AFRL-RI-RS-TR-2020-155

DOING MORE WITH LESS: ACCELERATING THE ANALYST FROM MODELING DOWN TO THE HARDWARE

STANFORD UNIVERSITY

AUGUST 2020

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2020-155 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
EDWARD VERENICH
Work Unit Manager

/ S /
JULIE BRICHACEK
Chief, Information systems
Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUGUST 2020	FINAL TECHNICAL REPORT	MAR 2017-FEB 2020

4. TITLE AND SUBTITLE

DOING MORE WITH LESS: ACCELERATING THE ANALYST FROM MODELING DOWN TO THE HARDWARE

5a. CONTRACT NUMBER
FA8750-17-2-0095

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Oyekunle A. Olukotun

5d. PROJECT NUMBER
D3MP

5e. TASK NUMBER
S0

5f. WORK UNIT NUMBER
05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Stanford University
450 Jane Stanford Way
Stanford CA 94305

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2020-155

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of our project is to build systems that make it easier than before for analysts to extract information from a wide variety of data sources and then use this information for analytical tasks, including prediction, regression, and ever sophisticated modeling. We have developed two systems. A TA2 system for improving model search by using developer exhaust and a TA3 system called snorkel for generating training sets using weal supervision. The snorkel system has been spectacularly successful, makes building high quality ML models much easier for a variety of problem domains. Snorkel has been used extensively by academia, government and industry.

15. SUBJECT TERMS

Model search , training set generation, data programming, weak supervision

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**EDWARD VERENICH**
U	U	U	UU	23	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

1 SUMMARY

The goal of our project (Doing More With Less: Accelerating the Analyst from Modeling Down to the Hardware) is to build a system that makes it easier than ever before for analysts to extract information from a wide variety of data sources and then use this information for analytical tasks, including prediction, regression and even sophisticated modeling. To bring this system to life, progress is required on three fronts: the ability to easily extract signals from unstructured information, the ability to construct high quality predictive models, and to do all three at scale. Thus, we propose to attack all three technical areas. Our team consists of four PIs: Peter Bailis, Kunle Olukotun, Chris Re, and Matei Zaharia who are experts in data analysis, optimization, machine learning, large-scale data systems, and hardware. We have developed two systems. A TA2 system for improving model search by using developer exhaust and a TA3 system called Snorkel for generating training sets using weak supervision. The Snorkel system has been spectacularly successful because it makes building high quality ML models much easier for a variety of problem domains. Snorkel has been used extensively by academia, government and industry.

2 INTRODUCTION

A salient feature of automatic featurization methods, including deep learning, is that these methods require large training sets, which are expensive to create. We have developed a new method of generating large training sets that we call data programming. The key idea is to view an analyst's actions in modeling the problem as implicitly specifying a generative model. Essentially, the analyst writes scripts using standard tools (e.g., Python notebooks) that label data. This creates a large - but noisy - training set. We denoise the training data using this implicit generative model, and then feed the result to automatic feature engineering methods. Thus, the analyst uses only standard, familiar tools, but leverages state of the art approaches under the covers to increase system recall and accuracy. In preliminary work, our denoising techniques result in up to a 6-point gain in F1 score on the TAC KBP benchmark over conventional deep learning approaches. Our initial hack-a-thons suggest that our approach is usable by users with a broad background in domains including law enforcement, pharmacogenomics, and clinical genomics.

In this report, we present our accomplishments for the TA2 and TA3 portions of D3M. Our goal is to enable users to more easily select the right models for solving their problems and to make the building large training sets for training these models much simpler.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 TA2: Model Search with Developer Exhaust

In a typical ML system, the model designers design the model, evaluate the performance and iteratively improve the model. This iterative development process generates a lot of by-products, which we refer to as developer exhaust. Such exhaust can include logs containing model performance info, model configuration files recording the hyperparameters and code snippets defining task pipelines. These by-products do not define the model or algorithm to be deployed, but they can be extremely useful. We argue these seemingly useless exhaust can help complex tasks in ML systems. For example, code snippets can support the development of auto-code generation for pipelines on a specific task; and model performance information can help with model architecture search. Model search is one of the most computationally expensive tasks in ML systems involving exponentially large search space. It aims to discover model with good performance in a specific model space. In this project we focus on accelerating model architecture search with historical developer exhaust (see Fig. 1).

