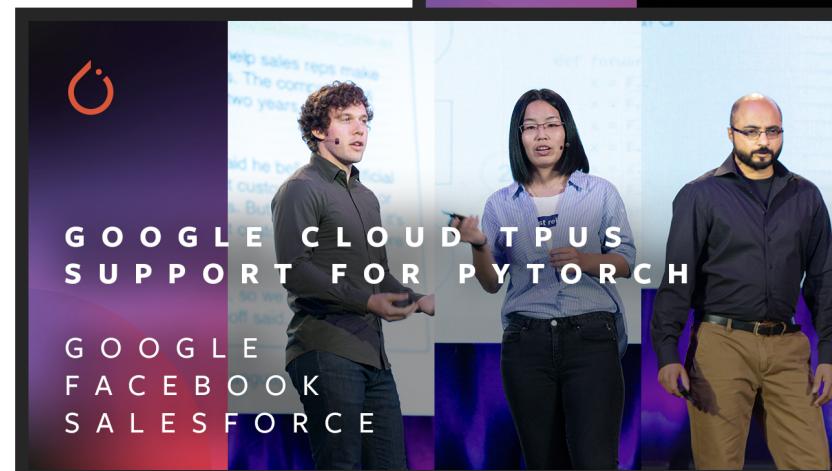
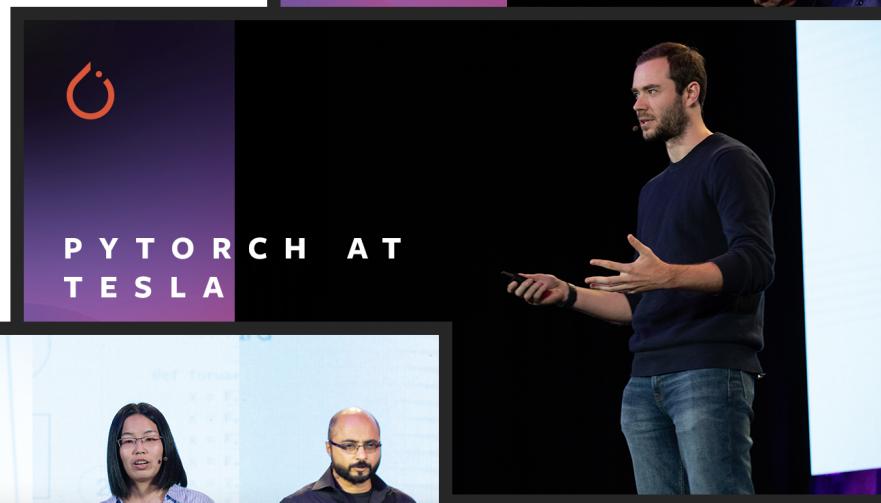
10 Oct 2018



Alfredo Canziani  
@alfcnz

## Atcold/pytorch-Deep-Learning-Minicourse

ing with PyTorch. Contribute to Deep-Learning-Minicourse development on GitHub.





One simple graphic: Researchers love PyTorch



Home > News > Preferred Networks Migrates its Deep Learning Research Platform to PyTorch

## News

tutorials that cover PyTorch, TensorFlow, intelligence conference in San Jose,



rise, which drew more than age of several machine learning indicated they used TensorFlow re using PyTorch or Keras.

Research lab  
(source: SMU Libraries D



ABOUT PROGRESS RESOURCES BLOG



JANUARY 30, 2020 • 1 MINUTE READ

# OpenAI → PyTorch

We are standardizing OpenAI's deep learning framework on PyTorch. In the past, we implemented projects in many frameworks depending on their relative strengths. We've now chosen to standardize to make it easier for our team to create and share optimized implementations of our models.

Share

O: Toru  
al  
iborate  
ment of  
PFN



6

~1,491

CONTRIBUTORS

50%+

YOY GROWTH

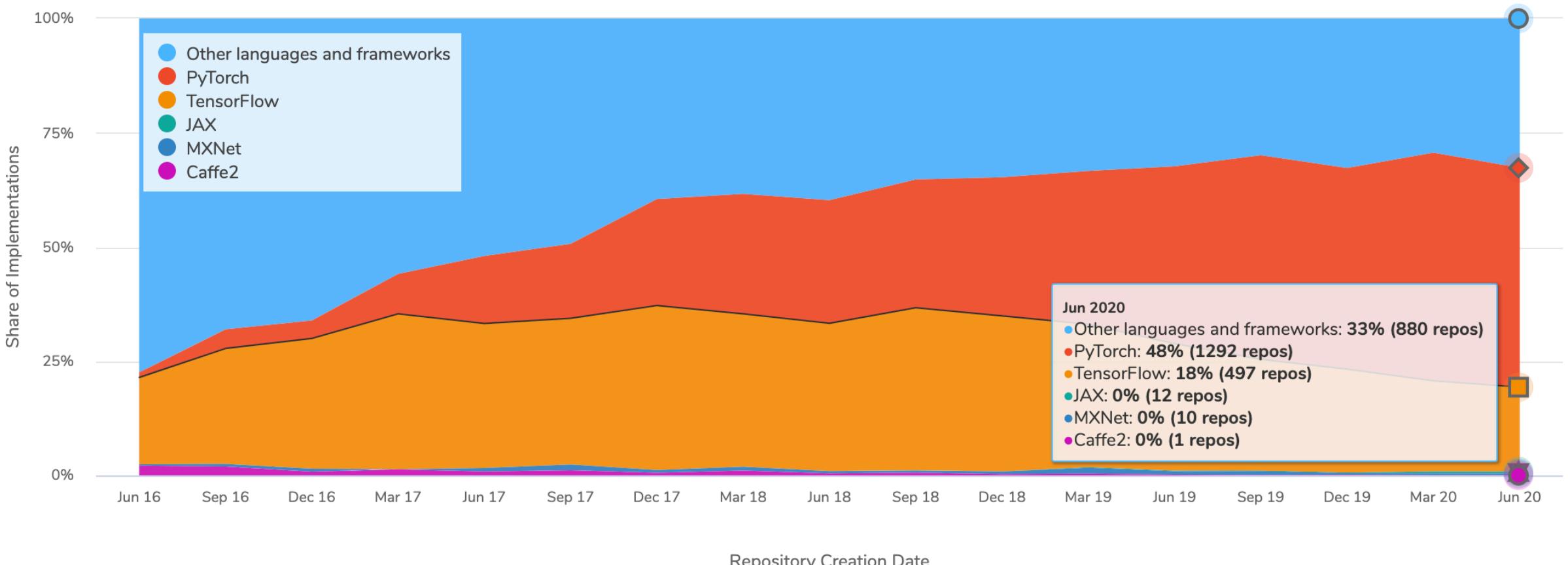
32K+

PYTORCH FORUM USERS



# GROWING USAGE IN OPEN SOURCE

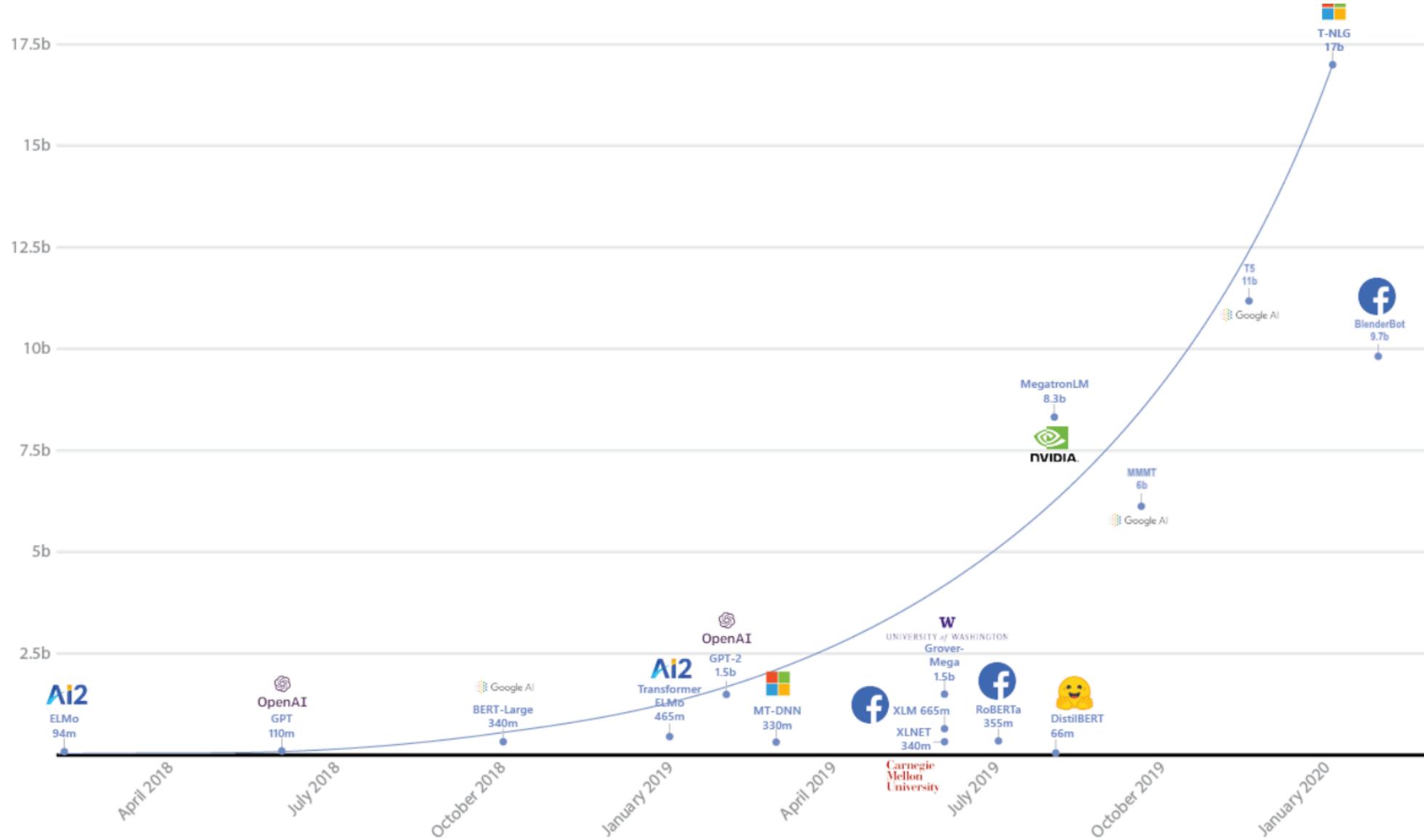
Paper Implementations grouped by framework







