DAWN: Infrastructure for Usable Machine Learning

Peter Bailis, Kunle Olukotun, Chris Ré, Matei Zaharia
It’s the Golden Age of Data*

Incredible advances in image recognition, natural language processing, planning, info retrieval

Society-scale impact: autonomous vehicles, personalized medicine, human trafficking

No end in sight for advances in ML

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

Major successes (e.g., AlphaGo, ImageNet) require hundreds to thousands of engineers.

Huge effort in data preparation, model tuning, experimentation, and productionizing.

*Domain experts cannot easily or cheaply build ML products.*
“Only a fraction of real-world ML systems is composed of ML code”
The DAWN Question

What if *anyone* with domain expertise could build their own production-quality ML products?

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

It’s happened before
It’s happened before: Search

**Before:** Decades of research on information retrieval, indexes, ranking, etc

**After:** any developer can add search to an application by linking a library (e.g. Solr, Lucene); everyone (i.e., non-expert users) uses search
It’s happened before: SQL

**Before:** raw access to disk, manual layout of records, network databases (CODASYL)

**After:** SQL forms basis for transactional engines, data warehousing, business intelligence tools

**Key idea:** end-to-end systems that tackle the barriers to access & production use
The DAWN Stack

Data Acquisition
- Snorkel
- DeepDive

Feature Engineering
- MacroBase (Streaming Data)