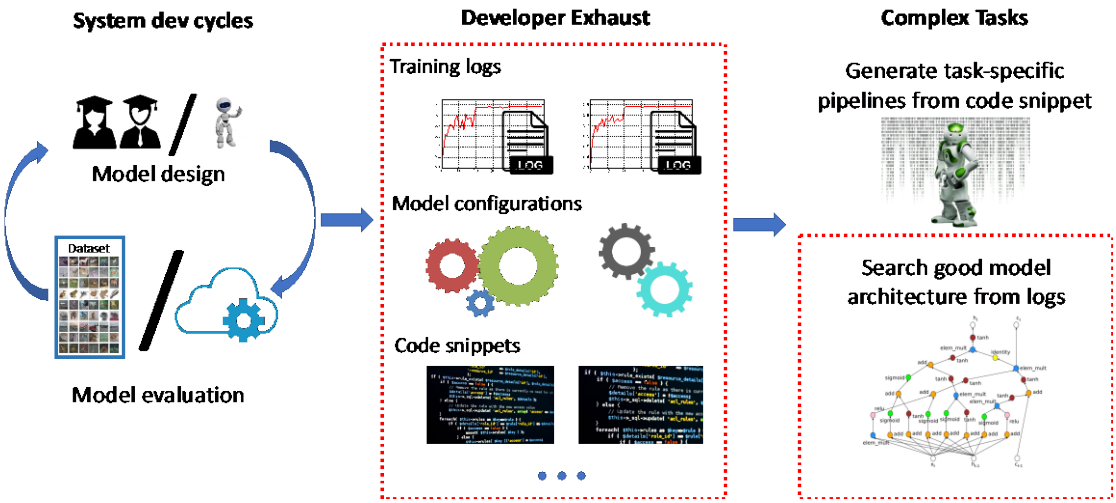
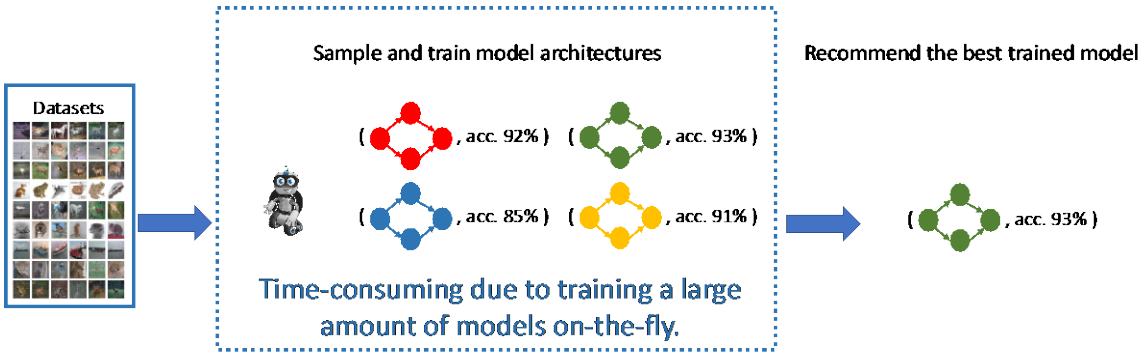


Figure 1. Using developer exhaust in ML systems.

A conventional model search system sequentially samples many promising model architectures in smart ways. Each of these sampled model needs to be fully trained to evaluate its performance. Fully training every sampled model makes model search very computationally intensive and time-consuming. For example, two representative approaches using reinforcement learning and evolutionary strategy can cost hundreds of hours on hundreds of GPUs (see Fig. 2). Given the huge computational cost of current approaches, we ask the question can we use ML exhaust to accelerate the model search process, by avoiding the need to train models on the fly?

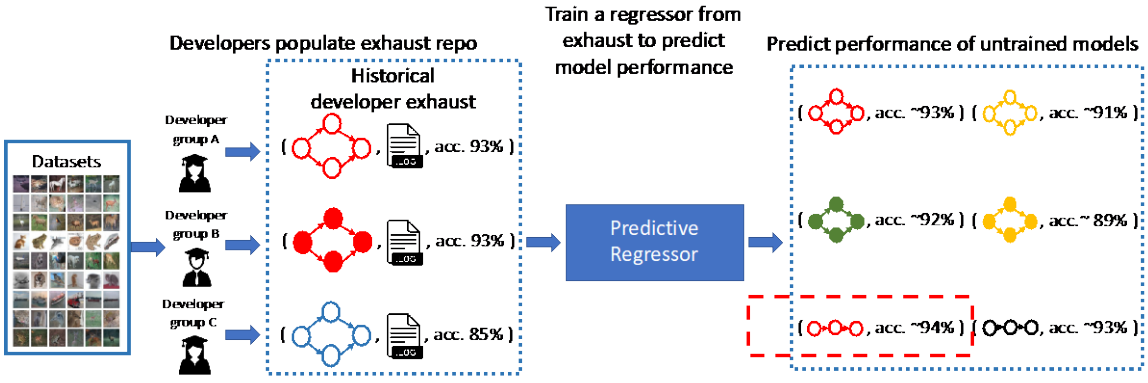


Algorithm	# of machines	Search duration
Neural Architecture Search (Zoph et al.)	450 GPUs	3-4 days
Large Scale Evolution (Real et al.)	250 GPUs	10 days

Figure 2. Conventional model architecture search.

To answer this question, we propose a new model search paradigm based on repositories of developer exhaust. During the historical development of models on similar tasks and similar model spaces, different developers contributes exhausts to a shared repository. This exhaust can provide useful information to model architecture search. For example, the model architectures configurations and its performance in logs. Using this useful information, we train a single regression model before-hand to predict the performance of untrained model architectures in the specific model space. During the actual model search process, we do not need to train each sampled model on-the-fly. Instead we use the trained regressor to predict the performance of many models within fractions of a second. In this way we avoid training models on-the-fly and significantly accelerate the model search process.

Some preliminary results are shown in Figure 3. We conduct experiment on model search for image recognition on CIFAR10. In this experiment, we designed two regressors, one nearest neighbor model based on manual designed distance metrics. To avoid manual featurization, we also designed an LSTM based model. In the comparison to two heuristic baselines, both the nearest neighbor and LSTM regressor can achieve 2.5% less absolute error in predicting test accuracy of untrained models.



Preliminary experiment results on ResNet-style CNNs for CIFAR10	Error in predicting test acc. of models
Baseline 1: predict with the perf. of a random model in the exhaust repo.	4.91%±1.06%
Baseline 2: predict with the perf. of the model with closest # of layer in the repo.	4.13%±1.25%
Nearest neighbor regressor	1.78%±0.33%
LSTM regressor	**1.54%±0.43%**

Figure 3. Accelerated architecture search with developer exhaust.