## PARAMETER COUNTS CONTINUE TO GROW





## PRUNING

State-of-the-art deep learning techniques rely on over-parametrized models that are hard to deploy.

Identify optimal techniques to compress models by reducing the number of parameters without sacrificing accuracy.

```
new_model = LeNet()
for name, module in new_model.named_modules():
    # prune 20% of connections in all 2D-conv layers
    if isinstance(module, torch.nn.Conv2d):
        prune.l1_unstructured(module, name='weight', amount=0.2)
    # prune 40% of connections in all linear layers
    elif isinstance(module, torch.nn.Linear):
        prune.l1_unstructured(module, name='weight', amount=0.4)

    # to verify that all masks exist
    print(dict(new_model.named_buffers()).keys())
```



# QUANTIZATION

EXPERIMENTAL

Neural networks inference is expensive

IoT and mobile devices have limited resources

Quantizing models enables efficient inference at scale

```
model = ResNet50()
model.load_state_dict(torch.load("model.pt"))

qmodel = quantization.prepare(
    model, {"": quantization.default_qconfig})

qmodel.eval()
for batch, target in data_loader:
    model(batch)

qmodel = quantization.convert(qmodel)
```



## EXAMPLE MODELS | W/ QUANTIZATION

	fp32 accuracy	int8 accuracy change	Technique	CPU inference speed up
ResNet50	76.1 Top-1, Imagenet	-0.2 75.9	Post Training	2x 214ms → 102ms, Intel Skylake-DE
MobileNetV2	71.9 Top-1, Imagenet	-0.3 71.6	Quantization-Aware Training	4x 75ms → 18ms OnePlus 5, Snapdragon 835
Translate / FairSeq	32.78 BLEU, IWSLT 2014 de-en	0.0 32.78	Dynamic (weights only)	4x for encoder Intel Skylake-SE

These models and more available on TorchHub - <https://pytorch.org/hub/>





## EXPENSIVE & EXPERIMENTAL

Unlike data pipelines, majority of ML jobs are ad hoc

Long running jobs complicate demand prediction & control

## Cost & Efficiency

ROI for jobs hard to estimate

Sub-linear scaling + huge scale = an easy way to waste resources



# PYTORCH ELASTIC

Enables users to write fault tolerant and elastic distributed PyTorch jobs

## Use cases

### *Fault tolerance*

Run on non-HPC cluster (e.g. cloud)

Mission critical production job

### *Dynamic Capacity Management:*

Run on leased capacity that can be preempted (e.g. AWS spot instances)

Shared pools where the pool size can change dynamically based on demand (e.g. autoscaling)

Open sourced 0.1.0.rc - <https://github.com/pytorch/elastic>

Integrated with Classy Vision - [https://classyvision.ai/tutorials/pet\\_aws](https://classyvision.ai/tutorials/pet_aws)

The screenshot shows a GitHub repository page for 'pytorch / elastic'. The repository has 10 issues, 4 pull requests, and 0 actions. It has 0 projects and 0 wiki pages. The current branch is 'master'. The file being viewed is 'elastic / USAGE.md'. The file has 165 lines (127 sloc) and is 7.2 KB. The content of the file is visible, starting with a large bold heading 'Usage'.

**Usage**

torchelastic requires you to implement a `state` object and a `train_step` to [how torch elastic works](#).

While going through the sections below, refer to the imagenet [example](#) for

## Implement state

The `State` object has two categories of methods that need to be implemented:

### `sync()`



# REALLY LARGE SCALE

Tens of billions of training examples

Some models larger than size of machine

Hundreds of experimental runs competing for resources

More than 1000+ different production models



## PYTORCH RPC (REMOTE PROCEDURE CALL)

Enables applications to run functions remotely, and automatically handles autograd if necessary.

### Use cases

*Scale out applications*

Parameter Server Framework

In RL, multiple observers streaming states and rewards to the agent for policy training

*Distributed model parallelism*

Allow large models to span multiple machines

Hogwild! training



## PYTORCH RPC COMPONENTS

### RPC

Run user code with given args on the specified destination

*Remote Reference (RRef)*

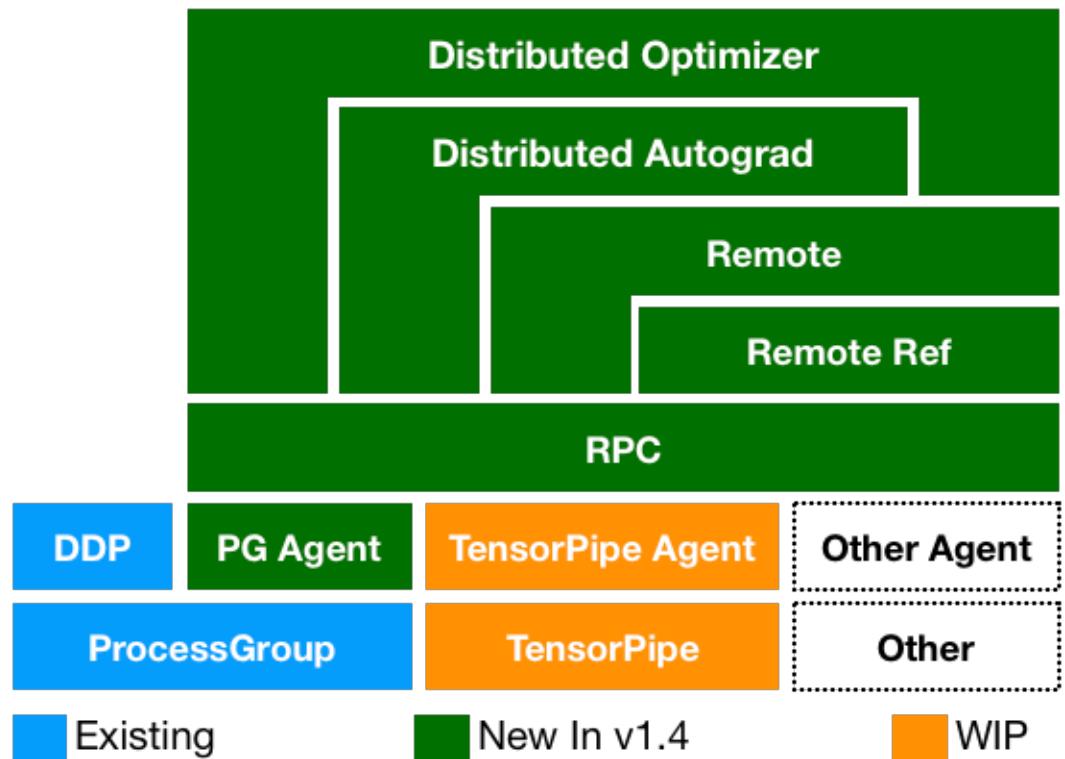
Tracks and maintains objects owned by a remote worker.

*Distributed Autograd*

Connects autograd graphs on different workers into one global graph and provides a similar backward API.

*Distributed Optimizer*

Automatically handles RRef of subnets and expose a similar API as local optimizers.







## INFERENCE AT SCALE

Deploying and managing models in production is difficult.

Some of the pain points include:

Loading and managing multiple models, on multiple servers or end devices

Running pre-processing and post-processing code on prediction requests.

