Machine Learning Design for Business

Brad Boehmke

2024-11-14

Table of contents

Welcome

Welcome to ***Machine Learning Design for Business***! This is the the online textbook that compliments the UC BANA 7085 course. This course aims to provide a framework for developing real-world machine learning systems that are deployable, reliable, and scalable. You will be introduced to ***MLOps*** as the intersection of DataOps, ModelOps, and DevOps; combined, these concepts provide scalable, reliable, and repeatable machine learning workflows. You will learn how MLOps enables organizations to overcome challenges in operationalizing machine learning models. Along the way, you will learn about important organizational issues including privacy, fairness, security, stakeholder management, project collaboration, and more!

Throughout this textbook you'll find embedded lectures, hands-on exercises, and additional resources that will allow you to explore the tools, techniques, and best practices required to successfully design and deploy machine learning systems in today's organizations.

Who should read this

This book is ideal for graduate students, data scientists, engineers, and analytics professionals who want to bridge the gap between building machine learning models and deploying them into reliable, scalable production environments. Those looking to deepen their understanding of operationalizing ML models will benefit from hands-on experience with MLOps workflows, including data management, model deployment, and monitoring. Individuals with a basic background in machine learning or software development will gain practical skills to streamline the machine learning lifecycle and align their work with organizational and business goals.

This book is also valuable for those interested in cross-functional collaboration, providing insights into how data science teams can work alongside engineering and business units to achieve successful machine learning outcomes.

This book is technical in nature, designed to provide hands-on experience in deploying, monitoring, and managing machine learning systems. While it's possible to successfully complete the book with minimal coding experience, having some familiarity with Python and basic machine learning concepts will be beneficial for understanding and applying the material. The course introduces technical tools and workflows, but it's structured to guide students of various backgrounds through each step. Those with prior experience in Python or ML will have a smoother time grasping the concepts and may find it easier to implement projects independently. However, foundational knowledge in programming or machine learning is not strictly

required, as core principles and step-by-step instructions are provided to support all readers in building effective MLOps pipelines.

> **Note**
>
> If you are new to Python, I recommend you check out https://bradleyboehmke.github.io/uc-bana-6043 to get up to speed with the basics.

Conventions used in this book

The following typographical conventions are used in this book:

- ***strong italic***: indicates new terms,
- **bold**: indicates package & file names,
- `inline code`: monospaced highlighted text indicates functions or other commands that could be typed literally by the user,
- code chunk: indicates commands or other text that could be typed literally by the user

```
1 + 2
```

```
3
```

In addition to the general text used throughout, you will notice the following code chunks with images:

> **Tip**
>
> Signifies a tip or suggestion

> **Note**
>
> Signifies a general note

> **Warning**
>
> Signifies a warning or caution

Software used throughout this book

TBD

Additional resources

TBD

1 Introduction to Machine Learning System Design

Machine Learning (ML) System Design is the discipline of creating reliable, scalable, and maintainable machine learning systems that solve real-world business problems. Unlike standard machine learning modeling, which focuses primarily on algorithm development and evaluation, ML System Design considers the broader ecosystem in which a model must operate. This involves defining the architecture, infrastructure, and processes that make a model functional and sustainable within production environments. As organizations increasingly rely on machine learning to make data-driven decisions, robust system design has become essential for delivering models that consistently meet organizational needs and can adapt to change.

Effective ML system design enables organizations to avoid many pitfalls of deploying machine learning in production. Poorly designed systems can lead to issues like inaccurate predictions due to data drift, high operational costs, or significant downtime during retraining or model updates. By adhering to design principles such as modularity, scalability, and reproducibility, ML engineers and data scientists can develop systems that are more resilient and easier to manage. These principles form the backbone of good ML system design and help ensure that models can not only be deployed but also monitored, improved, and scaled as needed.

This section will explore these design principles in detail and examine how they align with key stages of the ML system lifecycle, from data processing and model training to deployment, monitoring, and iteration. As Chip Huyen points out, "Machine learning in production is 10% modeling and 90% engineering" (Huyen 2022), highlighting how technical choices impact the system's reliability and long-term value for the organization. Understanding ML System Design and the lifecycle of ML systems is foundational for anyone working in machine learning, as it highlights how technical choices impact the system's overall reliability and long-term value for the organization.

