Aleksander Lempinen

MLOps approach for application specific performance tuning for machine learning systems

Master's Thesis in Information Technology

January 10, 2024

University of Jyväskylä

Faculty of Information Technology

Author: Aleksander Lempinen

Contact information: aleksander.lempinen@gmail.com

Supervisor: Tommi Mikkonen

Title: MLOps approach for application specific performance tuning for machine learning systems

Työn nimi: TODO samma på finska

Project: Master's Thesis

Study line: Educational Technology

Page count: 34+0

Abstract: TODO abstract

Keywords: LaTeX, gradu3, Master's Theses, Bachelor's Theses, user's guide

Suomenkielinen tiivistelmä: TODO tiivistelmä suomeksi

Avainsanat: MLOPS, TODO

Glossary

ML	Machine Learning
MLOps	Machine Learning Operations
TODO	TODO

List of Figures

List of Tables

Contents

1 Introduction

Luvun 1 jälkeen otsikon pitäisi olla selkeä

Machine learning (ML) and Artificial Intelligence (AI) have been a hot topic of discussion in the past decade. While there is a mountain of academic research on ML methods and tools, there is a lack of attention paid to practical real-world challenges encountered when developing or running ML systems. DevOps has previously addressed similar challenges in software engineering and a field of MLOps which is DevOps applied to ML has emerged. Machine learning operations (MLOps) focuses on solving challenges related to operating real world machine learning systems (Kreuzberger, Kühl, and Hirschl 2023).

Machine learning (ML) systems are widely adopted and many organizations successfully have ML models running in production. Examples of machine learning systems in different fields include recommender systems , targeted ads (Domingos 2012), drug design (Domingos 2012) or search engines (Domingos 2012).

Find ref

Find ref

Most recent breakthroughs that have generated media attention have been in the fields of computer vision in the form of latent diffusion models (LDM) (Rombach et al. 2022) such as Stable Diffusion (Stability AI 2022) for generating images from prompts and natural language processing in the form of large language models (LLM) (Touvron et al. 2023) such as ChatGPT (OpenAI 2022). There have also been great developments in tooling for machine learning such as Tensorflow, Pytorch or scikit-learn for model development, Ray, Horovod or DeepSpeed for distributed training and MLFlow or Tensorboard for machine learning monitoring.

TF, Pytorch, scikit-learn refs

Ray, Horovod and DeepSpeed refs

MLFlow, Tensorboard refs

Despite wide adoption and many successes, there are still challenges with machine learning systems in practice . The required amount of computation for machine learning has been on the rise. According to an OpenAI technical blog the trend is exponential and more compute leads to better performance (Amodei and Hernandez 2018). Increased compute requirements also mean increased costs such as financial, operational or environmental. Strubell et al. (2020) in their extended abstract bring attention to the environmental impact of training models and in particular hyperparameter tuning, during which costs of training many relatively inexpensive models quickly adds up.

TODO computational budgets, cost, efficiency

Find ref, state of MLOps?

In addition to cost there may be other requirements for machine learning systems. For example machine learning systems on the edge might encounter system requirements such as latency and energy use or have limited resources such as memory or compute (Chen and Ran 2019). Ways of meeting these requirements include hyperparameter tuning, reducing the amount of parameters in the model or model compression such as knowledge distillation (Chen and Ran 2019).

Early stopping has been used as a cost optimization technique to reduce training time by stopping training when performance of the model stops improving on the validation set (Prechelt 1998). More recent work on larger models shows that models might still improve later if training continues for a longer time (Hoffer, Hubara, and Soudry 2018). Using early stopping with other performance metrics such as system metrics has not been as thoroughly studied.

The aim of this thesis is to investigate whether using early stopping with system metrics leads to more efficient hyperparameter tuning when there are resource constraints. Investigation is limited to a small set of widely available machine learning algorithms and datasets that do not require a lot of computation. While more complex and effective hyperparameter optimization methods exist only the simplest are used to simplify the experiments for clarity. The theoretical significance of the thesis is to show that traditional hyperparameter optimization techniques can not only be used on machine learning performance metrics but also alternative metrics such as system metrics. The practical outcomes are reducing costs and allowing for quickly and efficiently tailoring models to fit specific system metric constrains.

This thesis is structured in the following manner: Chapter 2 contains background information about machine learning, DevOps and MLOps and how they relate to each other. Chapter 3 describes the performed experiments and their methods and design including research questions, datasets and algorithms used. Chapter **??** presents the results of the experiments. Chapter 4 revisits the research questions and discusses the interpretation of the results, limitations, related work and future work. Chapter 5 concludes the thesis by summarizing key findings.

2 Machine Learning Operations

Software involving machine learning adds additional complexity to the overall system. Developing, deploying and monitoring machine learning systems involves both traditional software system concepts and some new machine learning specific concepts. Section 2.1 introduces machine learning and evaluating model performance from a practical perspective. Section 2.2 introduces DevOps and performance evaluation. Section 2.3 combines machine learning and DevOps for production machine learning systems and introduces hyperparameter optimization and performance prediction.

2.1 Machine Learning

Real world applications of machine learning are often messy with a large number of decisions for the developer that can result in different behavior of the machine learning model. This section introduces machine learning from a practical standpoint including necessary performance metrics for model training and empirical performance evaluation.

