Scalable End-to-End ML Platforms: from AutoML to Self-serve

Presented by: Mia Garrard

Authored by: Igor L. Markov, Pavlos A. Apostolopoulos, Mia Garrard, Tianyu (Tanya) Qie, Yin Huang, Tanvi Gupta, Anika Li, Cesar Cardoso, George Han, Ryan (Payman) Maghsoudian, and Norm Zhou

arXiv:2110.07554

Meta AI

Agenda

01 Why self-serve?

02 Background & current landscape

03 Example platforms: Looper & PEX (Personalized Experiment)

04 Requirements to achieve “Self-serve”

05 Improvements and Deployment Experience

06 Discussion

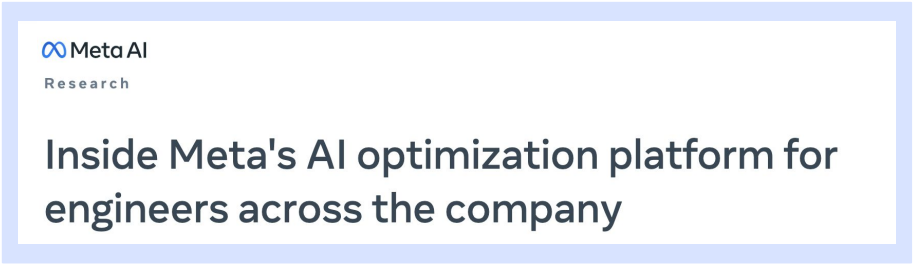
01 Why Self-serve ML Platforms?

Self-serve ML platforms

- Enable **dramatic scaling of adoption** of ML platforms by both subject matter experts and novices

- **Directly drive business value** by expanding the use of ML to previously untapped applications via reuse of ML capabilities and infrastructure with high client productivity

- Development requires:
  - AutoML techniques
  - Platform integration
  - Online testing of models and policies



From April 2022 Meta Blog post on Looper & PEX: ai.meta.com/blog/looper-meta-ai-optimization-platform-for-engineers/

Expanding Impact of ML via Self-serve

- The value proposition of “self-serve” end-to-end ML platforms:
  - **Shared engineering effort (new technologies, regular platform maintenance, and system upgrades) helps customers focus on applications**
  - Supported by AutoML by scaling configuration and optimization

- Integration with related platforms and infrastructure furthers the impact of these platforms, and **platform impact is directly related to economies of scale enabled by the self-serve quality.**

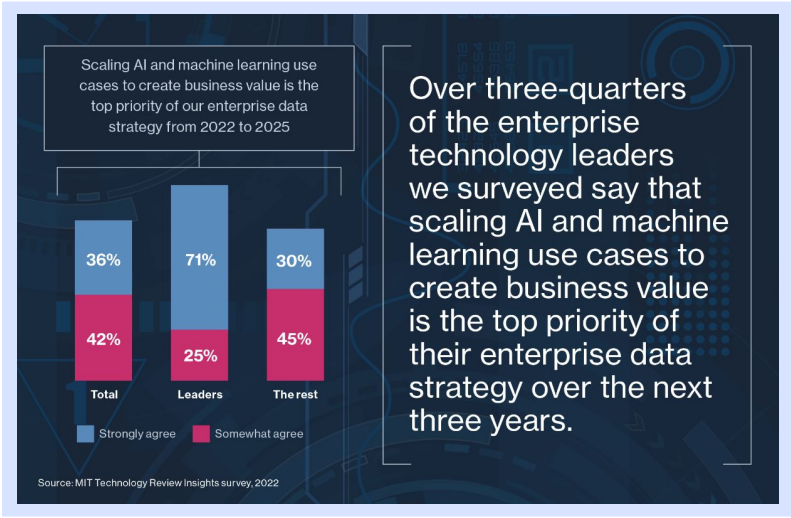
02 Background & Current Landscape

AI and ML usage across industry is pervasive

- AI is leveraged by almost all 600 companies, per CIO Vision 2025 survey conducted by MIT Technology Review
  - an overwhelming majority of those surveyed are not AI driven companies → early stages of adoption