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

Productionizing
- ModelQA

Hardware
- CPU
- GPU
- FPGA
- Cluster
- Mobile

New Hardware:
- FuzzyBit
- Plasticine CGRA

End-to-End Compilers: Weld, Delite
Example: MacroBase for Continuous Analytics

End-to-end system to **prioritize user attention**

multi-dimensional data streams → MacroBase → anomalies & explanations
<table>
<thead>
<tr>
<th>record_id</th>
<th>user_id</th>
<th>state</th>
<th>hw_make</th>
<th>hw_model</th>
<th>firmware_version</th>
<th>app_version</th>
<th>avg_temp</th>
<th>battery_drain</th>
<th>trip_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>131920</td>
<td>49e36c5b031141dd8cf240f7</td>
<td>CO</td>
<td>Lenovo</td>
<td>Lenovo_K910L</td>
<td>4.4.2</td>
<td>v21</td>
<td>79.252124</td>
<td>0.205834</td>
<td>40.910145</td>
</tr>
<tr>
<td>131921</td>
<td>a670eab2bc6d4e5991ea4269</td>
<td>WV</td>
<td>TCT (Alcatel)</td>
<td>4009A</td>
<td>7.1.1</td>
<td>v36</td>
<td>72.136380</td>
<td>0.184874</td>
<td>47.253076</td>
</tr>
<tr>
<td>131922</td>
<td>247c64e48a8743829c5f7199</td>
<td>UT</td>
<td>TCT (Alcatel)</td>
<td>4009A</td>
<td>7.1.1</td>
<td>v31</td>
<td>77.300103</td>
<td>0.230015</td>
<td>25.342140</td>
</tr>
<tr>
<td>131924</td>
<td>6bd9af7242ca480a96d75d0d</td>
<td>OH</td>
<td>HTC</td>
<td>HTC_M10u</td>
<td>6.0.1</td>
<td>v38</td>
<td>70.937014</td>
<td>0.454293</td>
<td>38.661161</td>
</tr>
<tr>
<td>131926</td>
<td>d449b12dcb6346d7af1021de</td>
<td>HI</td>
<td>HTC</td>
<td>HTC_Wildfire_S_A510b</td>
<td>6.0</td>
<td>v46</td>
<td>75.436764</td>
<td>0.151338</td>
<td>17.785555</td>
</tr>
<tr>
<td>131927</td>
<td>fff8907a14e4a50ab76bd46</td>
<td>HI</td>
<td>bq</td>
<td>Aquaris_E4.5</td>
<td>4.4.1</td>
<td>v38</td>
<td>70.208187</td>
<td>0.286005</td>
<td>60.443799</td>
</tr>
<tr>
<td>131929</td>
<td>8226cd65bb1f4d61a66cf4555</td>
<td>MI</td>
<td>TCT (Alcatel)</td>
<td>ALCATEL_one_touch_97</td>
<td>6.0.1</td>
<td>v35</td>
<td>73.113370</td>
<td>0.249834</td>
<td>16.881133</td>
</tr>
<tr>
<td>131930</td>
<td>30e726fadec6744b2ace2d76b</td>
<td>LA</td>
<td>TCT (Alcatel)</td>
<td>ALCATEL_ONE_TOUCH_60</td>
<td>5.0</td>
<td>v40</td>
<td>77.918077</td>
<td>0.405417</td>
<td>51.163642</td>
</tr>
<tr>
<td>131931</td>
<td>569f35993da246f4af83c2e</td>
<td>FL</td>
<td>Lava</td>
<td>S1</td>
<td>6.0.1</td>
<td>v44</td>
<td>76.558080</td>
<td>0.416760</td>
<td>42.252460</td>
</tr>
<tr>
<td>131932</td>
<td>9d2db241316c43378b8ec14c</td>
<td>AL</td>
<td>LGE</td>
<td>LG-D724</td>
<td>7.0</td>
<td>v29</td>
<td>76.760340</td>
<td>0.334446</td>
<td>37.922632</td>
</tr>
<tr>
<td>131933</td>
<td>4841c0da64e4648878461c</td>
<td>LA</td>
<td>Hisense</td>
<td>LED42K680X3DU</td>
<td>4.4.4</td>
<td>v49</td>
<td>77.138769</td>
<td>0.409485</td>
<td>23.345804</td>
</tr>
<tr>
<td>131934</td>
<td>d375d5a0e10d46cf9b91e343</td>
<td>MI</td>
<td>Techno</td>
<td>TECNO_P5S</td>
<td>6.0</td>
<td>v31</td>
<td>70.115019</td>
<td>0.179464</td>
<td>45.051123</td>
</tr>
<tr>
<td>131936</td>
<td>e4835a64d96e4e89997ce027</td>
<td>WI</td>
<td>ZTE</td>
<td>Z828</td>
<td>6.0.1</td>
<td>v35</td>
<td>71.615570</td>
<td>0.396389</td>
<td>47.662474</td>
</tr>
<tr>
<td>131937</td>
<td>cf00ae2105bb4e3c43b4364b2</td>
<td>FL</td>
<td>Spice</td>
<td>Spice_Mi-498H</td>
<td>5.0</td>
<td>v42</td>
<td>72.045184</td>
<td>0.327405</td>
<td>45.099422</td>
</tr>
<tr>
<td>131939</td>
<td>c94d264a846149f0f851c28e</td>
<td>RI</td>
<td>Infocus</td>
<td>InFocus_M320u</td>
<td>4.4.1</td>
<td>v49</td>
<td>73.543359</td>
<td>0.224504</td>
<td>19.069803</td>
</tr>
<tr>
<td>131940</td>
<td>c3c829d7ab5a4d09b52afe21</td>
<td>MI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131943</td>
<td>4e4566143b144be1809ad4d9</td>
<td>RI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131944</td>
<td>00e8ff83606b496392bedd49</td>
<td>NE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131946</td>
<td>00e8ff83606b496392bedd49</td>
<td>OK</td>
<td>LGE</td>
<td>LS670</td>
<td>4.0.4</td>
<td>v30</td>
<td>78.186519</td>
<td>0.381604</td>
<td>27.601968</td>
</tr>
<tr>
<td>131947</td>
<td>7a4bc456a54d20b1119d42</td>
<td>NV</td>
<td>Oppo</td>
<td>F1f</td>
<td>4.3.1</td>
<td>v38</td>
<td>77.434095</td>
<td>0.436364</td>
<td>40.869723</td>
</tr>
<tr>
<td>131948</td>
<td>c3c829d7ab5a4d09b52afe21</td>
<td>GA</td>
<td>Huawei</td>
<td>HUAWEI_Y320-U151</td>
<td>4.4.3</td>
<td>v42</td>
<td>77.715329</td>
<td>0.281726</td>
<td>23.077248</td>
</tr>
<tr>
<td>131949</td>
<td>2b0ac33a91f49bb6aa70cbe5</td>
<td>MO</td>
<td>ZTE</td>
<td>KPN_Smart_300</td>
<td>4.4.4</td>
<td>v39</td>
<td>75.368614</td>
<td>0.371224</td>
<td>44.295975</td>
</tr>
<tr>
<td>131950</td>
<td>5aab0148ea794d2eeacfc9a27</td>
<td>NV</td>
<td>Ketablet</td>
<td>TR10CS1</td>
<td>5.0</td>
<td>v49</td>
<td>79.459844</td>
<td>0.491424</td>
<td>37.653744</td>
</tr>
</tbody>
</table>

Too much data for manual inspection
Even harder when data is streaming
Database Configuration

Database URL: postgres
Base query: SELECT * from sensor_data;
Connected to postgres database!