1.1 ML system design

ML system design refers to the process of building, structuring, and integrating machine learning models into larger production environments to deliver reliable, scalable, and sustainable solutions. While traditional machine learning focuses on developing high-performing models, system design extends the focus to creating an entire ecosystem where these models can continuously operate, adapt, and provide value over time. It encompasses everything from

data pipelines and infrastructure to model versioning, monitoring, and scaling, ensuring that machine learning projects align with organizational goals and can be maintained as data, requirements, and technology evolve.

Therefore, ML system design includes multiple facets beyond the algorithms themselves:

> TODO
>
> Add wire diagram to depict ML system in an organization - showing business requirements driving MLOps components leading to the end user.

1.1.1 DataOps: Data Management and Pipelines

At the heart of any ML system is data, which is processed, transformed, and delivered to the model in a way that supports both training and inference. A well-designed ML system typically includes data pipelines that handle:

- Data ingestion: Collecting data from various sources, including databases, APIs, and real-time streams.
- Data processing: Cleaning, transforming, and structuring data for model consumption.
- Data validation: Ensuring that data quality is maintained and meets the standards required by the model.
- Data versioning and lineage: Tracking data versions and maintaining a record of data transformations, which are critical for reproducibility and regulatory compliance.

1.1.2 ModelOps: Model Lifecycle Management

A central aspect of ML system design is managing the entire model lifecycle, from development and testing to deployment and retirement. This includes:

- Experiment tracking and versioning: Maintaining a record of model versions, hyperparameters, and evaluation metrics to track performance and support reproducibility.
- Deployment pipelines: Automating model deployment processes to reduce manual errors and accelerate time-to-production.
- Monitoring and alerting: Establishing mechanisms for monitoring model performance in production and detecting issues such as data drift, concept drift, or system degradation.
- Continuous improvement: Supporting processes for regular retraining, fine-tuning, and updating models as data and requirements change.

1.1.3 DevOps Practices

Good ML system design brings DevOps practices to the world of machine learning. DevOps, short for "Development and Operations," is a set of practices and principles that combines software development (Dev) and IT operations (Ops) with the goal of shortening the software development lifecycle and delivering high-quality, reliable software more efficiently. At its core, DevOps is about fostering a culture of collaboration between development and operations teams, integrating automated processes, and leveraging continuous integration and continuous deployment (CI/CD) practices to streamline workflows. This includes:

- Automating code and model updates to streamline deployment and minimize downtime.
- Automation of testing and validation: Implementing automated tests to validate model performance, accuracy, and reliability.
- Compute resources: Setting up scalable, flexible compute resources (such as cloud instances, GPUs, or Kubernetes clusters) to handle model training and inference.
- Storage solutions: Designing data storage strategies that balance cost, speed, and accessibility.
- Networking and security: Ensuring secure data transfer and protecting sensitive information while allowing fast access for machine learning processes.
- Collaboration between teams: Encouraging collaboration across data science, engineering, and business teams to ensure that models meet organizational requirements and add real value.

! Driven by business requirements

The design of each key element in an ML system is ultimately driven by business requirements, as they define the purpose and value that the system should deliver. Business goals shape how data is processed, what metrics the model optimizes, and the reliability, scalability, and security standards it must meet. For example, a financial institution deploying a fraud detection model may prioritize stringent data validation and versioning to comply with regulatory standards, while a retail organization focused on real-time recommendations may emphasize rapid data ingestion and low-latency deployment.
Model lifecycle management is similarly guided by business needs; frequent retraining cycles, experiment tracking, and monitoring capabilities are crucial when a system must adapt to fast-changing consumer behavior or market trends. Infrastructure choices, such as selecting between cloud and on-premise solutions, also reflect organizational constraints, like budget or resource availability.
By aligning these elements with clear business objectives, organizations can create ML systems that are not only technically sound but also effective in achieving their strategic goals.

1.2 Why ML system design matters

ML system design addresses the challenges of productionizing machine learning. In a laboratory or research setting, machine learning models are often developed with a focus on achieving high accuracy or minimizing error on a given dataset. However, deploying these models in production involves a different set of concerns. In real-world applications, ML systems need to handle fluctuating data distributions, evolving user needs, and high availability. Without a robust design, machine learning models can quickly degrade in performance, lead to incorrect predictions, or even disrupt business operations.