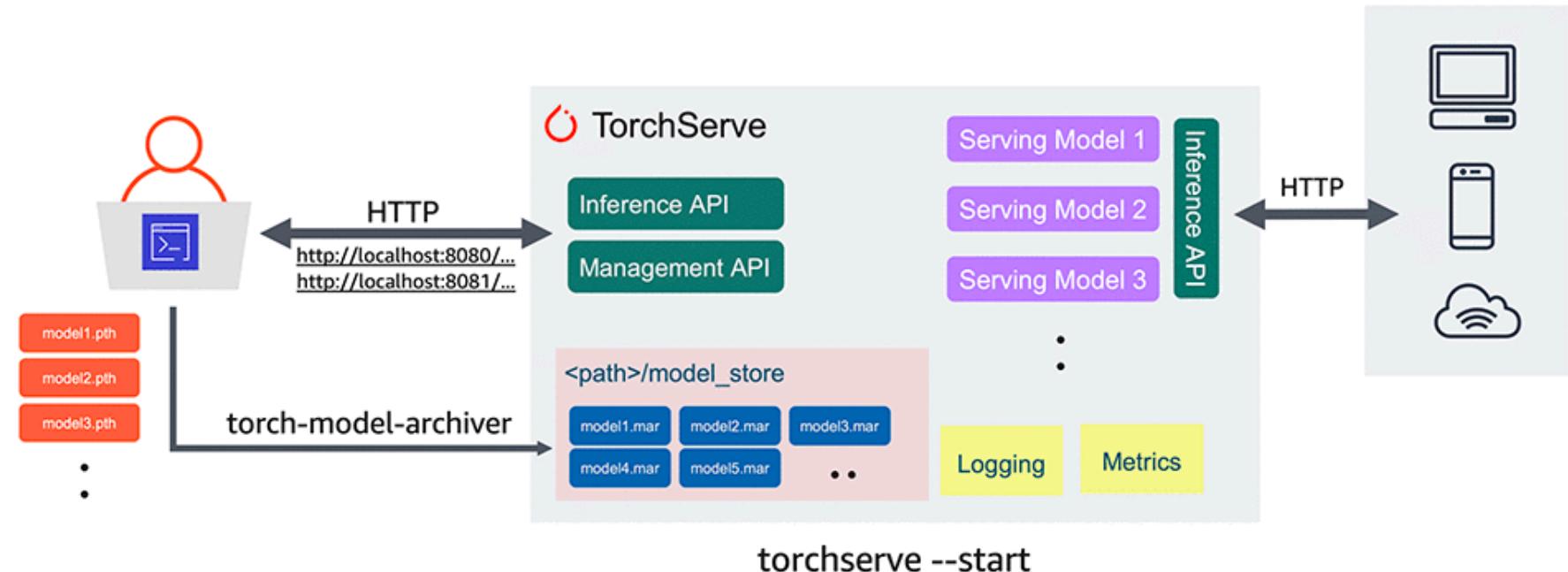
How to log, monitor and secure predictions

What happens when you hit scale?



# TORCHSERVE

EXPERIMENTAL



- Default handlers for common use cases (e.g., image segmentation, text classification) along with custom handlers support for other use cases
- Model versioning and ability to roll back to an earlier version
- Automatic batching of individual inferences across HTTP requests
- Logging including common metrics, and the ability to incorporate custom metrics
- Robust HTTP APIs - Management and Inference





## PYTORCH 1.6 RELEASED - END OF JULY

- Automatic mixed precision (AMP) training is now natively supported and a stable feature - thanks for NVIDIA's contributions;
- Native TensorPipe support now added for tensor-aware, point-to-point communication primitives built specifically for machine learning;
- New profiling tools providing tensor-level memory consumption information; and
- Numerous improvements and new features for both distributed data parallel (DDP) training and the remote procedural call (RPC) packages.



## DOMAIN LIBRARIES - SHIPPED WITH PYTORCH 1.6

### **torchvision 0.7**

- New pretrained models - Segmentation models FCN ResNet50 and DeepLabV3 ResNet50, trained on COCO.
- Added support for AMP (Automatic Mixed Precision) autocasting for torchvision models and operators

### • **torchtext 0.7**

- Experimental overhaul of torchtext's datasets - easier, compatible PyTorch's DataLoader and Sampler, and simpler out-of-the-box with preprocessing defaults.
- A new MultiheadAttention module which provides more flexibility for innovative transformer variants.
- Windows is now officially supported.
- The SentencePiece models have been extended with torchscript support.
- A pre-trained BERT example pipeline, along with a question answering fine-tuning example and two new Q&A datasets.

### • **torchaudio 0.6**

- Official support for Windows.
- New model module (with wav2letter included), new functionals (contrast, cvm, dcshift, overdrive, vad, phaser, flanger, biquad), datasets (GTZAN, CMU), and a new optional sox backend with support for torchscript.



GET STARTED

PYTORCH.ORG

The screenshot shows the 'Get Started' section of the PyTorch website. At the top, there's a navigation bar with links for 'Get Started', 'Ecosystem', 'Mobile', 'Blog', 'Tutorials', 'Docs', 'Resources', 'GitHub', and a search icon. Below the navigation is a large dark banner with the word 'GET STARTED' in white. A sub-instruction reads: 'Select preferences and run the command to install PyTorch locally, or get started quickly with one of the supported cloud platforms.' Below the banner, there are four buttons: 'Start Locally' (highlighted in red), 'Start via Cloud Partners', 'Previous PyTorch Versions', and 'Mobile'. On the left side, there's a sidebar with sections for 'Shortcuts', 'Prerequisites' (listing 'macOS Version', 'Python', 'Package Manager'), 'Installation' (listing 'Anaconda', 'pip'), and 'Verification' (listing 'Building from source', 'Prerequisites'). To the right of the sidebar, the main content area starts with a heading 'START LOCALLY'. It explains how to select preferences and run the install command, mentioning that 'Stable' represents the most currently tested and supported version. It also notes that 'Preview' is available for the latest, not fully tested and supported, 1.5 builds. It recommends Anaconda as the package manager. There's also a note about LibTorch being only available for C++. Below this text is a table for selecting PyTorch Build preferences. The 'Stable (1.5)' option is selected, showing options for Linux, Mac, Windows, Conda, Pip, LibTorch, and Source. The 'Language' row shows Python selected, with options for C++ / Java.

PyTorch Build	Stable (1.5)	Preview (Nightly)		
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python	C++ / Java		



# EDUCATIONAL RESOURCES

The screenshot displays the Udacity website interface. At the top, the Udacity logo is visible along with navigation links: Explore ▾, Catalog, Paths, Nanodegree, Career ▾, For Enterprise, Sign In, and a prominent blue "Get Started" button.

The main content area shows a "FREE COURSE" titled "Intro to Deep Learning with PyTorch" by "facebook Artificial Intelligence". The course description states: "Use PyTorch to implement your first deep neural network". A blue "START FREE COURSE" button is present.

To the right, a large banner for the "NANODEGREE PROGRAM" titled "Deep Learning" is displayed, with the tagline "Build Deep Learning Models Today". Below this, a call-to-action reads "Build cutting-edge AI projects supported by dedicated mentors. >>>".

Below the course listing, a section titled "About this Course" provides a brief description of the course content, mentioning practical experience with PyTorch through coding exercises and projects implementing state-of-the-art AI applications such as style transfer and text generation.

On the right side, there is a summary table with four rows:

COURSE COST	TIMELINE	SKILL LEVEL
Free	Approx. 2 Month	Intermediate
INCLUDED IN PRODUCT		
Rich Learning Content	Taught by Industry Pros	
Interactive Quizzes	Self-Paced Learning	

O





# EDUCATIONAL RESOURCES

The screenshot shows the fast.ai website with a dark background. The title "fast.ai" is prominently displayed in large white letters. Below it, the tagline "Making neural nets uncool again" is written in smaller white text. A navigation menu includes links for "Home", "About", "Our MOOC", and "Posts by Topic". At the bottom, there is a copyright notice: "© fast.ai 2019. All rights reserved."

**Our online courses (all are free and have no ads):**

- [Practical Deep Learning for Coders](#)
- [Cutting Edge Deep Learning for Coders](#)
- [Introduction to Machine Learning for Coders](#)
- [Computational Linear Algebra](#)

**Our software: [fastai v1 for PyTorch](#)**

**Take our course in person, March-April 2019 in SF: [Register here](#)**

**fast.ai in the news:**

- The Economist: [New schemes teach the masses to build AI](#)
- MIT Tech Review: [The startup diversifying AI workforce beyond just "techies"](#)
- The New York Times: [Finally, a Machine That Can Finish Your Sentence](#)
- The Verge: [An AI speed test shows clever coders can still beat tech\\_giants like Google and Intel](#)
- MIT Tech Review: [A small team of student AI coders beats Google's machine-learning code](#)
- Forbes: [Artificial Intelligence Education Transforms The Developing World](#)
- ZDNet: [fast.ai's software could radically democratize AI](#)

---

## **16 Things You Can Do to Make Tech More Ethical, part 1**

22 Apr 2019 *Rachel Thomas*



ALSO IN CHINESE :)

<https://pytorch.apacheCN.org/>



The screenshot shows a web browser displaying the PyTorch Chinese documentation. The page title is "Pytorch 中文教程". The main heading is "PyTorch". Below the heading, a subtext states: "PyTorch 是一个针对深度学习, 并且使用 GPU 和 CPU 来优化的 tensor library (张量库)". A navigation grid below the subtext contains six items:

<a href="#">Pytorch 1.0 中文版本</a> PyTorch	<a href="#">Pytorch 最新 英文教程</a> PyTorch	<a href="#">Pytorch 最新 英文文档</a> PyTorch
<a href="#">Pytorch 0.4 中文版本</a> PyTorch	<a href="#">Pytorch 0.3 中文版本</a> PyTorch	<a href="#">Pytorch 0.2 中文版本</a> PyTorch

At the bottom of the page, there is a note: "欢迎任何人参与和完善: 一个人可以走的很快, 但是一群人却可以走的更远。" followed by a bulleted list:

- 在线阅读
- ApacheCN 机器学习交流群 629470233
- ApacheCN 学习资源



AND KOREAN :):)

<https://pytorch.kr/>

The screenshot shows a web browser window with the URL 'pytorch.kr' in the address bar. The page title is 'PyTorch 문서 번역 프로젝트'. The main heading reads 'FROM RESEARCH TO PRODUCTION' over a background image of a city at night. Below the heading, there is a button labeled '시작하기 >'. At the bottom, there is a call-to-action '튜토리얼 문서를 번역하는 오픈소스 컨트리뷰톤에 참여해보세요' with a left arrow icon.