- Common concerns on **how to scale up** the use of AI and **speed up** AI development

- Enablement and scaling AI and ML use cases **across a wide variety of applications** is seen as mission critical by survey respondents



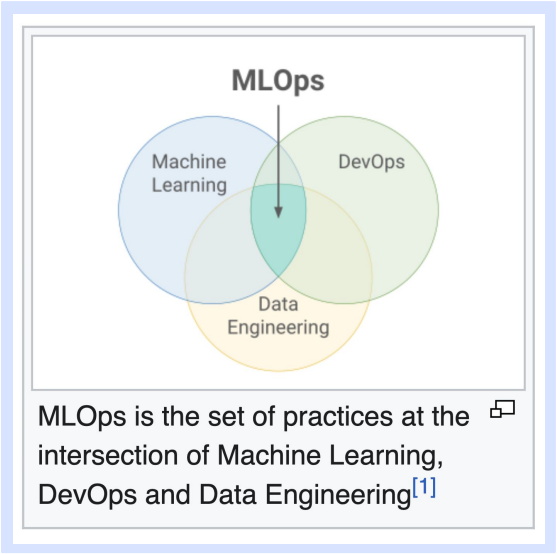
Shifting the focus from Kaggle paradigm → MLOps

kaggle

Source: Wikipedia Kaggle platform entry

- Goal is typically to train a model to **minimize a loss function**
  - Drives ML architecture development, optimization algorithm creation

- **Overlooks implementation & runtime tradeoffs**
  - Model size, inference latency



Source: Wikipedia MLOps entry

MLOps field broadens the evaluation of new ML model architectures and algorithms in the context of implementation and runtime trade-offs

ML Platform Development

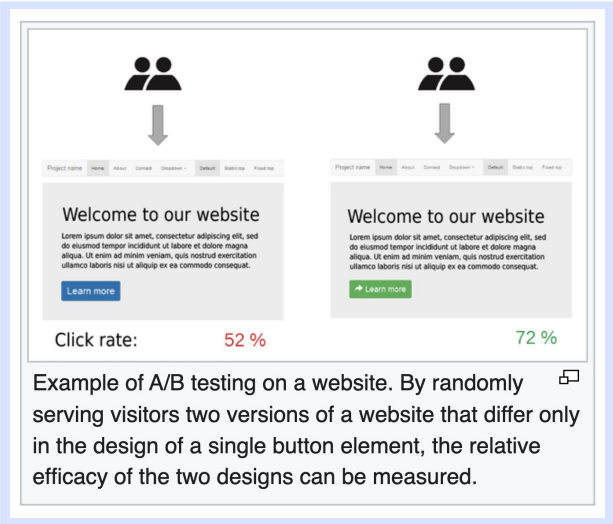
ML platforms often support and automate workflows that train ML models on data to perform prediction, estimation, ranking, selection, and other ML tasks

ML platform development is driven by the following tradeoffs:

(1) Customer needs vs. technology availability,
(2) Expert support, fine configuration, and optimization for high-end applications vs. automation to reduce cognitive load and engineering effort,
(3) System-level development (driven by traditional SW engineering, design and architecture considerations) vs. ML-driven development (initiated and driven by data, model and metric considerations),
(4) Reactive short-term efforts vs. proactive long-term plans,
(5) Orchestrating numerous point tools into entire workflows.

End-to-end ML platforms

- **Support workflows with a broader scope** including data collection and preparation as well as tracking and optimization of product-impact metrics to drive business value
- **Integration with A/B testing** is critical for product metric tracking
- **Automate data collection + model retraining**
- Can be general or specialized



Example of A/B testing on a website. By randomly serving visitors two versions of a website that differ only in the design of a single button element, the relative efficacy of the two designs can be measured.

Source: Wikipedia A/B testing entry

AutoML & E2E ML Platforms

- AutoML frameworks
  - are pervasive across industry
  - provide a consistent interface that supports interchangeability, composition, ML pipeline management, and live-product experimentation

- **Using AutoML frameworks in e2e ML platforms is critical**
  - Help manage ML lifecycle
  - Removing need for ML coding via integration with platform provides AutoML solution

- This is why our overall strategy in this work focuses on pervasive use of AutoML techniques, platform integration, and online testing.