By prioritizing system design, organizations can build ML workflows that are both responsive to change and maintainable over time. This approach reduces technical debt—the accumulation of outdated or poorly designed code, data dependencies, and complex interactions that are difficult to fix — highlighted by the authors in Hidden Technical Debt in Machine Learning Systems as a critical challenge in ML applications (Sculley et al. 2015). Poorly managed dependencies and hidden feedback loops, for instance, can lead to unpredictable model behavior and complicate future updates. Thoughtful ML system design mitigates these issues, allowing models to deliver consistent, reliable results while aligning predictions more closely with organizational metrics and objectives. Ultimately, well-architected ML systems are more adaptable, enabling models to evolve as business goals and data distributions shift over time.

1.3 Design principles for good ML system design

Creating an effective ML system involves more than just building an accurate model; it requires thoughtful design to ensure that the system remains reliable, scalable, and adaptable over time. Good ML system design incorporates a set of principles that help address real-world challenges in production environments, such as evolving data, model drift, and changing business requirements. The following design principles guide ML engineers and data scientists in building systems that meet these demands and deliver sustained business value.

> TODO
>
> Add image that shows all principles

1.3.1 Modularity and Abstraction

Modularity is the practice of dividing a system into independent, self-contained components, each responsible for a specific function. In an ML system, this could mean separating data ingestion, preprocessing, training, and monitoring into distinct modules that can operate independently (Sculley et al. 2015). This modularity simplifies updates and maintenance, as each component can be modified or scaled without disrupting the entire system. For example, if

a new feature engineering technique improves model accuracy, it can be implemented in the data preprocessing module without affecting the deployment pipeline.

Abstraction, on the other hand, hides complex details within each module, making the system easier to manage and more accessible to team members who don't need to understand every technical detail. As noted by Huyen (2022), abstraction helps bridge the gap between data scientists, engineers, and business stakeholders, allowing each to focus on high-level functionality without needing an in-depth understanding of every process. Together, modularity and abstraction improve collaboration, flexibility, and ease of scaling within ML systems.

1.3.2 Scalability

Scalability ensures that an ML system can handle increased demand—whether in terms of data volume, number of users, or computational load—without requiring extensive re-engineering. Designing for scalability involves choosing infrastructure and tools that allow for efficient resource allocation and growth. For instance, deploying a model on cloud infrastructure, where compute resources can be dynamically allocated, allows an ML system to scale up to handle peak loads and scale down during quieter periods, optimizing costs and performance (Levine et al. 2020).

Another aspect of scalability is planning for future expansion. An ML system might start with a few users or limited data, but as it proves valuable, the system may need to accommodate more data sources, larger user bases, or additional model versions. Scalable design considers these future needs from the outset, making it easier to extend the system's capabilities without significant architectural changes.

1.3.3 Reproducibility

Reproducibility is critical for ML systems, particularly in production settings where consistent results and traceability are essential. Reproducibility ensures that anyone with access to the same code, data, and configuration can recreate a model with the same results. This is essential for troubleshooting, compliance, and validating model performance. As described by Amershi et al. (2019), reproducibility allows teams to understand how specific predictions are made and to resolve any issues that arise in production more efficiently.

Achieving reproducibility requires careful versioning of data, models, and code. Version control tools like Git and DVC (Data Version Control) help keep track of changes to code and data over time, while tools like MLflow or Weights & Biases allow for experiment tracking and versioning of model parameters. Reproducibility is especially important in regulated industries, where organizations may need to demonstrate how a model made a particular prediction or prove compliance with specific standards.

1.3.4 Automation

Automation is a foundational principle in ML system design, as it reduces manual intervention, minimizes human error, and accelerates workflows. As described by Polyzotis et al. (2019), automation is particularly valuable in environments where frequent model retraining, testing, or deployment is necessary. For instance, an automated pipeline can retrain a model when new data is available, perform validation checks, and deploy the updated model to production, all without requiring human involvement.