3.2 TA3: The Snorkel System for Weak Supervision with Labeling Functions

The basic goal of the Snorkel system is to allow users to bring all possible sources of supervision to bear to allow us to program machine learning systems in a radically faster and easier way. In text problems, for which Snorkel was originally designed, we would like to be able to take information contained in such sources as natural language, external knowledge bases, patterns and dictionaries, and in the practices of expert developers, and rapidly combine that information to create training data that can be used to train an end machine learning model for tasks ranging from document classification to relation extraction and more.

Training Data as the Interface to ML

Today's state-of-the-art machine learning models are more powerful and easy to use than ever before- however, they require massive training datasets. Traditionally, these training datasets require slow and often prohibitively expensive manual labeling by domain experts. Snorkel[1] (Alexander Ratner, Bach, et al. 2017) is a system that instead lets subject matter expert (SME) users programmatically build and manage training datasets, enabling them to rapidly and flexibly build machine learning applications that easily plug into and leverage the power of modern infrastructure and model architectures.

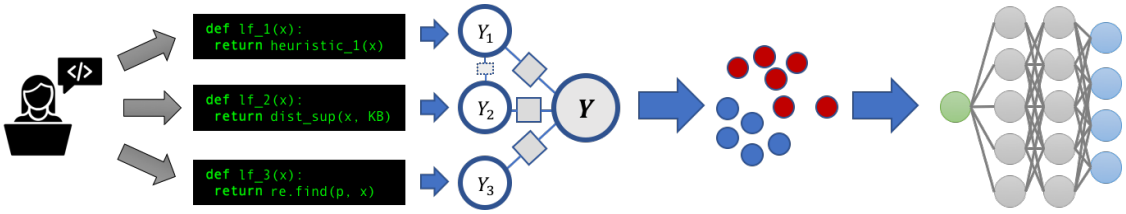
[1] snorkel.stanford.edu

Snorkel Real-World Highlights

This new interface to ML that Snorkel creates can lead to end-to-end application construction and maintenance in a fraction of the time and cost, enabling ML to be applied in scenarios where it was previously infeasible. Snorkel has been used in production applications at places like Google (Bach et al. 2019), IBM (Mallinar et al. 2018), Intel (Bringer et al. 2019), and others; has recently been used to achieve state-of-the-art performance on the GLUE and SuperGLUE language understanding benchmarks (Wang et al. 2019), and has been used extensively in medical settings, highlighted by two recent Nature Communication publications around automated GWAS curation (Kuleshov et al. 2019) and cardiac MRI classification (Fries et al. 2019), in various radiology and neurological monitoring settings where it has been used to replace person months of hand-labeling, and to extract information from electronic health record (EHR) data.

Expanding to New Operators in v0.9

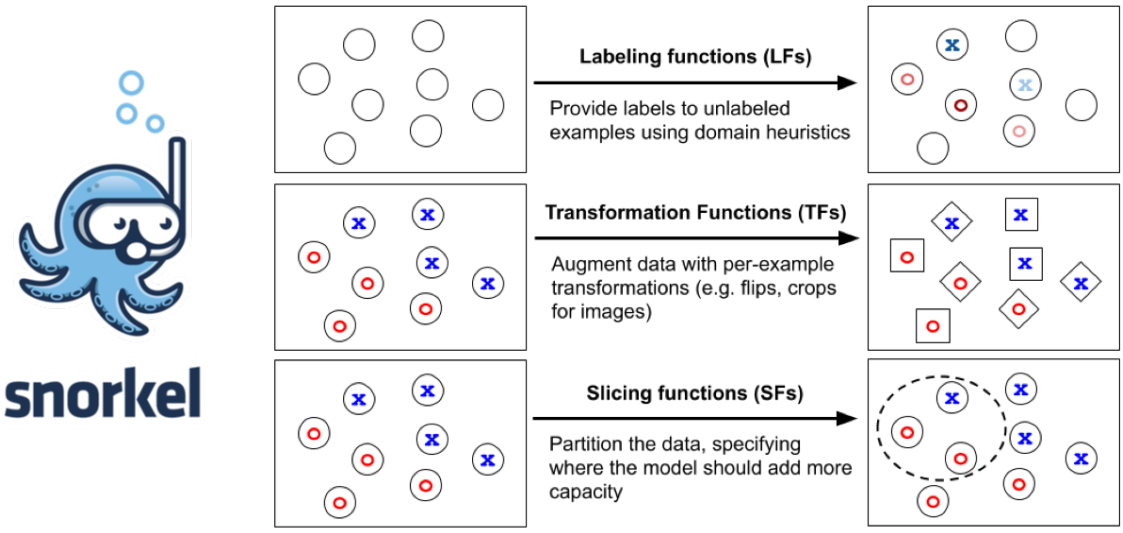
In prior releases of Snorkel, users primarily focused on a single functional operation: writing *labeling functions (LFs)*, black-box functions that labeled data points and could express various heuristics or other programmatic strategies for labeling training data. Snorkel then used a Gibbs-sampling based approach (Ratner et al. 2016) to automatically estimate the LF accuracies, re-weight and combine their noisy outputs, and use the resulting probabilistic training labels for training a user-specified ML model.



In Snorkel users can write labeling functions (LFs) to programmatically label training data, which Snorkel automatically reweights and integrates to produce a clean set of training labels.