- FLAML (Microsoft, [44]) - a lightweight AutoML library that handles common tasks (model selection, neural architecture search, hyperparameter optimization, model compression),
- Vizier (Google, [41]) - a uniform Python interface for black-box optimization intended for hyperparameter optimization that offers a variety of optimization algorithms,
- Ax (Meta, [3]) – a general ML tool for black-box optimization that allows users to explore large search spaces in a sample-efficient manner for services such as multi-objective neural architecture search, hyperparameter optimization, etc.
- Auto-sklearn[14, 15] - an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator,
- AutoGluon (Amazon, [12]) - an AutoML library that manages a variety of ML models, with provisions for image, text and tabular data, as well as multimodal data,
- LaLe (IBM, [4]) - an AutoML library for semi-automated data science,

03 Example Platforms: Looper & PEX

Usage of Looper & PEX platforms at Meta

- **~100 product teams** leverage for product metric improvement
- **3-4 million AI outputs per second**
- **Hundreds of use cases** with wide variety of applications
- Specialize in tabular data

Looper

- General purpose e2e ML platform
- Supports many ML tasks: classification, regression, ranking, decision making with Contextual Bandits/Reinforcement Learning
- Requires no ML experience
- Typical time to launch: ~1 month (vs. several months for traditional ML development cycle)

Figures from "Looper: an end-to-end ML platform for product decisions"

Looper: a General-purpose Platform

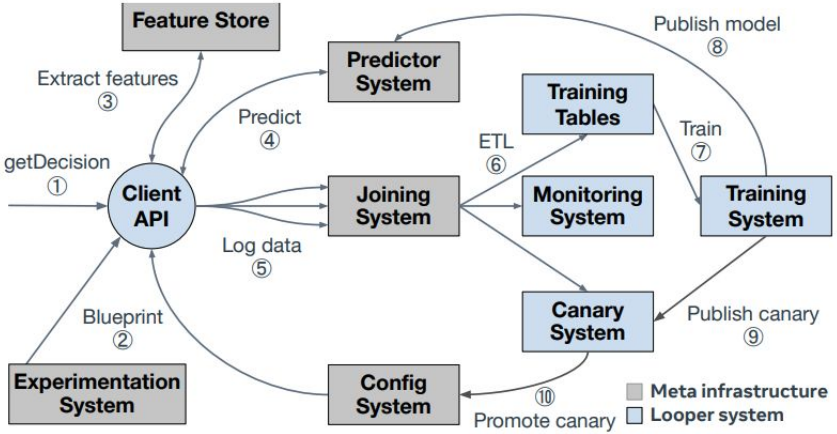


Fig. 2. Data flow in the Looper platform. Figure 3 expands the left side.