2.1.1 Practical machine learning

Writing programs and developing algorithms to complete specific tasks is a labor intensive task requiring professional programming expertise. A different approach is to develop generic algorithms that can change behavior by learning. The field studying these types of algorithms is called machine learning. Machine learning algorithms learn by applying an optimization algorithm to adjust set of parameters called a model and this process is called training the model (LeCun, Bengio, and Hinton 2015). A simplified machine learning workflow consists of splitting the data into training and test datasets, performing preprocessing separately for each dataset, training the model on the training dataset and finally evaluating the trained model on the test dataset.

TODO: ref

Machine learning is widely used in applications like search, drug design or ad placement and can be also known as data mining or predictive analytics (Domingos 2012). Developing machine learning systems, which are systems that are based on machine learning, can be a difficult task. Unlike traditional software development, experiments with both code and data as inputs are central to machine learning development (Zaharia et al. 2018) and reproducibility of the experiments is often problematic. While plenty of research focuses on machine learning methods or even datasets and data quality, the biggest bottleneck is human cycles (Domingos 2012). Faster iterations improve the machine learning developer or researcher experience. An important metric to pay attention to and optimize is the mean iteration cycle for machine learning developers.

Machine learning can be practiced with two different goals in mind. First is explanatory modeling with the purpose of scientific theory building and testing and the second is predictive modeling mostly used outside of scientific research (Shmueli 2010). One practical difference is that unlike predictive modeling, explanatory modeling rarely uses holdout test sets or cross validation for evaluation (Shmueli 2010). Lack or presence of evaluation on a test set can be used as a heuristic to quickly determine whether a machine learning project is explanatory or predictive in nature. However, even explanatory modeling benefits from evaluating the predictive power (Shmueli 2010). Domingos (2012) in their paper assume all machine learning is predictive in nature and state that machine learning should generalize beyond the training set. It is important to keep in mind the end goals of a machine learning project, because common practices in a research setting might not be applicable when creating machine learning systems.

Machine learning algorithms can be categorized as supervised learning, unsupervised learning or reinforcement learning. The main differences are related to whether the model learns TODO: ref by using "right answers" provided by labeled data in supervised learning, by finding structure in the dataset in unsupervised learning or by interacting with the world in reinforcement learning. Unsupervised learning has the advantage of not requiring labeled data which is an TODO: ref advantage for problems where labels are uncommon (Le et al. 2012).

2.1.2 Model evaluation

Performance evaluation of machine learning models is usually done empirically using cross-validation (Forman and Scholz, no date; Sokolova and Lapalme 2009). Cross-validation involves splitting the data into k-folds and using all but one of the folds for training and the last one for validating the performance of the model after which the procedure is repeated k times with each fold being used for validation (Cawley and Talbot, no date). For example 3-fold validation would use a third of the data for validation and two thirds for training repeated three times. The performance metrics collected during the computationally expensive cross validation are typically averaged (Cawley and Talbot, no date). These types of global averages might not be desirable and instead of random folds the data can be sliced according to some criterion such as by country and allow detecting performance differences between slices (Breck et al. 2017).

Machine learning training involves minimizing an optimization criterion such as log loss, squared hinge loss or Cauchy-Schwarz Divergence (Janocha and Czarnecki 2017). Different loss metrics are chosen depending on the application such as resistance to noisy data or labels (Janocha and Czarnecki 2017). The loss metric is sometimes not informative of model performance such as in classification tasks. In these cases performance metrics such as accuracy, precision, recall, specificity, error-rate , AUC and F-score are used (Sokolova and Lapalme 2009; Forman and Scholz, no date). Metrics such as accuracy are well defined, but the final F-score from cross-validation may be computed in several ways resulting in different results (Forman and Scholz, no date).

Even more informative metrics can be created for specific applications. For example Torrabla and Efros (2011) developed performance metrics to compare different datasets and determine a "market value" for the data by using the generalization performance of machine learning models on the datasets. Defining correctness of the prediction is an important part when defining performance metrics (Lin et al. 2014)

TODO: Define batch, training step, epoch

TODO: Algorithm example of a training loop

2.2 DevOps

DevOps is a well known topic in the field of software engineering that brings together development and operations. This section briefly introduces DevOps and provides and overview to the main benefits related to continuous integration, continuous deployment and continuous performance evaluation. Later it describes the importance of performance metrics with examples and wraps up the section by introducing performance prediction.

2.2.1 Benefits of DevOps

A common interpretation of DevOps is a focus on software quality, collaboration between development and operations, process speed and rapid feedback (Mishra and Otaiwi 2020; Waller, Ehmke, and Hasselbring 2015; Perera, Silva, and Perera 2017). Defining DevOps is difficult as there is no consensus on the exact definition (Smeds, Nybom, and Porres 2015; Mishra and Otaiwi 2020). DevOps can be viewed from different points of view such as culture, collaboration, automation, measurements and monitoring (Mishra and Otaiwi 2020; Waller, Ehmke, and Hasselbring 2015). In DevOps there is a focus on speed and quality with incremental changes that are recurrent and continuous (Mishra and Otaiwi 2020). The goal is to bridge the gap between development and operations (Smeds, Nybom, and Porres 2015). This is done through sharing tasks and responsibilities from development to deployment and support (Mishra and Otaiwi 2020).