Key areas for automation in ML systems include data ingestion and preprocessing, model training and evaluation, deployment, and monitoring. Automating these stages not only saves time but also ensures that the system operates consistently and can adapt to changing data and conditions. This is crucial in fast-paced environments, such as e-commerce or finance, where systems must quickly respond to new information.

1.3.5 Monitoring and Maintenance

Monitoring and maintenance are essential for keeping an ML system aligned with its intended goals over time. Once a model is deployed, its performance can be affected by data drift (changes in data distribution) or concept drift (changes in the underlying relationships within the data) (Gama et al. 2014). Without monitoring, these issues can lead to model degradation, resulting in inaccurate predictions or decisions that don't meet business objectives.

Good ML system design includes monitoring dashboards and automated alerts that track metrics such as prediction accuracy, latency, data drift, and system errors. Tools like Prometheus and Grafana can help set up performance monitoring, while specialized ML monitoring solutions can track model-specific metrics. Regular maintenance practices, such as retraining or recalibrating models, are also essential for sustaining system performance and adapting to new data patterns or business requirements.

1.3.6 Security and Compliance

Security and compliance are often overlooked in ML system design but are increasingly important, especially in regulated industries like healthcare, finance, and government. An ML system that processes sensitive data must include security measures to protect data integrity and confidentiality. Security considerations include encrypting data in transit and at rest, managing user access controls, and safeguarding against adversarial attacks (Papernot et al. 2017).

Compliance involves ensuring that the ML system adheres to industry regulations and organizational policies. For instance, GDPR compliance may require an ML system to provide

transparency into how predictions are made and to allow users to request data deletions. Designing with security and compliance in mind not only protects the organization but also builds trust with users and stakeholders.

> What's GDPR?
>
> The General Data Protection Regulation (GDPR) is a comprehensive privacy law enacted by the European Union in 2018, aimed at protecting the personal data and privacy of individuals within the EU and the European Economic Area (EEA). GDPR sets strict guidelines on how organizations must collect, store, process, and share personal data, giving individuals more control over their information. Key principles of GDPR include data minimization, purpose limitation, and transparency, along with rights for individuals such as the right to access, correct, and delete their data. Non-compliance with GDPR can lead to significant fines, making it essential for companies, especially those dealing with machine learning systems, to prioritize data security and privacy to ensure compliance.

1.3.7 Adaptability and Flexibility

Adaptability is the ability of an ML system to evolve as data, business requirements, and technologies change. In a dynamic business environment, the needs of an ML system may shift over time, requiring the model to be updated, new features to be added, or the infrastructure to be reconfigured (Sculley et al. 2015). Designing for adaptability involves using flexible frameworks, modular components, and tools that support rapid changes without extensive rework.

For example, a well-designed ML system might use containerization (e.g., Docker) to allow models to be easily moved across environments or use APIs to integrate new data sources without significant restructuring. This flexibility enables teams to quickly implement changes, test new ideas, and stay aligned with business goals, making the system more resilient to future changes.

1.3.8 Putting It All Together

These principles — modularity, scalability, reproducibility, automation, monitoring and maintenance, security, and adaptability — form the foundation of good ML system design. Each principle contributes to creating a system that is not only technically robust but also capable of delivering sustained business value. By applying these principles, ML engineers and data scientists can build systems that are flexible, resilient, and scalable, ensuring that machine learning models remain effective and reliable over time. Good ML system design allows organizations to move beyond individual models and create a stable, agile infrastructure that continuously adapts to meet evolving business demands.

1.4 Relation to ML system lifecycle stages

Good ML system design requires a holistic approach that considers how key elements and design principles intersect with each stage of the ML system lifecycle. The ML lifecycle can be broken down into distinct stages: data processing, model development, deployment, monitoring, and continuous improvement. By aligning design principles with these stages, organizations can create systems that are not only effective at deployment but sustainable, adaptable, and closely aligned with business objectives over the long term.

> TODO
>
> Add image that shows ML system lifecycle stages

1.4.1 Data Processing

The data processing stage encompasses everything from data ingestion and transformation to validation and storage. Principles like modularity and automation are especially valuable here, as they allow data pipelines to handle diverse data sources and adapt to changes without disrupting downstream processes. DataOps practices ensure the accuracy and integrity of data, including validation and lineage tracking, so the right data reaches the model reliably. Reproducibility also plays a crucial role at this stage, as versioned datasets and pipelines help maintain consistency, which is vital for model retraining and compliance.