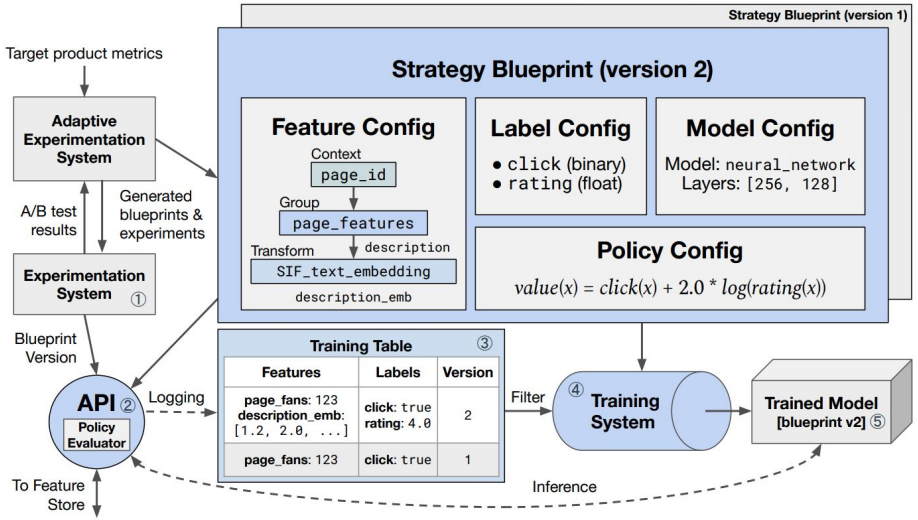


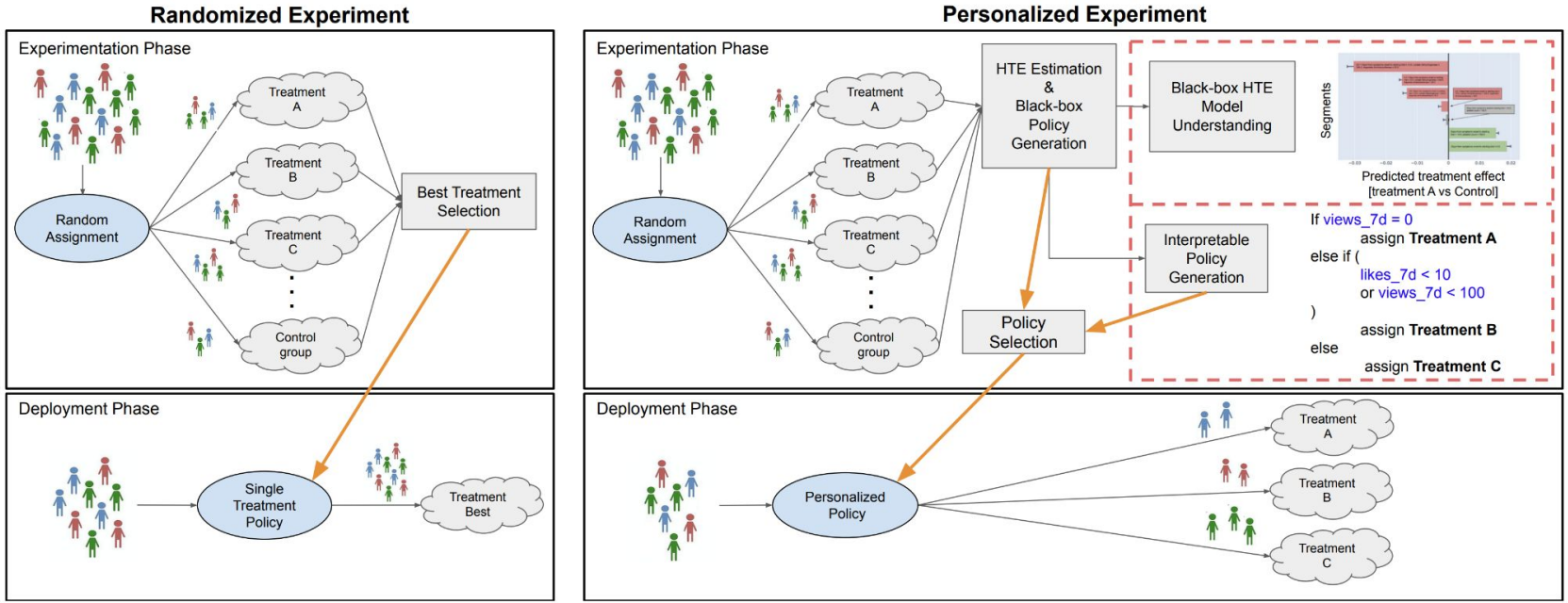
Fig. 3. The strategy blueprint and how it controls different aspects of the end-to-end model lifecycle. Continuation of Figure 2.