TODO: Add definition

Continuous integration, continuous deployment and continuous monitoring are well known practices in DevOps (Waller, Ehmke, and Hasselbring 2015) describing the automatic nature of integrating, deploying and monitoring code changes. Feedback includes performance metrics data which is then fed as an input during planning and development (Smeds, Nybom, and Porres 2015). Performance profiling and monitoring are similar activities and the main difference is whether it's done during the development process or during operations respectively (Waller, Ehmke, and Hasselbring 2015) with DevOps bridging the gap between them (Brunnert et al. 2015). Continuous benchmarking allows for detecting performance regressions during continuous integration (Waller, Ehmke, and Hasselbring 2015) and infrastructure monitoring with a feedback loop allows for performance optimization in production (Smeds, Nybom, and Porres 2015).

Performance evaluation is a useful tool for optimizing the overall system design and tailoring for a specific production environment in addition to correctly sizing resources (Brunnert et al. 2015; Waller, Ehmke, and Hasselbring 2015). Resource demands might change depending on the inputs (Brunnert et al. 2015) making it important to systematically measure performance not only based on code changes but also on configuration changes or even data changes. Performance evaluation is directly tied to defining and collecting performance metrics and monitoring.

2.2.2 Performance evaluation

Performance metrics are fundamental to all activities involving performance evaluation such as profiling or monitoring (Brunnert et al. 2015). Common metrics involve measuring the CPU, but other metrics such as memory usage, network traffic or I/O usage do not have clear definitions (Brunnert et al. 2015). Collecting metrics happens through hardware based monitors or software monitors instrumented into software through code modification or indirectly for example through middleware interception (Brunnert et al. 2015). Metrics can be event driven in which a monitor is triggered with every occurrence or based on sampling at fixed time intervals (Brunnert et al. 2015). The types of metrics collected and what information is expected depends on the performance goals and the life cycle of the software (Brunnert et al. 2015).

Metrics can be divided into application metrics such as response time or throughput and resource utilization metrics such as CPU utilization or available memory (Brunnert et al. 2015). There is little peer reviewed research available with specifics on which metrics are to be collected or how they are defined. Kounev et al. (2020) in their textbook on systems benchmarking bring up the following quality attributes for benchmark metrics: easy to measure, repeatable, reliable, linear, consistent and independent. Most metrics will not satisfy all of the above quality attributes and aggregated higher level composite metrics are requires (Kounev, Lange, and Von Kistowski 2020).

Measurement based performance evaluation requires a system to test while model based performance evaluation allows to predict the performance of the future system (Brunnert et al. 2015). This type of performance prediction allows for better planning and comparing use cases especially when an existing legacy system exists with measured performance metrics (Brunnert et al. 2015).

TODO: measuring CPU, memory, times (real, wall etc.)

2.3 MLOps

Machine learning operations or MLOps is a fairly new concept related to building and running real-world machine learning systems. This section introduces the concept of MLOps and provides context for the types of problems it aims to solve. Later in the section the concepts of hyperparameter optimization, performance prediction and early stopping are introduced. The section finishes with performance metrics related to machine learning systems and their business objectives and the performance of the overall system.

2.3.1 Production machine learning systems

While the focus of machine learning research has been on improving models, it is essential for the industry to be able to design production-ready machine learning pipelines (Posoldova 2020). The data often used for research is of higher quality than real-world data that is often messy, unstructured and unlabeled (Posoldova 2020). Continuous integration, continuous deployment and automated testing are also relevant to machine learning systems (Posoldova 2020) which are familiar concepts from DevOps. A new concept of MLOps addresses this issue of designing and maintaining machine learning systems just like DevOps addressed it for traditional software (Kreuzberger, Kühl, and Hirschl 2023).

Bring DevOps to MLOps

Managing technical debt is even more important in machine learning systems, because of machine learning specific issues that cannot be solved with traditional methods (Sculley et al. 2015). Main culprit for the challenges with machine learning systems is that data changes the behavior of the system and cannot be expressed with code alone (Sculley et al. 2015). Challenges like entanglement, correction cascades or feedback loops are common with machine learning systems and are difficult to diagnose with common tools (Sculley et al. 2015).

Requirements for a machine learning system are different depending on the task. For example speech and object recognition might have no particular performance requirements during training but has strict latency and computational resource restrictions when deployed to serve large amounts users (Hinton, Vinyals, and Dean 2015). MLOps has to take into account both machine learning performance metrics familiar from machine learning and software performance metrics familiar from DevOps and software engineering. Feedback from metrics collected during development and from monitoring of production systems are core MLOps principles (Kreuzberger, Kühl, and Hirschl 2023). For example possible meta-level requirements include users requesting data deletion, prohibitions on specific features like age or deperecated sources (Breck et al. 2017).

Performance measuring software is not new, but ML brings additional challenges in the form of models and data which requires a modified approach (Breck et al. 2017). It is also important to note, that not every data scientist or machine learning engineer working on machine learning systems has a software engineering background (Finzer 2013) and might lack the necessary knowledge to apply software engineering best practices to machine learning systems. Monitoring for machine learning systems has to be carefully designed (Sculley et al. 2015). Hyperparameter optimization is a kind of performance optimization, where the goal is to improve machine learning metrics. It is not always necessary to train the model to completion to verify that training code is correct and training loss is decreasing (Breck et al. 2017).