주요 기능 및 성능

# JOIN THE PYTORCH DEVELOPER COMMUNITY

---



[PyTorch.org](https://pytorch.org)



[Twitter.com/pytorch](https://twitter.com/pytorch)



[Youtube.com/pytorch](https://youtube.com/pytorch)



[Facebook.com/pytorch](https://facebook.com/pytorch)



[Medium.com/pytorch](https://medium.com/pytorch)

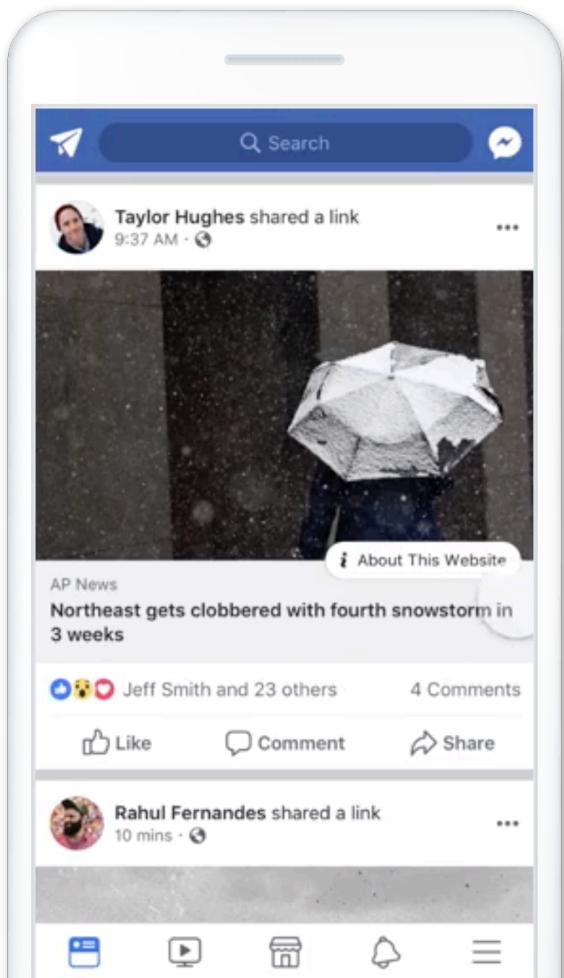




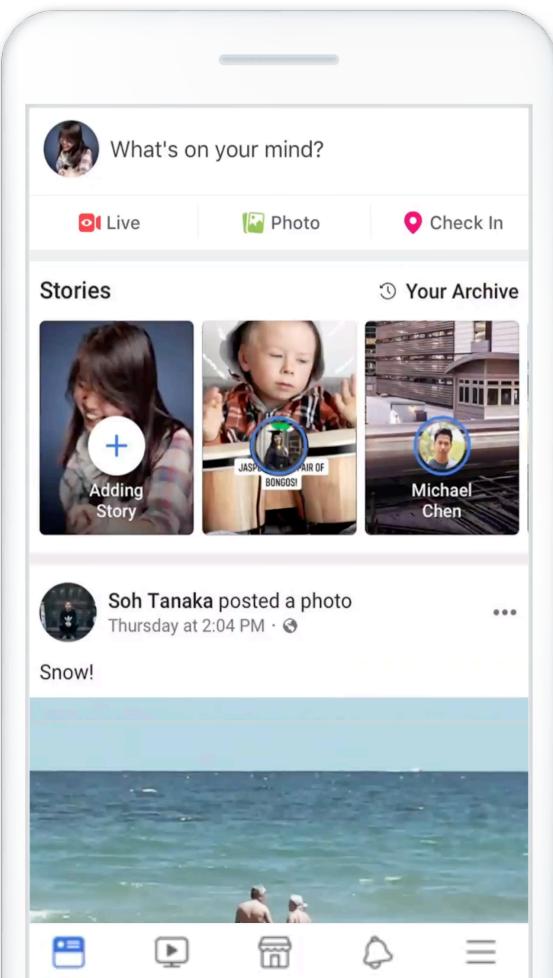


# RECOMMENDATION USE CASES

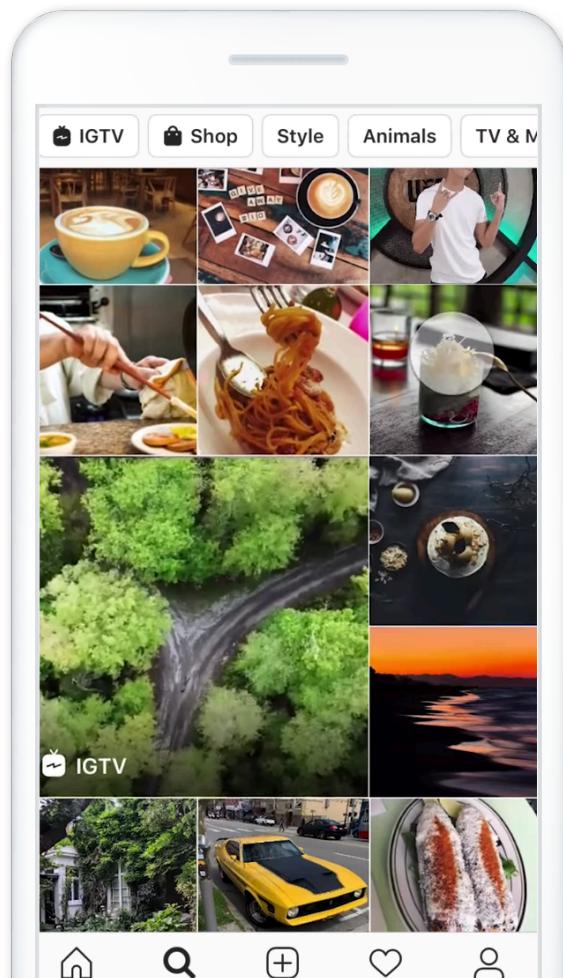