Figure 4. The original Snorkel System.

In the new 0.9 version of Snorkel, we expand the scope of Snorkel to support three key operations for building and managing training data, based on our research around Snorkel and weak supervision over the last several years: *labeling data*, for example using heuristic rules or distant supervision techniques; *transforming data*, for example to perform data augmentation and express invariances in the data; and *slicing data* into different critical subsets. In addition, Snorkel v0.9 includes integration of our latest research techniques for modeling and integrating the noisy output of these operators, such as learning policies over transformation functions (A. J. Ratner et al. 2017) and a new, theoretically-grounded matrix completion-style technique for estimating the accuracies and correlations of labeling functions (Alexander Ratner, Hancock, et al. 2019; Varma et al. 2019).



The new version of Snorkel (0.9) supports three core operations for building and managing training datasets programmatically: labeling, transforming, and slicing data. The goal of this workshop was to get feedback on these new operators from real SME users interested in Snorkel.

Figure 5. The New Snorkel (v0.9) System.

In order to assess the accessibility and practical utility of this expanded operator set in advance of the Snorkel v0.9 release, we organized a workshop. Based on our own empirical work (A. J. Ratner et al. 2017; Alexander Ratner, Hancock, et al. 2019), this expanded operator set can lead to higher empirical performance and a greater range of applicable use cases for Snorkel; in the results section we report on the workshop to assess the applicability and accessibility of this expanded operator set for a variety of subject matter expert use cases.

4 RESULTS AND DISCUSSION

4.1 TA2 Results

Our TA2 system for D3M implements a parallelized version of Hyperband (for both model selection and hyperparameter tuning) that is fully compatible with the internal TA1 and TA3 APIs. Our system did quite well in comparison to other teams and that the D3M evaluators were impressed with our results on the seed datasets. We did particularly well on the hidden evaluation datasets, where we passed 44/60 datasets. Our system was one of the only ones that passed the harder evaluation problems and paired successfully with the TA3 systems.

4.2 TA3 Results

Snorkel Real World Impact

Snorkel has had real-world impact in building a machine-curated database of Genome Wide Association Studies, the results were published in Nature Communications. We worked with collaborators in the Genomics department here at Stanford were able to run Snorkel over a large corpus of gene-wide association studies (also known as GWAS).

There is an existing human-curated database called GWAS Catalog that was built over the last 10 years. The Snorkel-powered system, GwasKB, was built from start to finish in about 6 months and identified 60% more relations while maintaining a precision of 0.89. We don't recommend replacing the human curators entirely, but a system like this has a lot of potential to increase their efficiency and potentially help recover additional findings that have been reported but were never recorded.

In collaboration with US Department of Veterans Affairs we have used Snorkel for device surveillance to identify patient outcomes from clinical notes without hand-labeled training data. The Snorkel based system Improved classification performance by 12.8–53.9% over previous rule-based methods and detected over six times as many complication events compared to using structured data alone (see Fig. 6)

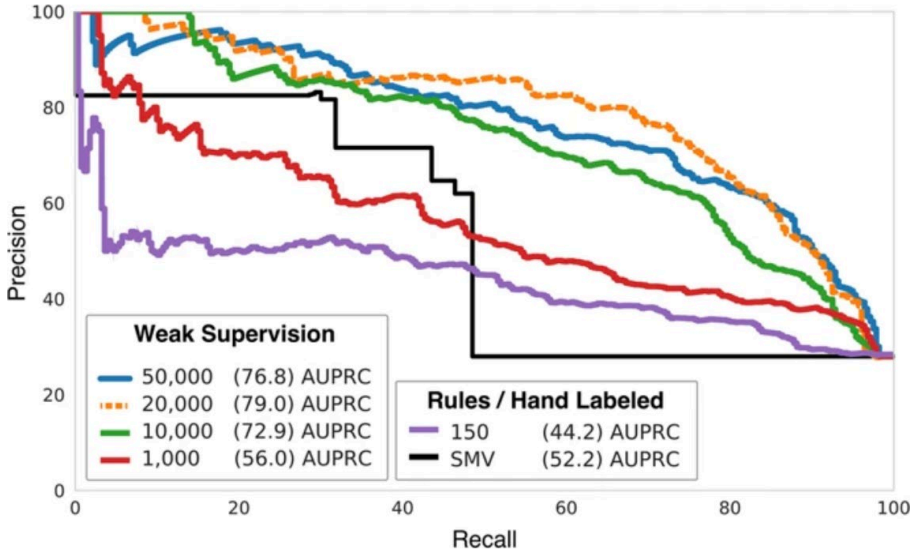


Figure 6. Snorkel detects over six times as many complication events compared to hand labeling.

Snorkel has been used extensively by industry. Snorkel is deployed at Google as Snorkel Drybell. Snorkel Drybell leverages organizational knowledge to improve highly engineered tasks such as topic and product classification by up to 17%. Ideas from Snorkel are used in the Apple Overton system, which was built to support engineers in building, monitoring, and improving production machine learning systems; reduces error rates 1.7-2.9x vs production systems. The Intel Osprey system for enabling SMEs to build models with imbalanced data built using Snorkel; On three relation extraction applications based on real-world deployments at Intel, gains of 18.5 precision points and 28.5 coverage points over prior systems

Snorkel Workshop

In June 2019*, the* Stanford team held a Snorkel workshop at Stanford for the purposes of (i) qualitatively evaluating Snorkel's ability to increase accessibility and efficiency of machine learning (ML) for a diverse range of subject matter experts (SMEs), (ii) collecting feedback on the new features and interface in the open source v0.9 pre-release, and (iii) continuing to build our growing user community.