2.3.2 Hyperparameter optimization

Parameters given as part of a configuration to the machine learning model are called hyperparameters (Yang and Shami 2020). Examples of hyperparameters include learning rate, number of layers in a neural network, regularization coefficients, batch size, step size or initialization conditions (Maclaurin, Duvenaud, and Adams 2015; Baker et al. 2017; Breck et al. 2017). Hyperparameter tuning or hyperparameter optimization can be defined as finding the optimal hyperparameter values by searching through possible hyperparameter values (Baker et al. 2017). This hyperparameter search can also demonstrate whether the training is stable and reliable (Breck et al. 2017).

The main goal of hyperparameter optimization is to reduce the amount of expert labor required for creating high-performance machine learning models (Baker et al. 2017). Another benefit of finding optimal hyperparameters is that it can help achieve state-of-the-art performance in machine learning systems (Maclaurin, Duvenaud, and Adams 2015). Hyperparameter optimization techniques include grid search, random search, gradient based optimization and Bayesian optimization and they have different benefits and limitations (Yang and Shami 2020).

Similar concepts to hyperparameter optimization are neural architecture optimization and meta modeling where model structure or modeling algorithm is treated as a tunable parameter (Baker et al. 2017). This allows for automating the creation of neural networks from scratch (Baker et al. 2017). The amount of potential neural network architecture configurations is large and checking them is computationally expensive (Baker et al. 2017).

Tuning hyperparameters is generally a difficult task (Maclaurin, Duvenaud, and Adams 2015). Traditional hyperparameter tuning methods such as Bayesian optimization are unfeasible for more than 10-20 hyperparameters (Maclaurin, Duvenaud, and Adams 2015). More advanced techniques are required if a larger amount of tunable hyperparameters is desired. Performance prediction is an important step to reduce the amount of computation required for neural architecture search and hyperparameter optimization (Baker et al. 2017). Memory consumption, power consumption and training time are relevant considerations which can be taken into account by setting boundary conditions to whether the hyperparameter tuning trial is worthy of continuing (Yu and Zhu 2020).

TODO: add things from (Shallue et al. 2019)

2.3.3 Performance prediction and early stopping

TODO: Computational budget, steps,

TODO: Tuning budget

Data gathered at the beginning of model training can be used to predict performance of the trained model given the chosen hyperparameters (Baker et al. 2017). A small sample of hyperparameter configurations can be used for training a performance prediction model which then can be used to predict the performance for the rest of hyperparameter configurations with only a small amount of training (Baker et al. 2017).

Early stopping is a technique in which model training is halted before completion to avoid wasting computational resources (Prechelt 1998). Early stopping can be based on a threshold value decided upon ahead of time or based on a performance prediction model (Baker et al. 2017). Low thresholds for rejection of suboptimal solutions will radically reduce the amount of computation required, but run the risk of rejecting an optimal solution as well

(Baker et al. 2017).

Machine learning systems in addition to machine learning performance metrics and system performance metrics will have their performance metrics tied to product or organization metrics such as user churn rate or click-through rate (Shankar et al. 2022). Important metrics from a machine learning system performance perspective include CPU usage, GPU usage, task completion time, inference time and latency (Cardoso Silva et al. 2020). Choosing the right metrics to evaluate a machine learning system is important and the metrics will be different for different machine learning systems (Shankar et al. 2022).

3 Experiments

3.1 Methodology

This thesis uses a methodology for machine learning experiment design (Fernandez-Lozano et al. 2016). The methodology consists of a workflow with the following steps: Dataset, Data Preprocessing, Model Learning and Best Model Selection. The main focus of the thesis is on Model Learning and Best Model Selection with an emphasis of using system performance metrics. Advanced preprocessing techniques or achieving state of the art model performance are out of the scope of this thesis.

TODO Vertailukriteeristö: tapana ohjelmistopuolella + tapana koneoppimispuolella

This master's thesis asks the following research questions:

- *RQ1*: How does system performance change over time during model training?
- *RQ2*: How do changes in hyperparameters affect system performance during model training?
- *RQ3*: How does early stopping on system performance criteria affect computational budgets during model training?

3.2 Experimental setup

3.2.1 Software and Hardware

Experiments were performed using Ray Tune (2.7.1) (Liaw et al. 2018). MLFlow (2.7.1) (Chen et al. 2020) was used for recording metrics and tracking experiments. Scikit-learn (1.3.2) (Pedregosa et al. 2011) for training, collecting machine learning performance metrics and evaluating machine learning models. Psutil (Rodola 2023) was used for collecting system performance metrics from the operating system. Hardware used to perform the experiments consisted of Intel Core i7-9700 @ 3.00GHz CPU and Nvidia 3060 GPU.

3.2.2 Datasets

OpenML (Vanschoren et al. 2014) was a source of benchmarking datasets for both classification (Bischl et al. 2017) and regression (Fischer, Feurer, and Bischl 2023) tasks. In total two classification task and two regression task datasets summarized in table 1 were chosen to keep the amount of computation reasonable.

The *mnist_784* dataset consisted of 70000 images of handwritten digits with each feature representing a pixel with the task to classify which digit the image represents. The *diabetes* dataset consisted of 785 measurements of female patients with the task to classify whether the patient tests positive for diabetes. The *wave_energy* dataset consisted of 72000 different positions for 16 buoys with the regression task to predict the total amount of energy produced. The 16 wave energy converter features were dropped as the target variable is total energy. The *red_wine* dataset consisted of 1599 measurements of red wine samples with the regression task of predicting the quality of the wine.