News Feed Ranking



Stories Ranking

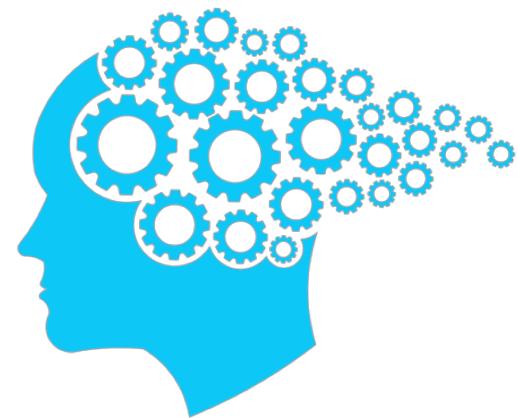


Instagram Explore



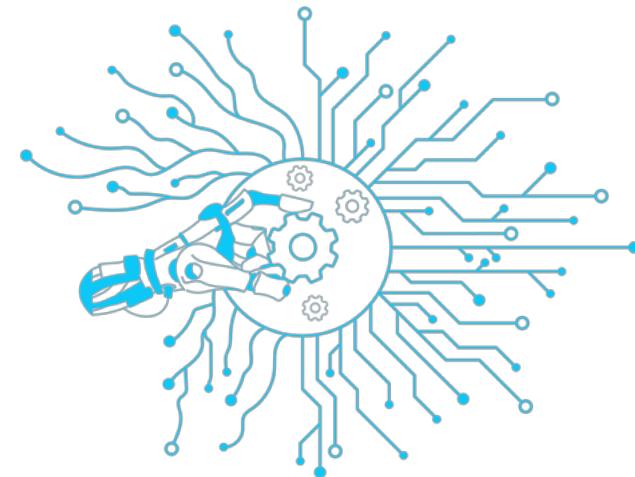
6

50%

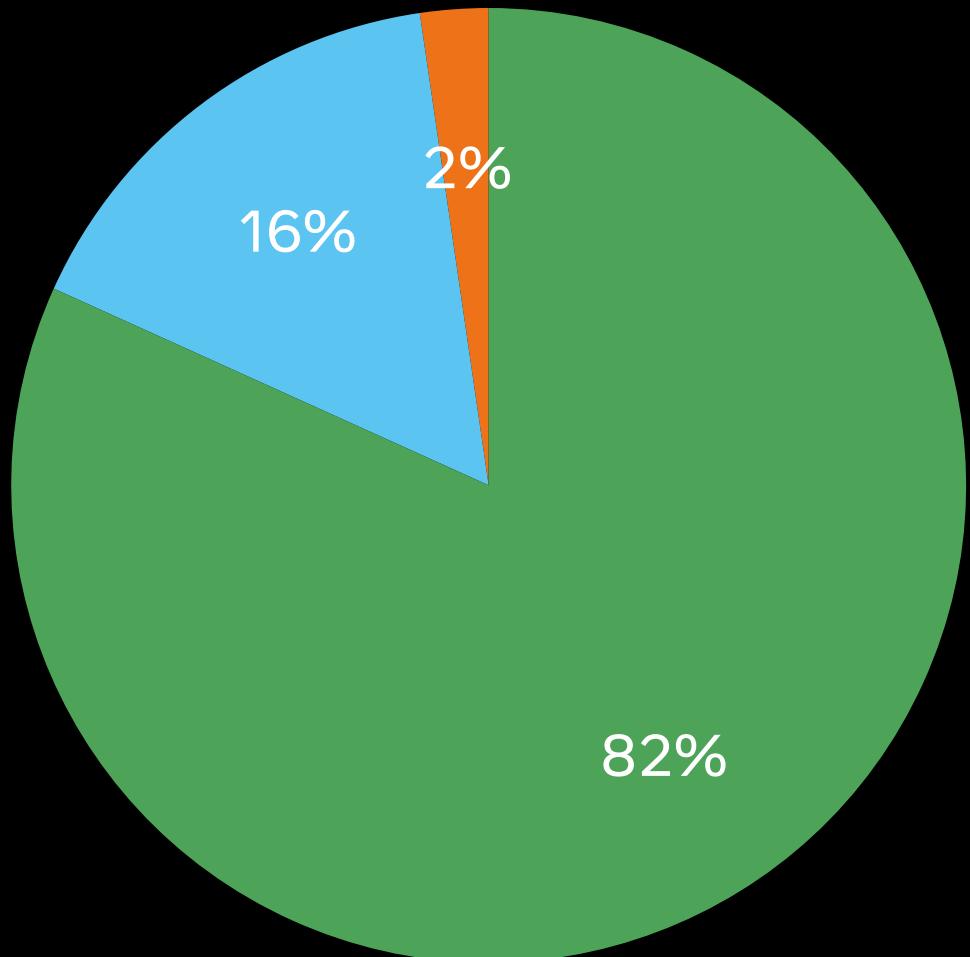


of all AI Training Cycles

80%



of all AI Inference Cycles

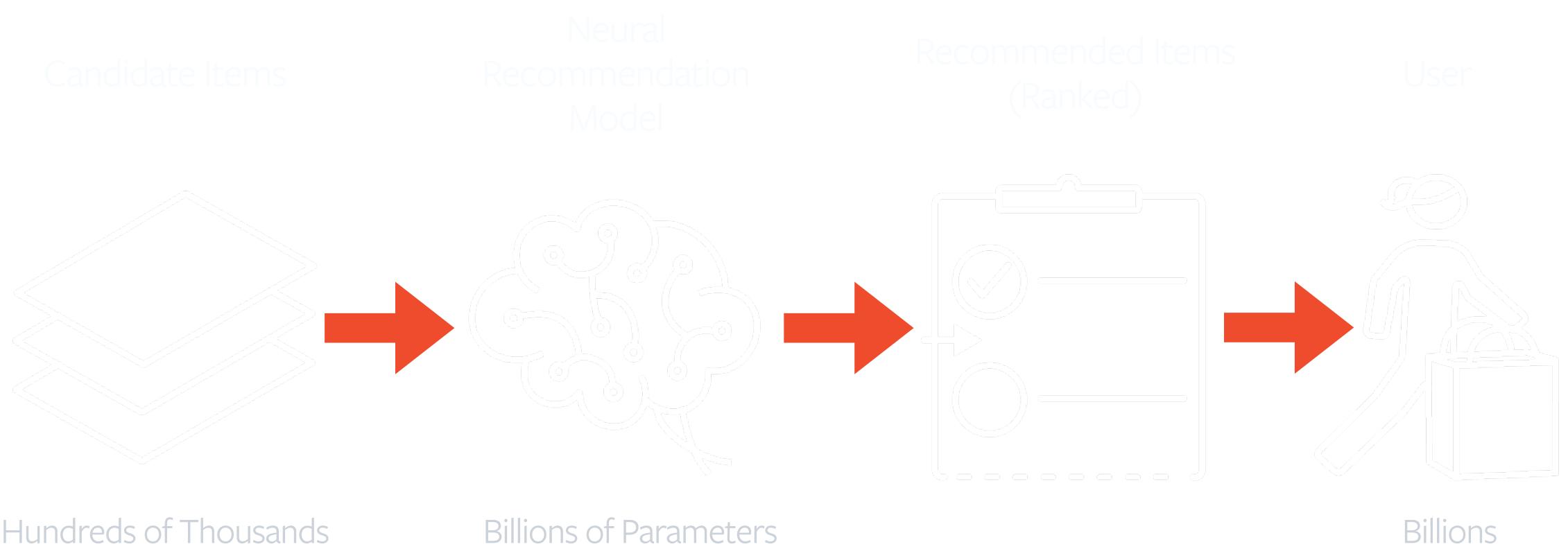


Why The  
Disconnect?

● Computer Vision

● RNN Translation

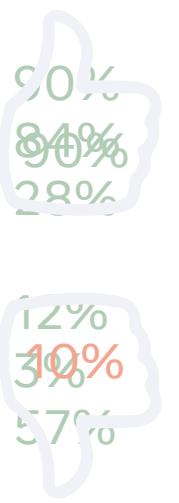
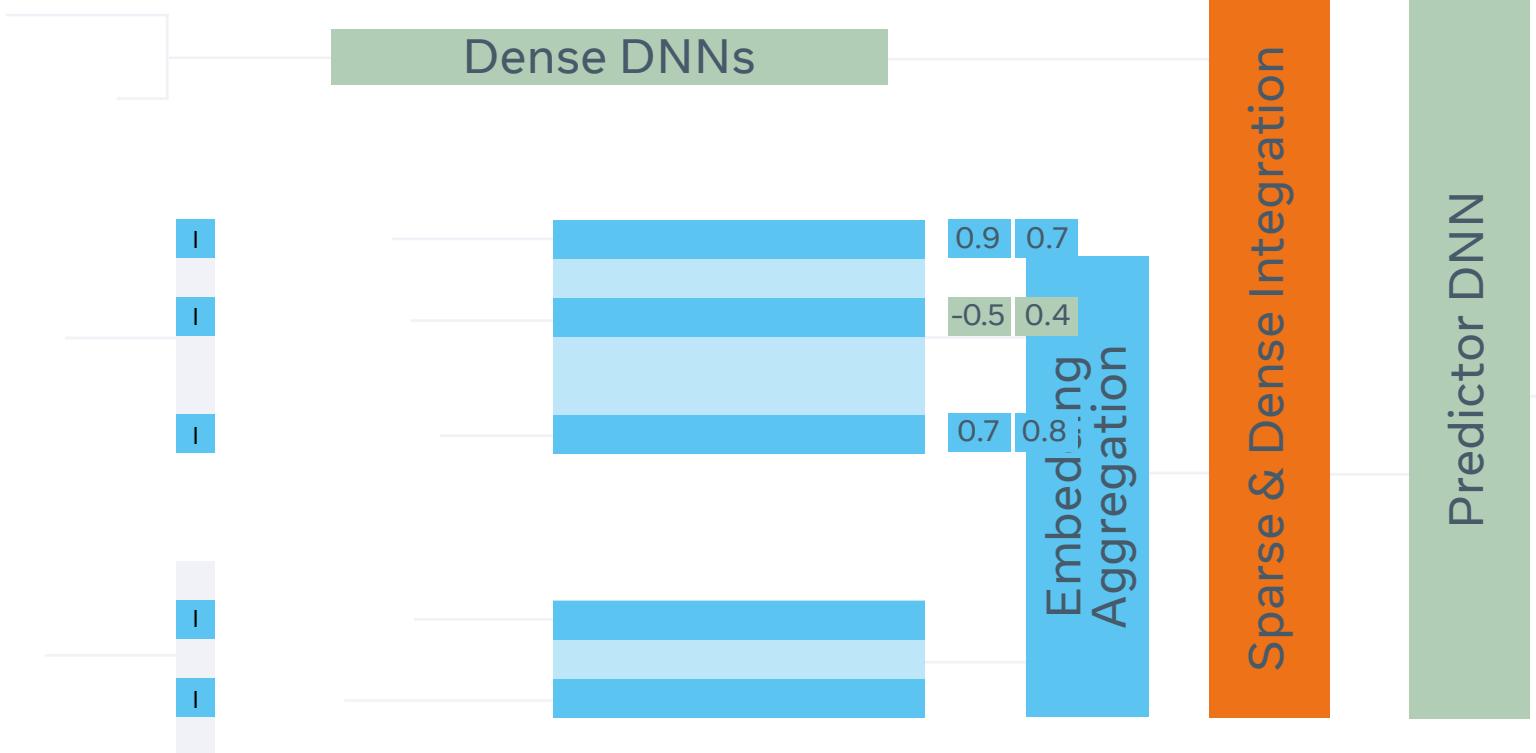
● Recommendation