Snorkel Workshop Details

The workshop was oversubscribed, and finally limited to 55 participants from government, industry, and medicine:

- Government:16
  - Organizations: CIA (6), IQT/Lab41 (3), FBI (2), Other (5)
  - Roles: analyst, data scientist
- Industry: 33
  - Organizations: Google (15), Facebook (4), Northrop Grumman (3), Salesforce (3), VMWare (4), Microsoft (2), Other (2)
  - Roles: software engineer, data scientist
- Medicine: 6
  - Organizations: Stanford (5), VA (1)
  - Roles: physician, postdoctoral researcher

The workshop was structured around the three core operations being brought together in the upcoming 0.9 version of Snorkel[2], representing the important ways that practitioners build and manage training datasets for ML today:

1. **Labeling** training data with labeling functions, e.g. by expressing heuristics or noisy sources of signal programmatically (Alexander Ratner, Hancock, et al. 2019; Alexander Ratner, Bach, et al. 2017).
2. **Transforming** training data with transformation functions, e.g. to perform dataset augmentation by creating transformed copies of data, thereby expressing data invariances in a model agnostic way (A. J. Ratner et al. 2017).
3. **Slicing** training data with slicing functions, in order to demarcate "slices" or subsets of the training dataset where classification is more critical and/or difficult.

These operators are all expressed by writing simple Python functions in Snorkel and are all focused on enabling users to interface with machine learning solely by **building, manipulating, and**

[2] The workshop used a pre-release version located on the redux branch.

managing training data. These key operators bring together several lines of D3M-funded research in this area from Stanford over the last few years, published in leading ML and systems conferences, and now (in the new v0.9 pre-release we used) together in one unified system / open source codebase. In addition, the new pre-release code integrated new algorithmic techniques such as a matrix-completion style approach to cleaning and integrating noisy labels (Alexander Ratner, Hancock, et al. 2019) and learning structure of correlations (Varma et al. 2019).

The workshop was structured around these three basic operators, with labeling functions covered on the first day and transformation and slicing functions covered on the second day. Each day was split up into presentations (both high-level and technical/theoretical deep dives for those interested) and Jupyter notebook-based tutorials in the first part, followed by two hours of open "office hours"-style sessions in the afternoon during which participants could discuss potential use cases of Snorkel with us and each other, attend breakout sessions on specific topics, and explore the tutorial notebooks and ask questions. Subjects of these breakout sessions included Snorkel applications in Natural Language Processing (NLP), images, time series, richly formatted data, and bioinformatics, designing slicing functions and transformation functions, multitask learning in Snorkel, and Snorkel theory. All material for tutorial (Day 1 and Day 2) and slide material used is available online.[3]

Aim (i): Evaluating Accessibility to Diverse SMEs

The first major goal of the workshop was to qualitatively evaluate the ability of diverse SMEs to use the three core operators in the new Snorkel codebase – labeling, augmentation, and slicing – to rapidly train machine learning models for new applications by efficiently building, manipulating, and managing their training data. We assessed this by developing a set of Jupyter notebook tutorials for participants to complete, and evaluating the degree to which users from a wide variety of backgrounds were able to develop these operators on their own in a limited period of time. This evaluation is a natural follow-up to that in our previous Snorkel workshop, in partnership with NIH's Mobilize Center, during which we found that Snorkel users were able to use labeling operators alone to program models in 2.5 hours that performed 45.5% better on average F1 score for a relation extraction task (Alexander Ratner, Bach, et al. 2019).

Labeling Functions

On Day 1 of the workshop, users built labeling functions to provide weak supervision for a simple relation extraction task, used Snorkel to train a generative model to combine those labeling functions, and subsequently trained a simple neural network. This same fundamental process has recently been used to build the largest automated Gene Wide Association Study (GWAS) database (Kuleshov et al. 2019), create models that successfully detect rare cardiac malformations on phase-contrast Magnetic Resonance Imaging (MRI) (Fries et al. 2019), and yield substantial improvements on critical industrial tasks such as topic and event classification (Bach et al. 2019).

[3] Day 1 tutorials: https://t.co/HoPqfBEg9Y. Day 2 tutorials: https://t.co/HoPqfBEg9Y. Slide material: https://t.co/f6XschVjVe.

During the workshop, 100% of individual participants wrote at least one labeling function of their own, assessed its performance on a hand-labeled development set, and trained a generative model using both their own labeling functions and a set provided in the tutorial. Each participant was able to accomplish this process in under 15 minutes. Given these observations, we expect that a diverse set of SMEs at government partners will be able to use labeling functions in Snorkel to rapidly create training sets to support new applications.

Transformation Functions

On Day 2 of the workshop, users extended their interaction with Snorkel to include transformation functions, a new operation for data augmentation. All users were able to successfully complete a set of tutorials focused on building transformation functions for the recent SuperGLUE benchmark, which contains multiple tasks that were identified as being particularly difficult for machine learning algorithms relative to human performance (Wang et al. 2019). To assess users' comfort level with writing augmentation functions, participants were split into five groups and tasked with providing candidate transformation functions for the Word in Context task that forms part of SuperGLUE. In under 15 minutes, all of these groups suggested transformation functions such as random insertion, synonymy replacement, and random word pair swaps that align with commonly used augmentations in natural language processing. Furthermore, 60% of these groups were able to encode one of these transformation functions using Snorkel tools during this same 15-minute period.