Dataset	Type	Task	Instances	Features
mnist_784	image	classification	70000	785
diabetes	tabular	classification	768	9
wave_energy	tabular	regression	72000	33
red_wine	tabular	regression	1599	12

Table 1. Summary of the datasets used.

3.2.3 Algorithms

Algorithms were chosen to support training in batches without being computationally heavy. Linear regression, logistic regression and support vector machine (SVM) are based on stochastic gradient descent (SGD) implementation found in Scikit-learn (Pedregosa et al. 2011). Algorithms and hyperparameters are summarized in Table 2. Model training, evaluation and hyperparameter optimization was performed in parallel with each worker process using one CPU core each.

Hyperparameters such as batch size, learning rate and regularization alpha are selected using grid search with the search space determined with preliminary experiments so that the optimal solution is not too close to the boundaries. Batch size search space was 2^i for all i from 1 until 2^i was equal to the number of samples in the dataset. Learning rate search space was 10^i for all i between -1 and -4 .

Parempi tapa esittää, että batch size oli 2. potenssi välillä 2-n ja learning rate oli 10. potenssi välillä 0.1 - 0.0001

TODO: Add early stopping

Algorithm	Loss	Hyperparameters
Linear regression	squared	batch size, learning rate, alpha
Logistic regression	log	batch size, learning rate, alpha
Support Vector Machine	hinge	batch size, learning rate, alpha

Table 2. Summary of the algorithms

3.2.4 Metrics and evaluation

Metrics to be evaluated can be divided into machine learning metrics and system performance metrics and are summarized in Table 3. Machine learning metrics consisted of training loss, validation loss, accuracy for classification and root mean square error for regression respectively. System compute performance was measured through mean training step time and cpu utilization percentage. System memory performance was measured through memory use of the process and computational budget was measured as elapsed wall-time required for training the model. Training loss was computed with each training step and the rest of the metrics were computed every 100 training steps. Machine learning metrics were computed using scikit-learn (Pedregosa et al. 2011) and system performance metrics were collected from the operating system using psutil (Rodola 2023).

In accordance with Ray documentation (The Ray Team 2023) to avoid double counting memory used by the object store the memory usage of the worker was computed in the following way:

$$\text{memory} = \text{resident set size (RSS)} - \text{shared memory usage (SHR)}$$

Machine learning models were validated by splitting the dataset into a 70% training set and a 30% test set. To make sure that measurements are not sensitive to chosen data ten-fold cross

Metric	Type
training loss	machine learning
validation loss	machine learning
accuracy	machine learning
root mean square error	machine learning
mean training step time	system performance
total training time	system performance
cpu utilization (%)	system performance
memory (MB)	system performance

Table 3. Summary of the metrics

validation using the training set was performed when tuning hyperparameters.

TODO rank-based analysis using Friedman and Nemenyi post-hoc test (Fischer, Feurer, and Bischl 2023), bonferroni correction

3.2.5 Machine learning workflow

The following workflow was performed for each dataset:

1. Downloading the dataset in OpenML Dataset format
2. Transforming an OpenML Dataset to a Ray Dataset
3. Splitting the data into training and test datasets
4. Splitting the training data into 10-folds
5. Normalizing the training data for each fold and applying the transformation to the test dataset
6. Tuning each hyperparameter for each fold
7. Training the model
8. Early stopping if criteria are met

3.3 Results

First set of experiments were performed to determine whether system performance is constant during model training. Second set of experiments were performed to determine how

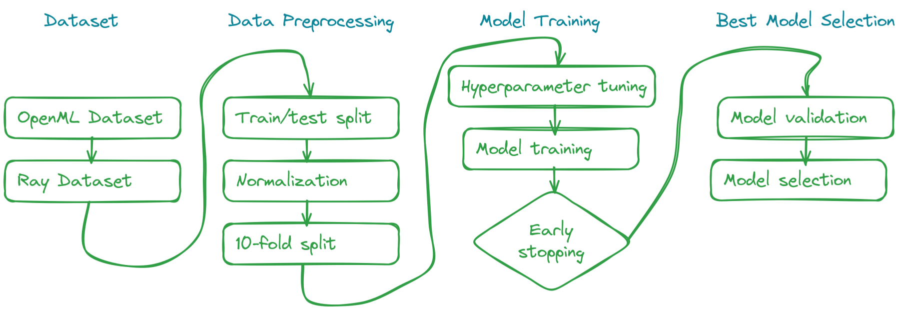

Figure 1. TODO placeholder image: ML workflow

changes in the hyperparameters affect system performance metrics. For each metric the average of all the runs was visualized and inspected for clear patterns with the focus on system performance metrics. Final set of experiments were performed to determine how setting a system performance criterion affects the computational budget can reduce the computational budget.

3.3.1 Changes in system performance during model training

The behavior of machine learning metrics was as expected with both the training loss and the validation loss decreased during training with the validation loss eventually plateaued. Accuracy increased during model training before plateauing.

Compute metrics mean training step time and cpu utilization percentage had variability but did not clearly decrease or increase during model training. Memory use steadily grew in a stepwise manner over time during model training.