Continuous  
(dense)  
features

Categorical  
(sparse)  
features

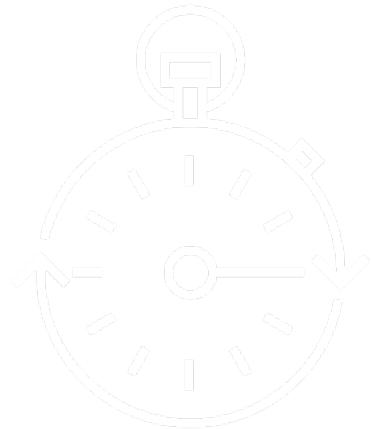


6

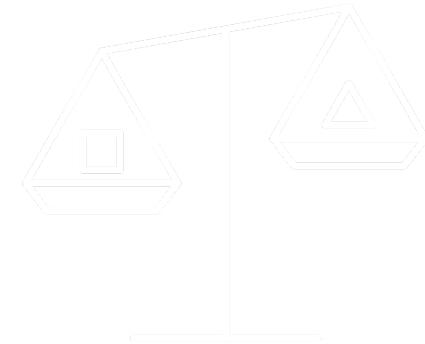
High Throughput

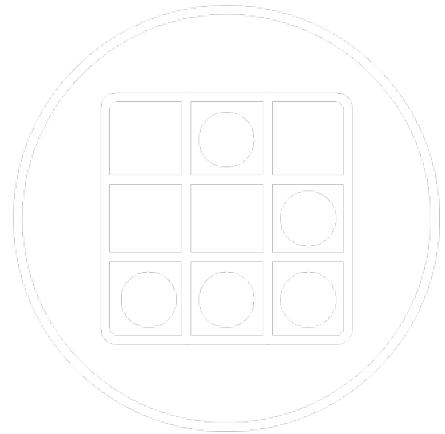


Low Latency

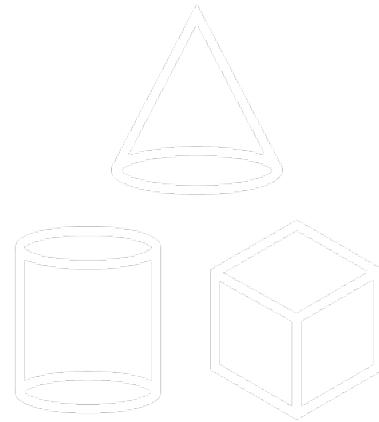


Latency-Bounded  
Throughput

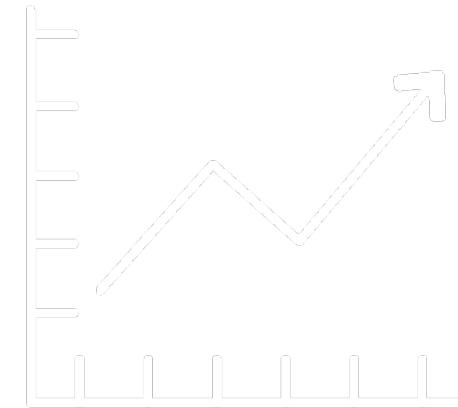




Embedding Tables



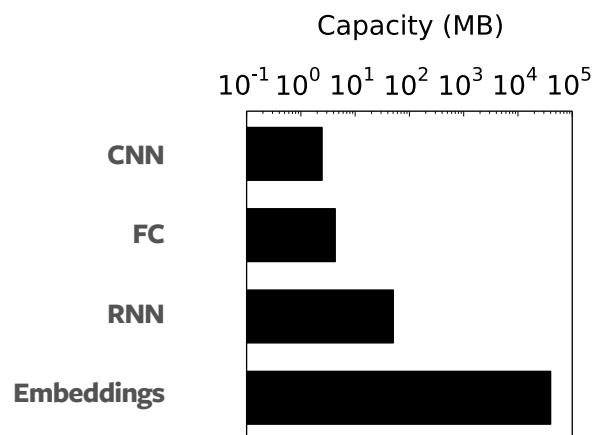
Model Heterogeneity



Performance Variance

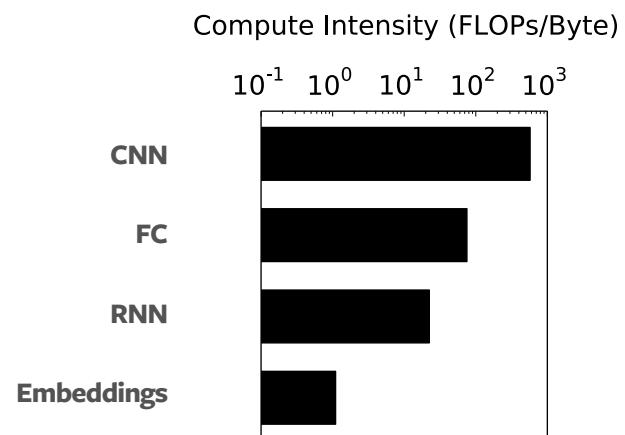
# CHALLENGES: EMBEDDING TABLES

## Storage Capacity



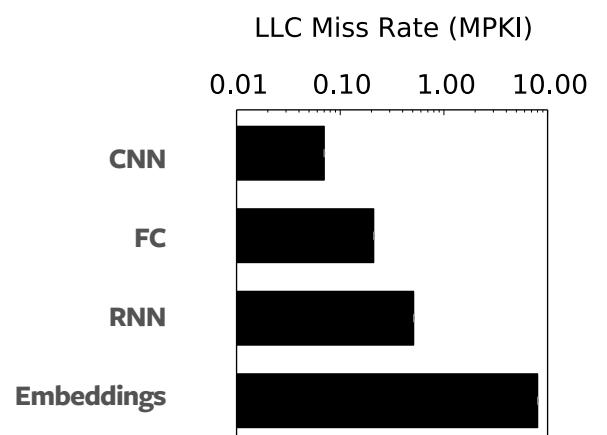
Orders of Magnitude  
Larger

## Compute Intensity



Orders of Magnitude  
Fewer FLOPS/Byte

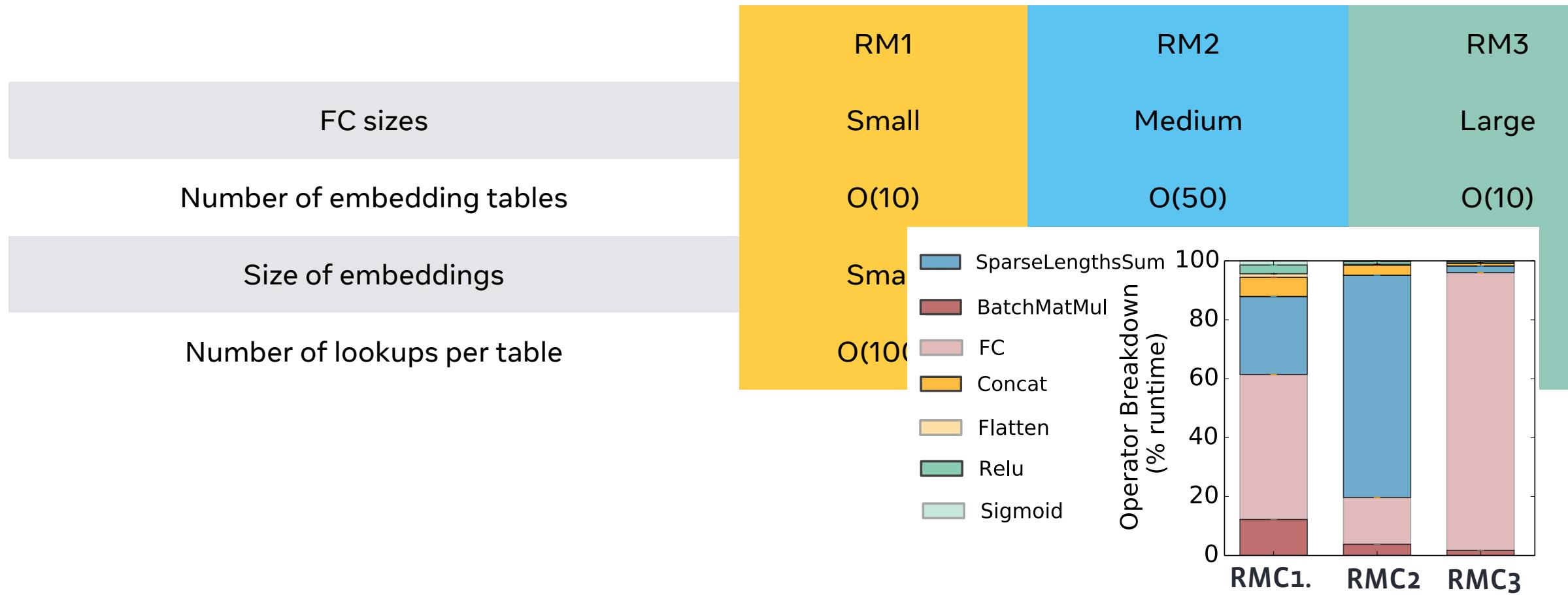
## Memory Irregularity



Sparse, Irregular  
Memory Accesses

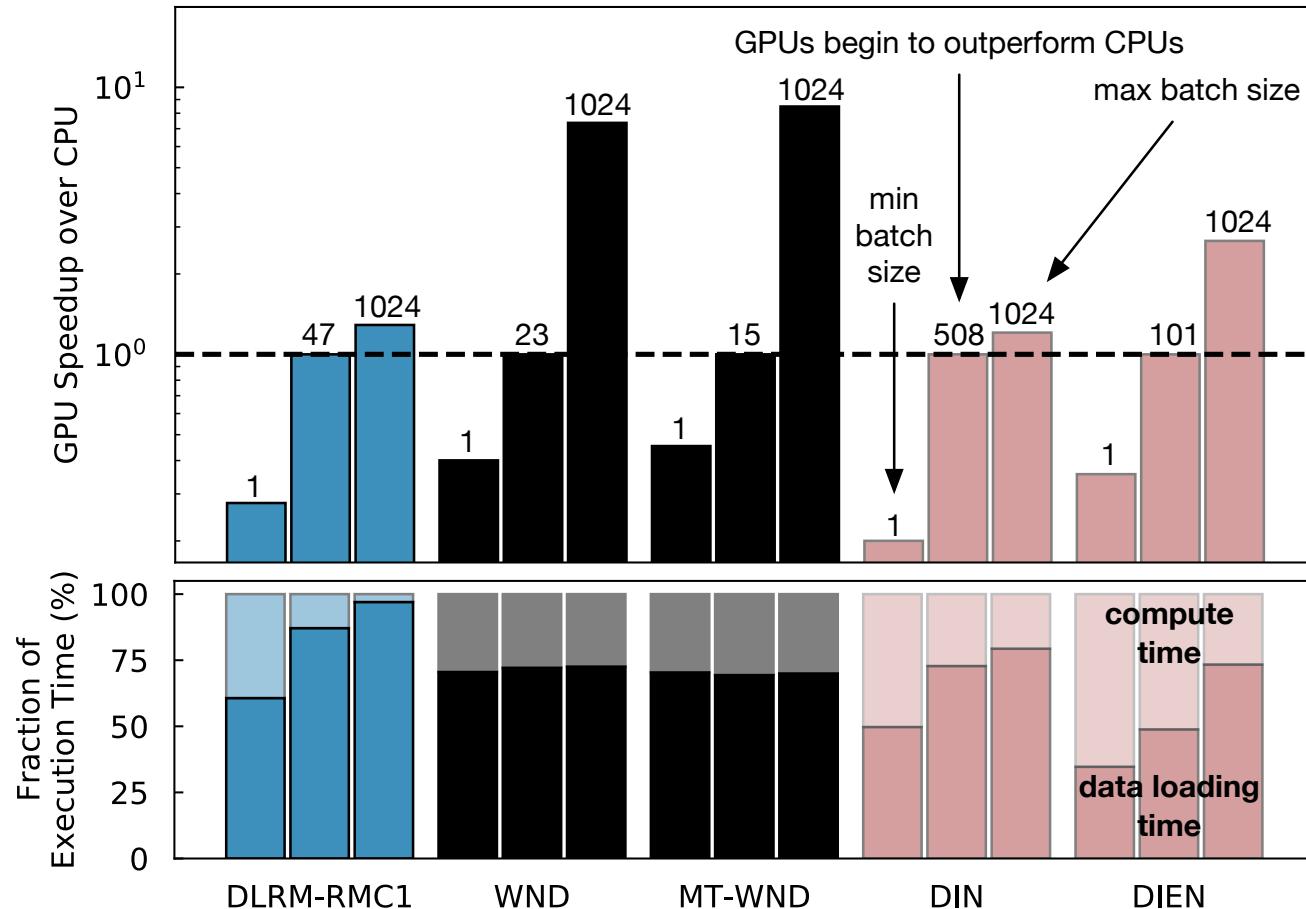
# CHALLENGE: MODEL HETEROGENEITY

Three Facebook Recommendation Models



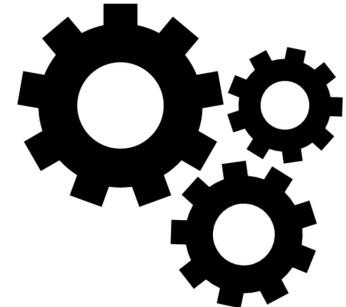
# CHALLENGE: OPTIMAL SYSTEM CONFIG VARIES

*Batch Sizes, Compute Platforms*



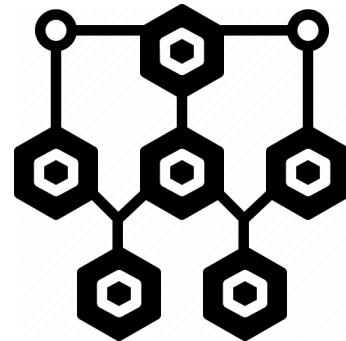
# HOW TO BUILD RECOMMENDER SYSTEMS

Personalization API Builders  
AutoML



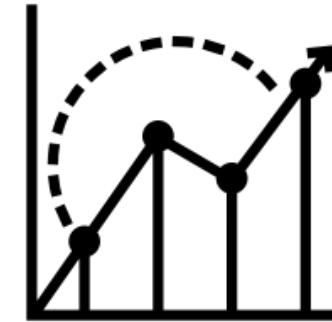
AWS Personalize

Frameworks & Libraries



Microsoft Recommenders  
Nvidia Merlin

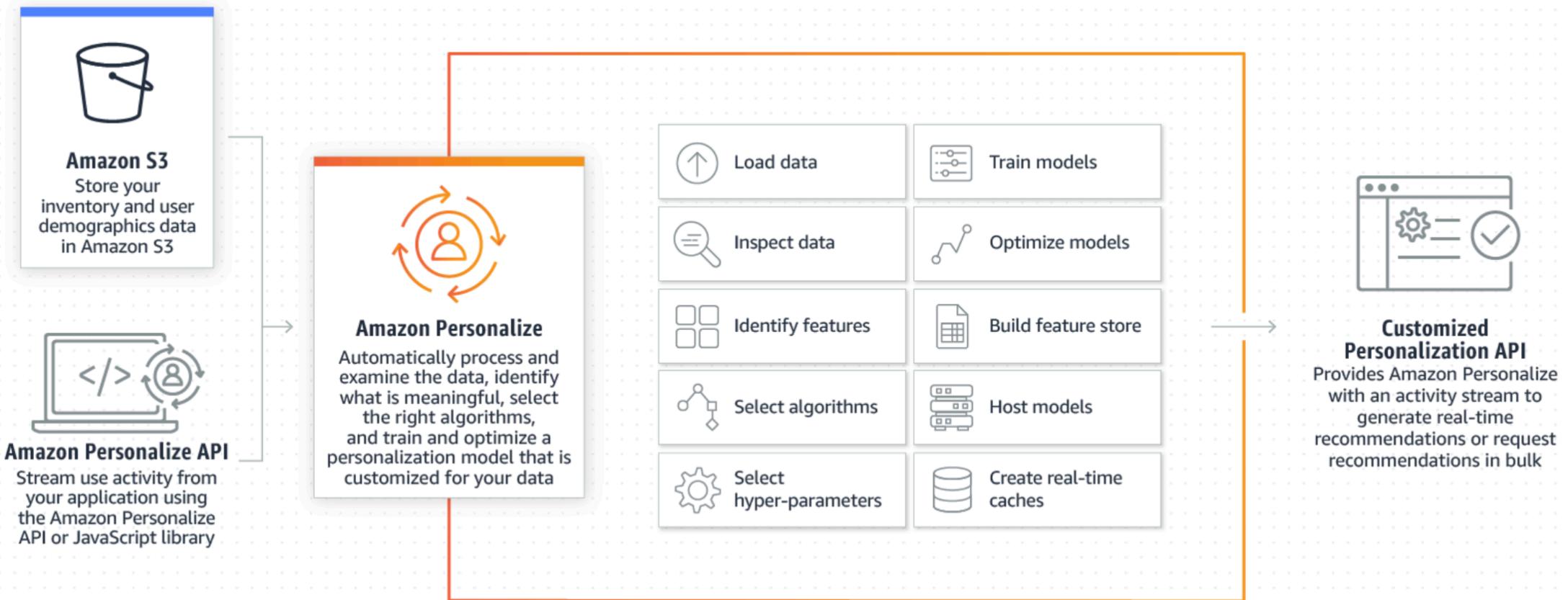
Models & Benchmarks



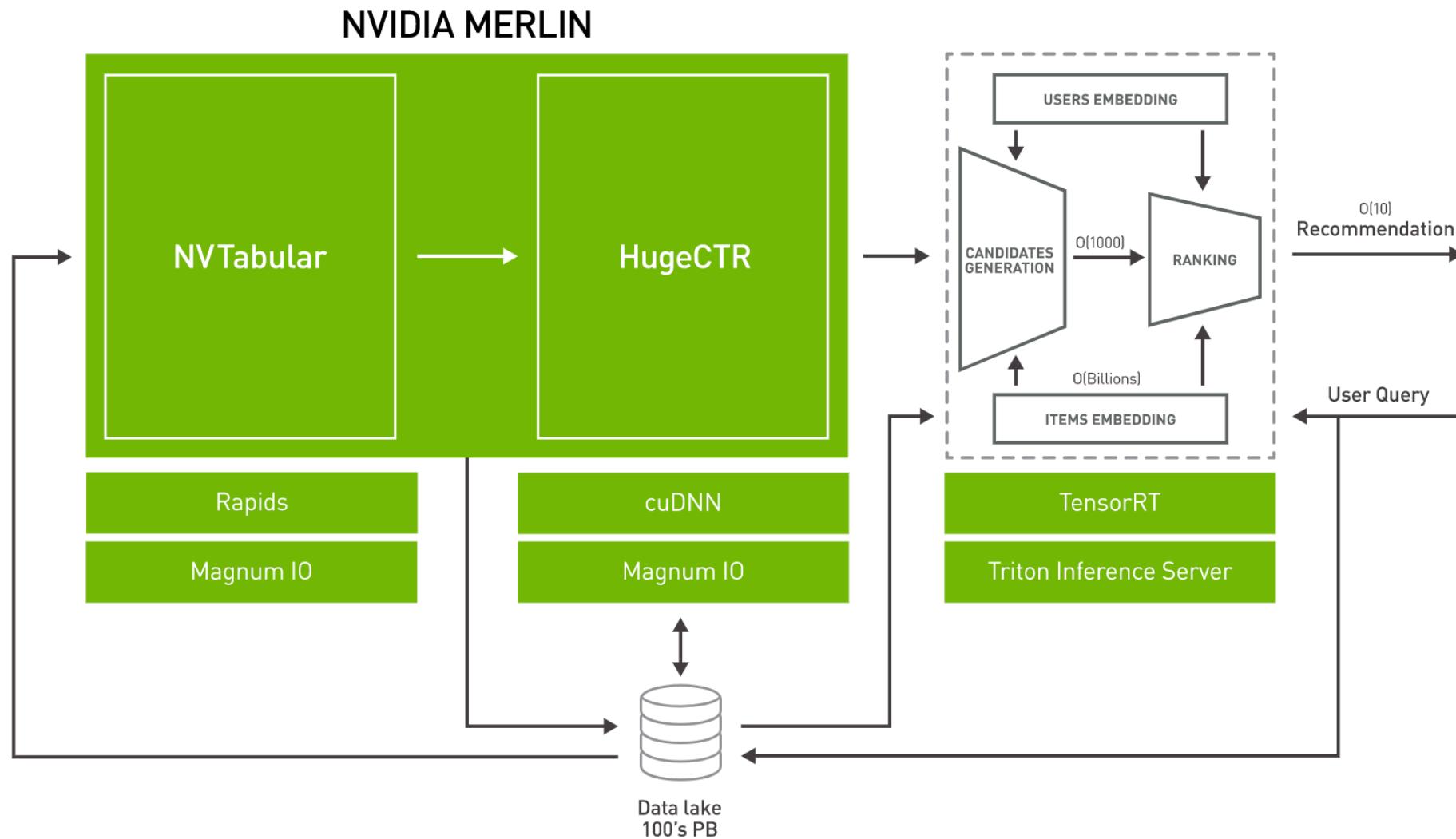
Facebook DLRM  
MLPerf

Open Datasets - Criteo Ads Dataset

# AWS PERSONALIZE API BUILDER - AUTOML BASED



# NVIDIA MERLIN RECOMMENDER SYSTEMS FRAMEWORK



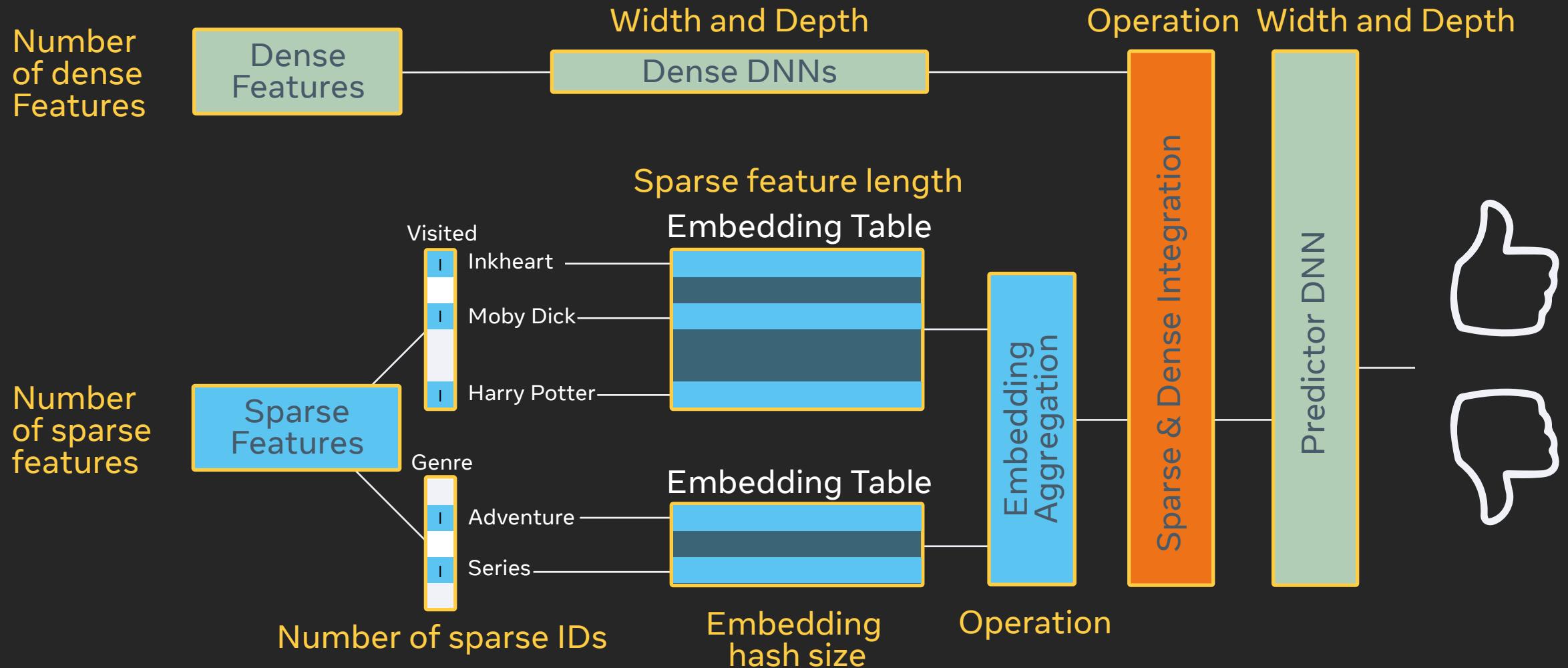


# MICROSOFT RECOMMENDERS LIBRARY

Algorithm	Environment	Type	Description