Slicing Functions

Finally, users were also introduced to the concept of improving performance on critical data subsets via slicing functions, which identify subsets of the data that could benefit from additional representation. While users did not compose their own slicing functions, all of the participants completed tutorials demonstrating how slicing functions could be used on multiple tasks from the SuperGLUE benchmark. This idea of data slicing is particularly compelling in the context of increasingly common multi-task learning problems, which they are natively supported by the new Snorkel API. In fact, the same techniques and interfaces for slicing and multitask learning that users worked with at the workshop were recently used to achieve a recent state-of-the-art result on the SuperGLUE benchmark.

Given the ease and efficiency with which users were able to interact with our workshop materials, we expect that these new interfaces for slicing functions, transformation functions, and multitask learning will prove useful to SMEs as they apply Snorkel to their own applications.

Aim (ii): User Feedback on Snorkel v0.9

A second major goal of the workshop was to obtain feedback from real users on how Snorkel could be improved for deployment to their specific use cases. We conducted one-on-one meetings with every participating organization and synthesized this feedback to inform our future development efforts. From a software standpoint, several recurring suggestions were:

- **Integrating scalable tools**: Participants were interested in leveraging common tools like Docker, Dask, CoreNLP, IBM GNER, Conda, and others to allow for enterprise-scale deployment.

- **Handling additional input types**: Participants suggested building preprocessors to make it easy to use additional data types (e.g. time series, graphs, etc.) in Snorkel, and allowing users to provide data in NumPy to speed conversion to TensorFlow, MxNet, and other frameworks.

- **Incorporating multi-language support**: Government participants in particular were interested in multi-language support for a variety of workloads.

We also received several frequently asked questions during our user feedback sessions, which are useful for improving both the core system and associated materials:

- **How can users measure distributional shift?** A major concern for users is understanding when the data distribution changes enough to render their trained models ineffective. In Snorkel, this may require writing new LFs/TFs/SFs to rapidly retrain a model.
- **Do we have practical tips for building Snorkel applications?** New users often ask questions including when to stop writing LFs, if they should aim for precision or recall with new LFs, and how to debug the generative model. Understanding additional theory is helpful but building in automated checks to suggest to a user what strategies they might try based on the form of their label matrix would be helpful.

- **How do labeling functions, transformation functions, and slicing functions fit together?** Several practical issues that users encounter could be addressed with tutorials and examples that demonstrate how the theory underlying these techniques translates into practice. For instance, it is not intuitive to first-time users that conflict between the LFs is not only acceptable, but in fact required for any meaningful learning to occur. Similarly, defining appropriate slices is not always easy for first-time users.

Aim (iii): Building an Open Source Snorkel Community

Participants from all backgrounds expressed interest in translating Snorkel directly to their internal use cases on a wide variety of problem sets. Within government, we expect follow-up from users within CIA, IQT, FBI, SOCOM, and USN. Similar enthusiasm for uptake was observed for academic and industry users.

In order to best support this type of translation, we aim to build an open-source software community around the Snorkel project. Encouragingly, many participants from academia, government, and industry expressed interest in contributing back to the open source Snorkel software repository. A couple of highlights are listed below:

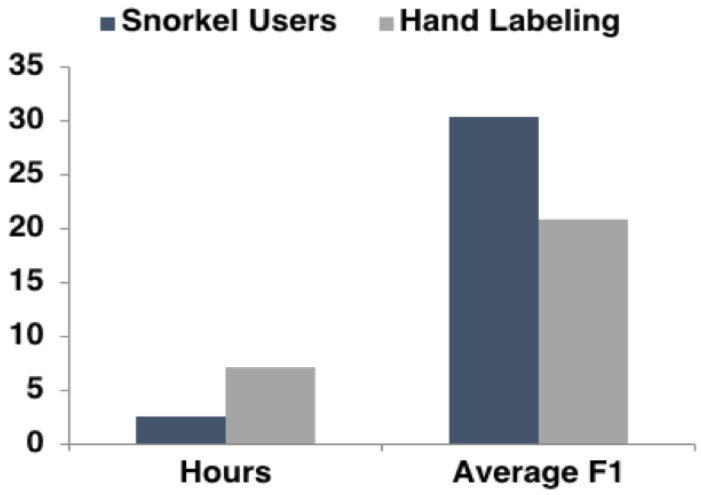
- **Interfaces for writing labeling functions from user behavior**: One participant from industry was keen to write LFs over user customer interaction with a specific product, such that better recommendation engines could be passively supervised. Tools and interfaces for this application could be added to the open-source repository.

- **Enterprise-scale deployment tools**: Industry and government participants were particularly keen to contribute to the development and maintenance of interfaces for deployment tools such as Docker, Spark, Dask, and others, in order to make the code easily extensible to organization-scale workloads.

- **Modality-specific preprocessors**: For handling image, time-series, graph, multi-language, and other data types, users from all backgrounds expressed interest in contributing these types of tools to widen the problem sets to which Snorkel can be easily applied.