Add figure

add msre

add details

add figure

add figure

add details

add shape of different algorithms and datasets

3.3.2 Effects of hyperparameter changes on system metrics

Batch size

Learning rate

alpha

3.3.3 Effects of early stopping on computational budgets

ML performance threshold

ram threshold

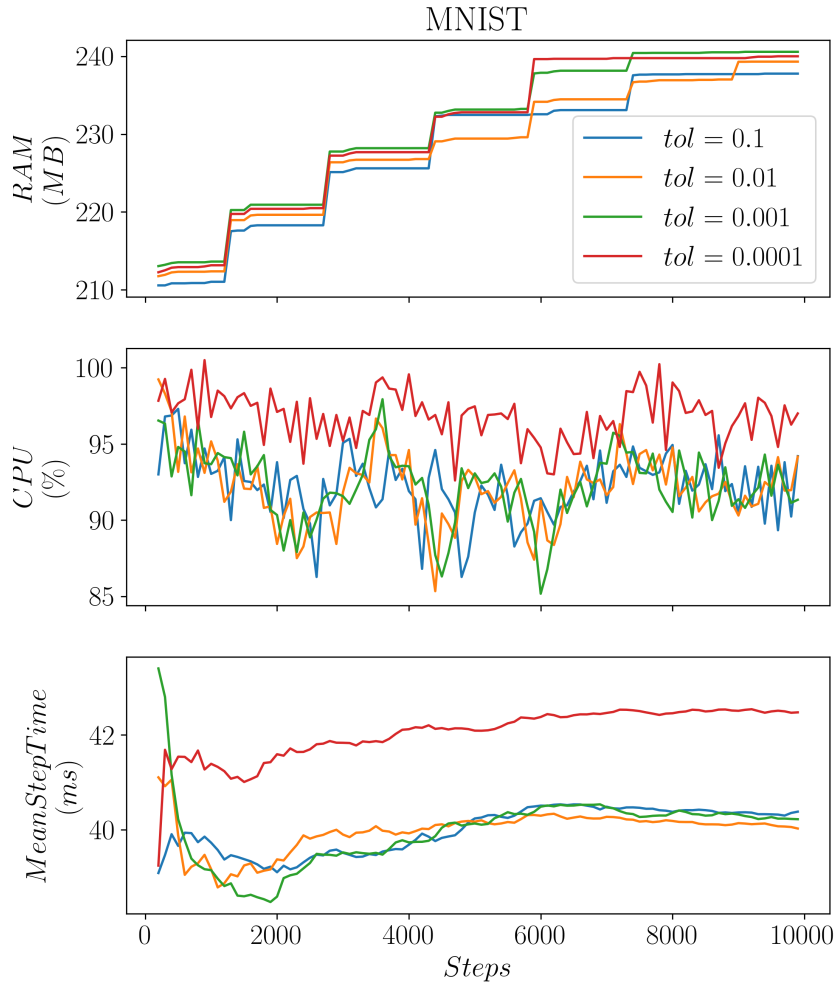

Figure 2. TODO placeholder image: Effect of tol hyperparameter on system performance metrics

4 Discussion

4.1 Research Questions revisited

4.1.1 Research question RQ1

4.1.2 Research question RQ2

4.1.3 Research question RQ3

4.2 Interpretation

4.2.1 Implications for research

4.2.2 Implications for practice

4.3 Limitations

4.3.1 Datasets

4.3.2 Machine Learning algorithms

4.3.3 Metrics

4.3.4 Training and validation

4.3.5 Inference

4.4 Related Work

To find relevant related work both reverse snowballing and forward snowballing is used on a set of MLOps papers previously known to the author.

Cardoso Silva et al. (2020) in their paper identify key system metrics for monitoring a production machine learning system. Key metrics identified include task completion time, CPU and GPU usage, memory usage, disk input/output and network traffic and their collection was implemented in a tool called *Ubenchmark* (Cardoso Silva et al. 2020). The researchers in particular focus on empirically monitoring and performance benchmarking of the machine learning system.

Karl et al. (2023) in their overview paper approach additional performance requirements such as energy efficiency, inference latency or memory consumption hyperparameter search through a multi-objective optimization lens.

4.5 Future Work

TODO This is a discussion chapter

5 Conclusions

Summary

TODO This is a conclusions chapter

Bibliography

Amodei, Dario, and Danny Hernandez. 2018. *AI and Compute.* https://openai.com/research/ai-and-compute. Visited on July 29, 2023.

Baker, Bowen, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. *Accelerating Neural Architecture Search Using Performance Prediction.* Visited on January 25, 2023. doi:`10.48550/arXiv.1705.10823`. arXiv: `1705.10823 [cs]`.

Bischl, Bernd, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. 2017. *OpenML Benchmarking Suites.* https://arxiv.org/abs/1708.03731v3. Visited on November 16, 2023.

Breck, Eric, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley. 2017. "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction". In *Proceedings of IEEE Big Data.*

Brunnert, Andreas, Andre van Hoorn, Felix Willnecker, Alexandru Danciu, Wilhelm Hasselbring, Christoph Heger, Nikolas Herbst, et al. 2015. *Performance-Oriented DevOps: A Research Agenda.* Visited on January 17, 2023. doi:`10.48550/arXiv.1508.04752`. arXiv: `1508.04752 [cs]`.

Cardoso Silva, Lucas, Fernando Rezende Zagatti, Bruno Silva Sette, Lucas Nildaimon dos Santos Silva, Daniel Lucrédio, Diego Furtado Silva, and Helena de Medeiros Caseli. 2020. "Benchmarking Machine Learning Solutions in Production". In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA),* 626–633. doi:`10.1109/ICMLA51294.2020.00104`.