Schema Information and Selection

<table>
<thead>
<tr>
<th>Clustering Attribute?</th>
<th>Target Metric? Lo/Hi</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td>reading_id</td>
<td>int8</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>device_id</td>
<td>int8</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>state</td>
<td>varchar</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>model</td>
<td>varchar</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>firmware_version</td>
<td>varchar</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>temperature</td>
<td>numeric</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>power_drain</td>
<td>numeric</td>
</tr>
<tr>
<td>Column</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>app_version</td>
<td>v50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hw_model</td>
<td>em_i8180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hw_make</td>
<td>Emdoor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Support: 0.8226
Ratio Out/In: 472.92
Records: 849

**Correlated attributes**

**Key metric**

![Histogram of battery drain](image)
MacroBase Query Architecture

- Extract domain-specific signals
- Identify data in tails
- Find disproportionately correlated attributes

- Outliers: {iPhone6, Canada}, {iPhone6, USA}, {iPhone5, Canada}
- Inliers: {iPhone6, USA}, {iPhone6, USA}, {iPhone5, USA}
MacroBase Summary

End-to-end system to prioritize user attention

• No ML expertise needed: MacroBase uses general models and tunes them automatically
• No separate step for production use
• Co-design from algorithms to HW

Early users: automotive, cloud, mobile apps, manufacturing

Open source: github.com/stanford-futuredata/macrobase
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)

**Hardware**
- CPU
- GPU
- FPGA
- Cluster
- Mobile

**New Hardware**
- FuzzyBit
- Plasticine CGRA

**End-to-End Compilers**
- Weld, Delite
Weld: Rethinking the Interface to Data Analytics Libraries

Standard approach: users combine libraries using function calls that pass data via memory

Problem: for data-intensive apps, data movement cost dominates on modern hardware!

5-30x slowdowns in NumPy, Spark, TensorFlow, …
Weld’s Approach

Diverse Analytics Tasks

- SQL
- machine learning
- graph algorithms

Common Runtime

Weld IR

Diverse Hardware Platforms

- CPUs
- GPUs
- FPGAs
...
Results: Existing Frameworks

Integration effort: 500 lines glue, 30 lines/operator
Results: Cross-Library Optimization

Pandas + NumPy

- Current
- Weld, no CLO
- Weld, CLO
- Weld, 12 core

Spark SQL UDF

- Scala UDF
- Weld

CLO = cross-library optimization

Open source: weld.stanford.edu
The DAWN Stack

Data Acquisition
- Snorkel
- DeepDive

Feature Engineering
- MacroBase (Streaming Data)
- Data Fusion

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

Productionizing
- ModelQA

hardware
- CPU
- GPU
- FPGA
- Cluster
- Mobile

New Hardware: FuzzyBit, Plasticine CGRA

End-to-End Compilers: Weld, Delite
NoScope: Fast CNN-Based Video Queries

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

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

Result: same accuracy but 100-3000x faster through:
  • Scene-specific distillation
  • Temporal + spatial locality

bit.ly/NoScopeArxiv
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)

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

Interfaces
- CPU
- GPU
- FPGA
- Cluster
- Mobile

Algorithms
- Systems
- Hardware

Software
- Interfaces
- Algorithms
- Systems
Training data is key enabler, barrier to entry

How can we leverage data that’s expensive to label at scale?
Snorkel’s Approach: Weak Supervision

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><strong>81.5</strong></td>
<td>79.6</td>
<td><strong>88.4</strong></td>
</tr>
<tr>
<td>Snorkel: Logistic Regression</td>
<td>79.1</td>
<td>79.6</td>
<td><strong>88.4</strong></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>

[github.com/HazyResearch/snorkel](github.com/HazyResearch/snorkel)
DAWN: machine learning for everyone via novel techniques and interfaces that span hardware, systems, and algorithms

Find out more at dawn.cs.stanford.edu

Peter Bailis  Chris Ré  Kunle Olukotun  Matei Zaharia