Figures from "Looper: an end-to-end ML platform for product decisions"

Personalized Experiment (PEX): a Specialized Platform

- Specialized platform enabling product teams to leverage heterogeneous treatment effects to optimize end-user experience at the individual level
- Optimizes directly for product metrics, as opposed to ML metrics (such as loss function)
- Supports HTE meta-learners and RL models
- Leverages decision policy tuning via offline(counterfactual-based)/ online Bayesian Optimization via the Adaptive Experimentation platform *(open source: Ax)*
- Requires no ML experience

Personalized Experiment (PEX)



Figures from "Interpretable Personalized Experimentation"

Commonalities between Looper & PEX

- **Full custody of data** from automatic data collection to causal model evaluation by means of A/B testing with product metrics
  - Different context and APIs, but result is **simplified maintenance, reduced engineering effort by eliminating known pitfalls with data collection**

- **Customers are not required to define or implement new model types**
  - Select from a variety of model types manually or automatically, with no ML code needed from customers

- Both platforms are **config driven, maintain reproducible ML models, and regularly retrain models to adapt to data drift**
  - Automatic model evaluation and promotion of models

04 Requirements to achieve “Self-serve”

Data & Product Impact in Self-serve

- **Data handling:** platforms should maintain full custody of data throughout the ML lifecycle
  - Prevents potential human errors
  - Saves manual effort of data pipelining and evaluation

- **Product impact evaluation and optimization:** platforms should be able to conduct (1) *observational tasks* and (2) *interventional tasks*
  - **Observational tasks:** evaluate, learn, and model the impact of AI outputs on product end-metrics
  - **Interventional tasks:** optimize prediction mechanisms (models, decision policies) to improve product metrics

- Leveraging counterfactual policy evaluation and online experimentation (A/B Tests) within the platform automates metric related needs

10 Requirements for Self-serve

1. **Low cognitive barrier** to entry and low requirements for ML experience
   a. UI avoids unnecessary dependencies on ML, platform, and data science concepts and explains them via tooltips when unavoidable
   b. “Hide” as many routine tasks behind automation as possible
2. **Automated data collection** from applications and customization of subsequent data preprocessing
   a. Normalization, outlier removal, data imputation, down/up sampling, etc
3. **AutoConf**
   a. Automated selection of ML problem foculation, ML tasks (ranking, classification,etc), model type and default parameters
   b. Followed by workflow automated traditional AutoML (parameter selection, network architecture, search, etc)
4. **Product-impact metrics** – tracking and automatic optimization, support for:
   a. Counterfactual policy evaluation
   b. Online casual evaluation (such as A/B testing)

10 Requirements for Self-serve

5. **Sufficient ML Quality** with limited manual configuration and optimization effort
    a. Comparisons are made to custom AI solutions and AutoML tools/services
6. **Full management of hosting** of data, models, and other components with modest resource utilization
7. **Adaption to data drift *(calibration & retraining)*** to ensure model freshness
8. **Resilience and robustness to disruptions in data and system environment** with minimal recurring customer-side maintenance effort
    a. Includes delayed/missing data, resource limitations/outages in the system, etc
9. **Customer-facing monitoring** and root-causing of customer errors
10. **Scalable internal platform maintenance** and white-glove customer support

Table 1. Applying the notion of self-serve to the Looper and PEX platforms.

10 REQUIREMENTS	LOOPER	PEX
Low cognitive barrier to entry	Broader scope of UI	Specialized UI and data sources
Automated data collection	Generic Looper APIs	Labels from A/B experiments; API to experimentation system
AutoConf	Selection of ML task, model type, features, default params; Hyperparameter tuning Decision-policy tuning for binary classification; Value-model tuning for multiclass classification and multimodel multitasks;	Fixed ML task, selection of HTE meta-learner, RL, or derived heuristic policy, feature selection, offline/online policy optimization and param tuning
Product-impact metrics	Tracking and optimization are critical in most applications	
Sufficient ML quality	Last 1-2% model quality viz. loss functions (SOTA) is not critical in many cases. Overall quality is most affected by the decision policies, and then by the ML model.	Somewhat more important than for Looper. Greatly affected by label selection
Full management of hosting	Clients use their storage quota for data, pipelines, but management is fully automated. This includes automatic canarying and promotion of retrained models.	
Adaptation to data drift	For nonstationary data: automatic model calibration, model retraining and promotion; decision policy re-tuning.	
Resilience, robustness to disruptions	Data distribution monitoring, real-time alerts; handling of missing data	
Client-facing monitoring and root-causing of client errors.	Alerts on anomalies in data and AI outputs. Help diagnose and address client and platform errors.	
Scalable internal platform maintenance	Maintenance load is due to: (i) company-wide system environment changes, (ii) client activity, (iii) use-case issues: quotas, missing data, etc.	