Cawley, Gavin C, and Nicola L C Talbot. No date. "On Over-fittingg inModel Selectionn andSubsequent Selection Biass inPerformance Evaluation".

Chen, Andrew, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, et al. 2020. “Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle”. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning,* 1–4. DEEM’20. New York, NY, USA: Association for Computing Machinery. ISBN: 978-1-4503-8023-2, visited on October 24, 2023. doi:10.1145/3399579.3399867.

Chen, Jiasi, and Xukan Ran. 2019. “Deep Learning With Edge Computing: A Review”. *Proceedings of the IEEE* 107, number 8 (): 1655–1674. ISSN: 0018-9219, 1558-2256, visited on July 29, 2023. doi:10.1109/JPROC.2019.2921977.

Domingos, Pedro. 2012. “A Few Useful Things to Know about Machine Learning”. *Communications of the ACM* 55, number 10 (): 78–87. ISSN: 0001-0782, visited on March 14, 2023. doi:10.1145/2347736.2347755.

Fernandez-Lozano, Carlos, Marcos Gestal, Cristian R. Munteanu, Julian Dorado, and Alejandro Pazos. 2016. “A Methodology for the Design of Experiments in Computational Intelligence with Multiple Regression Models”. *PeerJ* 4 (): e2721. ISSN: 2167-8359, visited on February 15, 2023. doi:10.7717/peerj.2721.

Finzer, William. 2013. “The Data Science Education Dilemma”. *Technology Innovations in Statistics Education* 7 (2). Visited on January 17, 2023. doi:10.5070/T572013891.

Fischer, Sebastian Felix, Matthias Feurer, and Bernd Bischl. 2023. “OpenML-CTR23 – A Curated Tabular Regression Benchmarking Suite”. In *AutoML Conference 2023 (Workshop).* Visited on November 16, 2023.

Forman, George, and Martin Scholz. No date. “Apples-to-Apples in Cross-Validation Studies: Pitfalls in Classifier Performance Measurement”.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. 2015. *Distilling the Knowledge in a Neural Network.* Visited on February 3, 2023. doi:10.48550/arXiv.1503.02531. arXiv: 1503.02531 [cs, stat].

Hoffer, Elad, Itay Hubara, and Daniel Soudry. 2018. *Train Longer, Generalize Better: Closing the Generalization Gap in Large Batch Training of Neural Networks.* Visited on August 2, 2023. doi:10.48550/arXiv.1705.08741. arXiv: 1705.08741 [cs, stat].

Janocha, Katarzyna, and Wojciech Marian Czarnecki. 2017. *On Loss Functions for Deep Neural Networks in Classification.* Visited on August 27, 2023. doi:10.48550/arXiv.1702.05659. arXiv: 1702.05659 [cs].

Karl, Florian, Tobias Pielok, Julia Moosbauer, Florian Pfisterer, Stefan Coors, Martin Binder, Lennart Schneider, et al. 2023. "Multi-Objective Hyperparameter Optimization in Machine Learning – An Overview". *ACM Transactions on Evolutionary Learning and Optimization* (). Visited on October 17, 2023. doi:10.1145/3610536.

Kounev, Samuel, Klaus-Dieter Lange, and Jóakim Von Kistowski. 2020. *Systems Benchmarking: For Scientists and Engineers.* Cham: Springer International Publishing. ISBN: 978-3-030-41704-8 978-3-030-41705-5, visited on August 23, 2023. doi:10.1007/978-3-030-41705-5.

Kreuzberger, Dominik, Niklas Kühl, and Sebastian Hirschl. 2023. "Machine Learning Operations (MLOps): Overview, Definition, and Architecture". *IEEE Access* 11:31866–31879. ISSN: 2169-3536. doi:10.1109/ACCESS.2023.3262138.

Le, Quoc V., Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. 2012. *Building High-Level Features Using Large Scale Unsupervised Learning.* Visited on February 3, 2023. doi:10.48550/arXiv.1112.6209. arXiv: 1112.6209 [cs].

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning". *Nature* 521, number 7553 (): 436–444. ISSN: 1476-4687, visited on June 15, 2023. doi:10.1038/nature14539.

Liaw, Richard, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. *Tune: A Research Platform for Distributed Model Selection and Training.* Visited on February 22, 2023. doi:10.48550/arXiv.1807.05118. arXiv: 1807.05118 [cs, stat].

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. "Microsoft COCO: Common Objects in Context". In *Computer Vision – ECCV 2014,* edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, 740–755. Lecture Notes in Computer Science. Cham: Springer International Publishing. ISBN: 978-3-319-10602-1. doi:10.1007/978-3-319-10602-1_48.

Maclaurin, Dougal, David Duvenaud, and Ryan P. Adams. 2015. *Gradient-Based Hyperparameter Optimization through Reversible Learning.* Visited on February 3, 2023. doi:10.48550/arXiv.1502.03492. arXiv: 1502.03492 [cs, stat].

Mishra, Alok, and Ziadoon Otaiwi. 2020. "DevOps and Software Quality: A Systematic Mapping". *Computer Science Review* 38 (): 100308. ISSN: 1574-0137, visited on January 17, 2023. doi:10.1016/j.cosrev.2020.100308.

OpenAI. 2022. *Introducing ChatGPT.* https://openai.com/blog/chatgpt. Visited on July 24, 2023.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python". *Journal of Machine Learning Research* 12 (85): 2825–2830. ISSN: 1533-7928, visited on October 25, 2023.