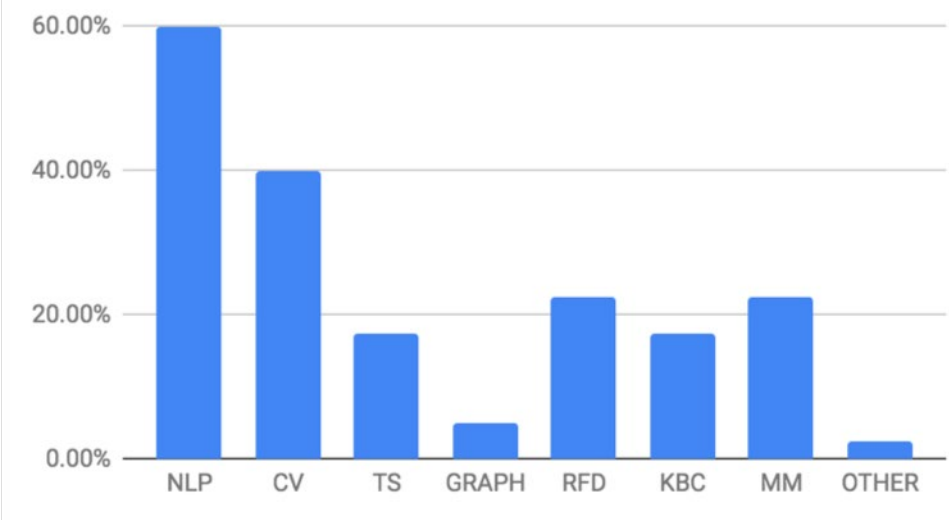
Moving forward, we hope to maintain an active open-source environment for extending and improving the current Snorkel codebase.

Insights from Workshop



A plot of hours and F1 score achieved on a user study task at a previous Snorkel workshop [Ratner et al., 2019a] as compared with an equivalent time spend hand-labeling training data.

Figure 7. Impact of the Snokel system.



The distribution of problem domains of interest to Snorkel users at the latest workshop across natural language processing (NLP), computer vision (CV), time series (TS), graph, richly formatted data (RFD), knowledge base completion (KBC), multi-modal (MM), and other.

Figure 8. Problem types that interest Snorkel Users.

The workshop demonstrated several important points that could be useful for the broader D3M program:

- **Most SMEs are familiar with Jupyter**: None of our workshop participants had any issues with the Jupyter-based interface. This suggests that python-based tools deployed via notebook could be a useful approach for TA3 interfaces.
- **Users are more familiar with labeling and augmenting than with error analysis**: Most users could directly relate to the process of labeling and augmenting their data using labeling and transformation functions. However, users were much less familiar with the idea of performing error analysis and using those results to inform a second stage of modeling. In practice, this is a critical step that should get more attention from machine learning systems intended to assist SMEs in building effective models.
- **Wide range of application interest**: Workshop participants expressed interest in a wide variety of problem sets, which fall into three distinct categories with respect to support in Snorkel and D3M:
  - *Supported by Snorkel and D3M*: NLP, computer vision, object detection, time series.
  - *Supported by D3M, but not yet by Snorkel*: Graph analytics – Snorkel can theoretically support this, but graph-specific tools have not yet been built.
  - *Supported by Snorkel, but not yet by D3M*: richly formatted data applications (PDF, HTML, charts, tables, etc.), knowledge base construction, multi-task/multi-modal problems.

4.3 Discussion

The Snorkel team is responding to recommendations from the participants of the workshop by building tools to support identified use cases, continuing to add educational materials to assist users in getting started, and maintaining an active open-source community to improve the software. In the immediate term, we are focused on the following action items:

- Supporting enterprise-scale deployment tools such as Spark and Docker

- Providing tutorials, examples, and specialized code to handle a wider variety of modalities (images, time series, etc.)

- Promptly engaging users who wish to contribute to the open-source codebase – we will prioritize issues and pull requests from our participants

- Holding a follow up workshop that focuses on data slicing – this concept is less familiar to users, but can be critically important to modern machine learning workflows

- Working with DoD and government users to support their deployment needs – we look forward to further engagement with workshop participants, and welcome direct interaction with D3M participants and transition partners

More broadly, we will continue to gather feedback on our system and ensure that the codebase is as effective as possible in allowing SMEs to rapidly build machine learning models by building, manipulating, and managing their training datasets.

5 CONCLUSIONS

The key result of our project (Doing More with Less: Accelerating the Analyst from Modeling Down to the Hardware) is Snorkel. Snorkel big idea is to build models by programmatically building & modifying the training dataset using three core operations to manipulate training data:

- Labeling (LFs)
- Transforming (TFs)
- Partitioning/ “slicing” (SFs)

- New core modeling techniques and capabilities

Our results show that Snorkel works to reduce the burden of labeling training data and improve the accuracy of the models that are trained. Snorkel has been used by subject matter experts in academia, industry and government to build high quality ML models with less time and effort.

6 PUBLICATIONS SUPPORTED BY D3M

1. Bach, Stephen H, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, et al. 2019. "Snorkel Drybell: A Case Study in Deploying Weak Supervision at Industrial Scale." In *Proceedings of the 2019 International Conference on Management of Data*, 362–75. ACM.

2. Bringer, Eran, Abraham Israeli, Alex Ratner, and Christopher Ré. 2019. "Osprey: Weak Supervision of Imbalanced Extraction Problems Without Code." In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, 4. ACM.

3. Fries, Jason A, Paroma Varma, Vincent S Chen, Ke Xiao, Heliodoro Tejeda, Priyanka Saha, Jared Dunnmon, et al. 2019. "Weakly Supervised Classification of Aortic Valve Malformations Using Unlabeled Cardiac Mri Sequences." *Nature Communications*, In Press.