6 Additional Capabilities *(depending on application)*

1. **Open architecture**
   a. Offer platform components and partial workflows individually → servers more advanced customers
2. **Customizations to common ML tasks**
   a. Succinct high-level APIs that use relevant concepts, support relevant model arch, loss functions, regularizations, output constraints, diagnostics, etc (ex: ranking and selection)
3. **Reproducibility of models**
   a. All necessary code and data are available, or comparable data is available
4. **Meta-learning, including transfer learning**
   a. Automatically choosing learning parameters, reusing and adapting trained models to new circumstance
5. **Interpretable ML model**
   a. Provide platform clients insight into model behavior without understanding model internals
6. **Fairness in ML**
   a. Address various ways to evaluation fairness and ways to train ML models to improve those metrics

6 ADDITIONAL CAPABILITIES	LOOPER	PEX
Open architecture	Training models on given data/comparable to Google AutoML tables; Logging as a separate service	N/A
Customizations to ML tasks	Ranking as a service, selection (top N out of a large predefined set), etc	The formal ML task is fixed, but outputs are used for a variety of applications (UI or value model optimizations, etc).
Reproducibility of models	Automatic model refresh (i) to adapt to data drifts, (ii) to meet data retention limits related to data laws.	
Meta- and transfer learning	Transfer-learning across related apps, where some training data can be reused	Meta-learners (T learner, X learner, etc)
Interpretable ML	Feature importance analysis (mostly for model building); Monotone constraint for model output w.r.t. feature values.	In addition to feature importance analysis, automatic user segmentation analysis provided to understand meta-learners; Automated heuristic policy distillation possible
Fairness in ML	Pending guidance for individual applications	

05 Improvements and Deployment Experience

Metric Improvement via automated updates to ML models and Decision Policies

- **Saves ≥ 2 weeks/use case of eng time** to run validation experiments
- Enables **automatic online experiments** for re-tuning of decision policies
- **Increases client trust** via accurate tracking and optimization of their goals
- **Delivers repeated** product metric improvement
  - Example: a use case which determines whether to prefetch certain content *(stories, posts, reels, etc)* to a given device (*smartphone, etc)* launched a newly tuned decision polices which improved top-line product metrics for different device types with overall success of **0.73%** with neutral computational cost

Empirical validation: Client-driven PEX Improvement

- Product team evaluation of **PEX vs Hand-tuning vs ML tool-box and selection of PEX**
  - Interested in *(1)* necessary time investment and *(2)* product metric performance
  - **PEX outperformed the tool** across the time and metric axises
  - PEX performed slightly worse than hand tuning with respect to metric impact, but **significantly better wrt time investment: ~2-3 weeks of active eng time vs 6 months**

- **Customer survey results**
  - 6/6 customers had at least some unfamiliarity with ML concepts
  - 6/6 indicated strong interest in self servability of the platform
  - Results highlighted need for more “complex” decisions to be further automated
    - Drove creation of base set of features automatically included in PEX use cases

06 Discussion

Trade-offs

- Platform engineering team development prioritization necessitates **balance between customization and development of widely applicable features**
  - Some highly specialized but highly impactful applications may require too much customization to benefit from a generalizable self-serve ML platform

- Seasoned ML engineers may benefit from automation of boring or error-prone routines, but will generally have less direct control than they are accustomed to having during development cycle
  - Supports our stance for open-architecture end-to-end ML platforms

Benefits of end-to-end self-serve ML platforms

- Shifts focus from performance of ML model *(ie loss function optimization)* to product metric performance

- Democratizes use of ML techniques beyond ML experts

- Impact grows with economy of scale usage of these platforms

- Wide adoption of platforms enables usage of integrated ML techniques

Scalable End-to-End ML Platforms: from AutoML to Self-Serve