Perera, Pulasthi, Roshali Silva, and Indika Perera. 2017. "Improve Software Quality through Practicing DevOps". In *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer),* 1–6. doi:10.1109/ICTER.2017.8257807.

Posoldova, Alexandra. 2020. "Machine Learning Pipelines: From Research to Production". *IEEE Potentials* 39, number 6 (): 38–42. ISSN: 1558-1772. doi:10.1109/MPOT.2020.3016280.

Prechelt, Lutz. 1998. "Automatic Early Stopping Using Cross Validation: Quantifying the Criteria". *Neural Networks* 11, number 4 (): 761–767. ISSN: 0893-6080, visited on August 2, 2023. doi:10.1016/S0893-6080(98)00010-0.

Rodola, Giampaolo. 2023. *Giampaolo/Psutil.* Visited on November 2, 2023.

Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. *High-Resolution Image Synthesis with Latent Diffusion Models.* Visited on July 22, 2023. doi:10.48550/arXiv.2112.10752. arXiv: 2112.10752 [cs].

Sculley, D., Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. “Hidden Technical Debt in Machine Learning Systems”. In *Advances in Neural Information Processing Systems,* volume 28. Curran Associates, Inc. Visited on August 13, 2023.

Shallue, Christopher J., Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. 2019. “Measuring the Effects of Data Parallelism on Neural Network Training”. *Journal of Machine Learning Research* 20 (112): 1–49. ISSN: 1533-7928, visited on September 28, 2023.

Shankar, Shreya, Rolando Garcia, Joseph M. Hellerstein, and Aditya G. Parameswaran. 2022. *Operationalizing Machine Learning: An Interview Study.* Visited on December 7, 2022. doi:10.48550/arXiv.2209.09125. arXiv: 2209.09125 [cs].

Shmueli, Galit. 2010. “To Explain or to Predict?” *Statistical Science* 25, number 3 (). ISSN: 0883-4237, visited on March 14, 2023. doi:10.1214/10-STS330. arXiv: 1101.0891 [stat].

Smeds, Jens, Kristian Nybom, and Ivan Porres. 2015. “DevOps: A Definition and Perceived Adoption Impediments”. In *Agile Processes in Software Engineering and Extreme Programming,* edited by Casper Lassenius, Torgeir Dingsøyr, and Maria Paasivaara, 166–177. Lecture Notes in Business Information Processing. Cham: Springer International Publishing. ISBN: 978-3-319-18612-2. doi:10.1007/978-3-319-18612-2_14.

Sokolova, Marina, and Guy Lapalme. 2009. “A Systematic Analysis of Performance Measures for Classification Tasks”. *Information Processing & Management* 45, number 4 (): 427–437. ISSN: 0306-4573, visited on August 27, 2023. doi:10.1016/j.ipm.2009.03.002.

Stability AI. 2022. *Stable Diffusion Public Release.* https://stability.ai/blog/stable-diffusion-public-release. Visited on July 24, 2023.

Strubell, Emma, Ananya Ganesh, and Andrew McCallum. 2020. "Energy and Policy Considerations for Modern Deep Learning Research". *Proceedings of the AAAI Conference on Artificial Intelligence* 34, number 09 (): 13693–13696. ISSN: 2374-3468, visited on July 29, 2023. doi:10.1609/aaai.v34i09.7123.

The Ray Team. 2023. *Memory Management — Ray 2.7.1.* https://docs.ray.io/en/latest/ray-core/scheduling/memory-management.html.

Torralba, Antonio, and Alexei A. Efros. 2011. "Unbiased Look at Dataset Bias". In *CVPR 2011,* 1521–1528. Colorado Springs, CO, USA: IEEE. ISBN: 978-1-4577-0394-2, visited on August 27, 2023. doi:10.1109/CVPR.2011.5995347.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. 2023. *LLaMA: Open and Efficient Foundation Language Models.* Visited on July 24, 2023. doi:10.48550/arXiv.2302.13971. arXiv: 2302.13971 [cs].

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2014. "OpenML: Networked Science in Machine Learning". *ACM SIGKDD Explorations Newsletter* 15, number 2 (): 49–60. ISSN: 1931-0145, visited on November 16, 2023. doi:10.1145/2641190.2641198.

Waller, Jan, Nils C. Ehmke, and Wilhelm Hasselbring. 2015. "Including Performance Benchmarks into Continuous Integration to Enable DevOps". *ACM SIGSOFT Software Engineering Notes* 40, number 2 (): 1–4. ISSN: 0163-5948, visited on January 17, 2023. doi:10.1145/2735399.2735416.

Yang, Li, and Abdallah Shami. 2020. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice". *Neurocomputing* 415 (): 295–316. ISSN: 09252312, visited on January 25, 2023. doi:10.1016/j.neucom.2020.07.061. arXiv: 2007.15745 [cs, stat].

Yu, Tong, and Hong Zhu. 2020. *Hyper-Parameter Optimization: A Review of Algorithms and Applications.* Visited on January 10, 2024. doi:10.48550/arXiv.2003.05689. arXiv: 2003.05689 [cs, stat].

Zaharia, M., A. Chen, A. Davidson, A. Ghodsi, S. Hong, A. Konwinski, Siddharth Murching, et al. 2018. “Accelerating the Machine Learning Lifecycle with MLflow”. *IEEE Data Eng. Bull.* Visited on March 14, 2023.