4. Kuleshov, Volodymyr, Braden Hancock, Alexander Ratner, Christopher Re, Serafim Batzoglou, and Michael P Snyder. 2019. "A Machine-Compiled Database of Genome-Wide Association Studies." *Nature Communications*, In Press.

5. Mallinar, Neil, Abhishek Shah, Rajendra Ugrani, Ayush Gupta, Manikandan Gurusankar, Tin Kam Ho, Q Vera Liao, et al. 2018. "Bootstrapping Conversational Agents with Weak Supervision." *arXiv Preprint arXiv:1812.06176*.

6. Ratner, A., C. De Sa, S. Wu, D. Selsam, and C. Ré. 2016. "Data Programming: Creating Large Training Sets, Quickly." In *Neural Information Processing Systems (NeuRIPS)*.

7. Ratner, Alexander, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. "Snorkel: Rapid Training Data Creation with Weak Supervision." *Proceedings of the VLDB Endowment* 11 (3): 269–82.
8. ———. 2019. "Snorkel: Rapid Training Data Creation with Weak Supervision." *VLDBJ Special Edition*.

9. Ratner, Alexander, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. "Training Complex Models with Multi-Task Weak Supervision." *Proceedings of AAAI*.

10. Ratner, Alexander J, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. “Learning to Compose Domain-Specific Transformations for Data Augmentation.” In *Advances in Neural Information Processing Systems*, 3236–46.

11. Varma, Paroma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Re. 2019. “Learning Dependency Structures for Weak Supervision Models.” In *International Conference on Machine Learning*, 6418–27.

12. Wang, Alex, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. “Superglue: A Stickier Benchmark for General-Purpose Language Understanding Systems.” *arXiv Preprint arXiv:1905.00537*.

13. Zhao, T., Zhang, Y. and Olukotun, K., "Serving Recurrent Neural Networks Efficiently with a Spatial Accelerator". In *Proceedings of the 2 nd SysML Conference*, Palo Alto, CA, USA, 2019.

14. Zhang, Y., Rucker, A., Villim, M., Prabhakar, R. and Olukotun, K., "Scalable Interconnects for Reconfigurable Spatial Architectures". In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019.

15. Swamy, T., Rucker, A., Shahbaz, M., Yadwadkar, N., Zhang, Y. and Olukotun, K., "Taurus: An Intelligent Data Plane". In *P4 Workshop*, 2019.

16. Hadjis, S. and Olukotun, K., "TensorFlow to Cloud FPGAs: Tradeoffs for Accelerating Deep Neural Networks". In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 360-366, Sep. 2019.

17. Rucker, A., Shahbaz, M., Swamy, T. and Olukotun, K., "Elastic RSS: Co-Scheduling Packets and Cores Using Programmable NICs". In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*, pp. 71–77, Beijing, China, 2019.

18. Singhal, R., Zhang, Y., Ullman, D., Prabhakar, R. and Olukotun, K., "Efficient Multiway Hash Join on Reconfigurable Hardware". In *Technology Conference on Performance Evaluation and Benchmarking*, Los Angeles, CA, USA, 2019.

19. Koeplinger, D., Feldman, M., Prabhakar, R., Zhang, Y., Hadjis, S., Fiszel, R., Zhao, T., Nardi, L., Pedram, A., Kozyrakis, C. and Olukotun, K., "Spatial: A Language and Compiler for Application Accelerators". In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 296--311, Philadelphia, PA, USA, 2018.

20. Nardi, L., Koeplinger, D. and Olukotun, K., "Practical Design Space Exploration". In *CoRR*, Vol. abs/1810.05236, 2018.

21. Prabhakar, R., Zhang, Y., Koeplinger, D., Feldman, M., Zhao, T., Hadjis, S., Pedram, A., Kozyrakis, C. and Olukotun, K., "Plasticine: A reconfigurable architecture for parallel patterns". In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 389-402, June 2017.

22. De Sa, C., Feldman, M., Ré, C. and Olukotun, K., "Understanding and optimizing asynchronous low-precision stochastic gradient descent". In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 561-574, June 2017.

23. Koeplinger, D., Prabhakar, R., Zhang, Y., Delimitrou, C., Kozyrakis, C. and Olukotun, K., "Automatic Generation of Efficient Accelerators for Reconfigurable Hardware". In *Proceedings of the 43rd International Symposium on Computer Architecture*, pp. 115--127, Seoul, Republic of Korea, 2016.

24. De Sa, C., Olukotun, K. and Re, C., "Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling". In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pp. 1567--1576, New York, NY, USA, 2016.

7 LIST OF ACRONYMS

API	Application Programming Interface
CIFAR10	Image dataset used for training models
CV	Computer Vision
D3M	Data Driven Discovery of Models, DARPA program
F1	Statistical classification measure
GLUE	Language understanding benchmark
GPU	Graphical Processing Unit
GWAS	Gene Wide Association Study
KBC	Knowledge base competition
LFs	Labeling functions
LTSM	Long Term Short Memory neural network architecture
ML	Machine Learning
MM	Multi Model
MxNet	Neural network training library
MRI	Magnetic resonance imaging

NIH	National Institute of Health
NLP	Natural Language Processing
NumPy	Python numerical computation library
PI	Principal Investigator
RFD	Richly Formatted Data
SME	Subject Matter Expert
SFs	Slicing functions
TA(x)	Technical Area (x)
TensorFlow	Neural network training library
TS	Time Series
TFs	Transformation functions