Infrastructure for Usable Machine Learning

Matei Zaharia
Stanford DAWN
It’s the Golden Age of ML*

Incredible advances in image recognition, natural language, planning, information retrieval

Society-scale impact: self-driving cars, real-time translation, personalized medicine

*for the best-funded, best-trained engineering teams
Building ML Products is Too Hard

Major successes (e.g., Siri, Alexa, Autopilot) require hundreds to thousands of engineers.

Most effort in data preparation, QA, debugging, productionization: not modeling!

Domain experts can’t easily build ML products.
“Only a fraction of real-world ML systems is composed of ML code”
The Stanford DAWN Project

How can we enable any domain expert to build production-quality ML applications?

- Without a PhD in machine learning
- Without being an expert in systems
- Without understanding the latest hardware

Peter Bailis  Chris Ré  Kunle Olukotun  Matei Zaharia
The DAWN Stack

Data Acquisition
- Snorkel
- DeepDive

Feature Engineering
- MacroBase (Streaming Data)
- Data Fusion

Model Training
- ModelSnap
- AutoRec, SimDex (Recommendation)
- Mulligan (SQL+graph+ML)

Productionizing
- ModelQA
- NoScope (Video)

Interfaces

Hardware
- CPU
- GPU
- FPGA
- Cluster
- Mobile

End-to-End Compilers: Weld, Delite

New Hardware: FuzzyBit, Plasticine CGRA
Training Data is the Key to AI

**Image search, speech, games:** labeled training data is cheap & easy to obtain

**Medicine, document understanding, fraud:** labeled data requires expensive human experts!

How can we leverage data that’s expensive to label at scale?
Snorkel Project (Chris Ré): Labeling Functions, not Labels

1) User writes *labeling functions*: short programs that may not always give right label
   - E.g. regex to search in text

2) Snorkel simultaneously learns *noise* in LFs and a *noise-aware* target model (e.g. LSTM)

<table>
<thead>
<tr>
<th>System</th>
<th>NCBI Disease (F1)</th>
<th>CDR Disease (F1)</th>
<th>CDR Chem. (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaggerOne (Dogan, 2012)*</td>
<td>81.5</td>
<td>79.6</td>
<td>88.4</td>
</tr>
<tr>
<td>Snorkel: Logistic Regression</td>
<td>79.1</td>
<td>79.6</td>
<td>88.4</td>
</tr>
<tr>
<td>Snorkel: LSTM + Embeddings</td>
<td>79.2</td>
<td><strong>80.4</strong></td>
<td>88.2</td>
</tr>
</tbody>
</table>
NoScope: Fast CNN-Based Queries on Video

**Opportunity:** CNNs allow more accurate queries on visual data than ever

**Challenge:** processing 1 video stream in real time requires a $1000 GPU

**Result:** 100-1000x faster with <1% loss in accuracy
Key Idea: Model Specialization

Given a target model and a query, train a much smaller *specialized model*

When this model is unsure, call original

+ Cost-based optimizer to select an efficient model cascade
NoScope Results

40x faster @ 99.9% accuracy
5858x faster @ 96% accuracy

36.5x faster @ 99.9% accuracy
206x faster @ 96% accuracy

VLDB ‘17, github.com/stanford-futuredata/noscope
New Work: Blazelt Query Engine

Accelerates complex, SQL-like queries using model specialization + statistical techniques

SELECT timestamp
FROM taipei
GROUP BY timestamp
HAVING SUM(class='bus') >= 1
    AND SUM(class='car') >= 5
LIMIT 10 GAP 300

https://arxiv.org/abs/1805.01046
The DAWN Stack

Data Acquisition
- Snorkel
- DeepDive

Feature Engineering
- MacroBase (Anomaly Detection)
- Data Fusion

Model Training
- ModelSnap
- NoScope (Inference)
- AutoRec, SimDex (Recommendation)
- Mulligan (SQL+graph+ML)

Productionizing
- ModelQA

Hardware
- End-to-End Compilers: Weld, Delite
- New Hardware: FuzzyBit, Plasticine CGRA

Interfaces
- Systems
- Algorithms

Platforms
- CPU
- GPU
- FPGA
- Cluster
- Mobile
...
Composition in Data Apps

ML app developers compose functions from dozens of high-level libraries

- Python packages, Spark packages, R, …
The Problem

Even if each individual function is well-optimized, the combined app may be highly inefficient

Traditional way to compose libraries: function calls that exchange data via buffers in memory

```
data = pandas.parse_csv(string)
filtered = pandas.dropna(data)
avg = numpy.mean(filtered)
```

5-30x overheads in NumPy, Pandas, TensorFlow, etc
Weld’s Approach

SQL  machine learning  graph algorithms

Common IR

CPU  GPU
Results: Individual Libraries

Porting ~10 common functions per library
Results: Cross-Library

Pandas + NumPy Pipeline

- Current: 9x
- Weld, no CLO: 30x
- Weld, CLO: 30x
- Weld, 12 core: 240x

CLO = cross-library optimization
Running Time [sec; log10]

CIDR ’17, VLDB ‘18, https://weld.rs
“Weld without Weld”: Splittability Annotations

Data movement optimization and auto parallelization for **unmodified, black-box functions**

```c
#include <splittable.h>

void vdAdd(vec_t *a,
            vec_t *b,
            vec_t *res);
```

**S**: “split arrays the same way”

Competition performance to Weld without rewriting libraries!
Machine Learning at Industrial Scale: ML Platforms
ML at Industrial Scale: ML Platforms

If you believe ML will be a key part of future products, what should be the development process for it?

Today, ML development is ad-hoc:
• Hard to track experiments: every data scientist has their own way
• Hard to reproduce results: won’t happen by default
• Difficult to share & manage models

Need the equivalent of software dev platforms
ML Platforms

A new class of systems to manage the ML lifecycle

Pioneered by company-specific platforms: Facebook FBLearner, Uber Michelangelo, Google TFX, etc

+ Standardize the data prep / training / deploy loop:
  if you work with the platform, you get these!

– Limited to a few algorithms or frameworks
– Tied to one company’s infrastructure
Databricks MLflow

Open source, open-interface ML platform (mlflow.org)

**Projects:** package code & data for reproducible runs

**Experiment tracking:** record code, params & metrics via a REST API

**MLflow models:** package models as functions to deploy to backends
Databricks MLflow

Open source, open-interface ML platform (mlflow.org)

**Projects:** package code & data for reproducible runs

**Experiment tracking:** record code, params & metrics via a REST API

**MLflow models:** package models as functions to deploy to backends

Many open questions left in designing such platforms!
Databricks MLflow

Open source, open-interface ML platform (mlflow.org)
Databricks MLflow

Open source, open-interface ML platform ([mlflow.org](http://mlflow.org))

Many open questions left in designing such platforms!
Conclusion

The limiting factors for ML adoption are in dev and productionization tools, not training algorithms.

Many of these are still very unexplored in research!

Follow DAWN for our research in this area: dawn.cs.stanford.edu
Thank you!