Simple Algorithm for Recommendation (SAR)*	Python CPU	Collaborative Filtering	Similarity-based algorithm for implicit feedback dataset
Surprise/Singular Value Decomposition (SVD)	Python CPU	Collaborative Filtering	Matrix factorization algorithm for predicting explicit rating feedback in datasets that are not very large
Neural Collaborative Filtering (NCF)	Python CPU / Python GPU	Collaborative Filtering	Deep learning algorithm with enhanced performance for implicit feedback
Restricted Boltzmann Machines (RBM)	Python CPU / Python GPU	Collaborative Filtering	Neural network based algorithm for learning the underlying probability distribution for explicit or implicit feedback
FastAI Embedding Dot Bias (FAST)	Python CPU / Python GPU	Collaborative Filtering	General purpose algorithm with embeddings and biases for users and items
Alternating Least Squares (ALS)	PySpark	Collaborative Filtering	Matrix factorization algorithm for explicit or implicit feedback in large datasets, optimized by Spark MLLib for scalability and distributed computing capability
Vowpal Wabbit Family (VW)*	Python CPU (train online)	Collaborative, Content-Based Filtering	Fast online learning algorithms, great for scenarios where user features / context are constantly changing

# DLRM: Deep Learning Recommendation Model

A Configurable Benchmark for E2E Models





## RESEARCH TOOLS FOR RECOMMENDERS

- DLRM: Configurable Benchmark for E2E Models
- MLPerf includes DLRM + Criteo Ads open dataset (Kaggle, 1 TB)
- Ecosystem partners - Optimized models and workflows
  - Nvidia optimized DLRM on Ampere GPUs
  - Intel optimized DLRM with BFloat16
  - DLRM on TPUs with GCP, Reference architectures with Azure, AWS (coming soon)
- Papers with code - search for recommendation systems
- PyTorch 1.6 with support for building Heterogenous (DDP + RPC) models
  - DDP for Dense features, DMP for Sparse features



# RECOMMENDATION SYSTEMS ARE IMPORTANT

1

**Are Important**

2

**Are Underinvested**

3

**Have Unique Systems Challenges**

4

**New benchmarks and datasets  
are NOW available**

## REFERENCES

- DLRM: <https://ai.facebook.com/blog/dlrm-an-advanced-open-source-deep-learning-recommendation-model/>
- MLPerf DLRM: <https://github.com/mlperf/inference/tree/master/v0.5/recommendation>
- MLPerf Recommendation Benchmark paper: <https://arxiv.org/abs/2003.07336>
- DLRM Paper: <https://arxiv.org/abs/1906.00091>
- DeepRecSys paper: <https://arxiv.org/abs/2001.02772>
- Nvidia DLRM: [https://ngc.nvidia.com/catalog/resources/nvidia:dlrm\\_for\\_pytorch](https://ngc.nvidia.com/catalog/resources/nvidia:dlrm_for_pytorch)
- Intel Optimized DLRM blog: <https://www.intel.com/content/www/us/en/artificial-intelligence/posts/intel-facebook-boost-bfloat16.html>
- Microsoft Recommenders: <https://azure.microsoft.com/en-us/blog/building-recommender-systems-with-azure-machine-learning-service/>
- Nvidia Merlin: [https://ngc.nvidia.com/catalog/resources/nvidia:dlrm\\_for\\_pytorch](https://ngc.nvidia.com/catalog/resources/nvidia:dlrm_for_pytorch)
- PyTorch XLA on Google TPUs: <https://cloud.google.com/tpu/docs/tutorials/pytorch-pod>
- AWS Personalize: <https://aws.amazon.com/personalize/>
- Papers with code: <https://paperswithcode.com/task/recommendation-systems>