1.4.2 Model Development

Model development involves experimentation, model training, and evaluation. Here, modularity and reproducibility are essential for managing experiments and enabling easy comparison of results across different model versions. ModelOps practices, including experiment tracking and model versioning, facilitate tracking hyperparameters, code versions, and results, ensuring models can be accurately reproduced and refined. Scalability is another key consideration, especially for large datasets or complex models that require significant computational resources. In this stage, infrastructure choices, such as cloud or GPU-based solutions, allow the system to scale up as needed.

1.4.3 Deployment

The deployment stage focuses on moving models from development to production in a seamless, automated, and reliable manner. Automation is paramount, with CI/CD pipelines automating

the testing, validation, and deployment processes to minimize human error and expedite time-to-production. DevOps principles enable automated deployment processes that integrate testing and validation steps, reducing downtime and increasing reliability. Security also becomes critical, ensuring that data access and model endpoints are protected against unauthorized access or adversarial attacks.

1.4.4 Monitoring and Maintenance

Once a model is live, continuous monitoring is essential to track key metrics such as accuracy, latency, and model drift, ensuring that the model performs as expected over time. Monitoring and maintenance principles are implemented through tools and dashboards that monitor data drift and system performance. Alerts can notify teams of anomalies or degradation, enabling quick action to retrain or update the model if needed. Automation also supports regular retraining workflows, particularly in dynamic environments where data patterns shift frequently. Maintenance ensures that the ML system remains robust and responsive to changing data and requirements.

1.4.5 Continuous Improvement

The continuous improvement stage focuses on enhancing the model based on feedback from monitoring and evolving business needs. Adaptability and flexibility are essential here, allowing the system to integrate new data, retrain models, and test enhancements with minimal friction. Scalability and modularity enable models to grow and improve without requiring complete redesigns of the system. In practice, this means designing systems with modular components that can easily integrate new models or data sources, ensuring that the system remains aligned with both technical advancements and organizational goals.

1.5 Summary

In this chapter, we introduced the concept of Machine Learning (ML) System Design, focusing on how it creates a foundation for building reliable, scalable, and adaptable machine learning systems that can deliver value in real-world production environments. Unlike traditional model development, ML system design considers the entire ecosystem, encompassing data pipelines, model lifecycle management, and operational infrastructure, all shaped by business requirements. By following core design principles — such as modularity, scalability, reproducibility, automation, monitoring, and security — ML systems can avoid common production pitfalls like model drift, high operational costs, and downtime, ensuring that models are not only deployed successfully but also perform reliably over time.

The chapters that follow will expand on each element of ML system design, providing readers with a deep understanding of the key concepts and principles that enable the creation of robust ML systems. Readers will gain the ability to identify essential design decisions that align with business goals, as well as the tools and practices necessary to implement these considerations effectively. By the end of this book, readers will be equipped to build, manage, and maintain ML systems that are both technically sound and responsive to evolving business needs.

1.6 Exercise

Read one or more of the following case studies. These case studies cover different perspectives, elements, and principles of ML system design.

1. What are the primary business objective(s) for the ML system discussed? How does the system add value for both the organization and its customers?
2. Analyze key elements and design principles discussed. For example:
    - DataOps: How does the organization handle data ingestion, processing, and versioning to support a constantly changing catalog and user preferences?
    - ModelOps: What strategies does the organization use to manage the lifecycle of its models, including retraining, versioning, and experimentation?
    - DevOps: How does the organization ensure continuous integration and deployment for its models? What automation, testing, and monitoring practices are used to maintain system reliability?
    - Design Principles: Identify examples of modularity, scalability, automation, and monitoring within the organization's ML system. Explain how each principle contributes to the system's success.

Note

Note that these articles will likely mention technical concepts and tools that you may not be familiar with and feel like they are over your head. Do not worry about this! This book will help to bridge some of these gaps.

Case Studies

- Netflix: Netflix's Recommendation System: Algorithms, Business Value, and Innovation
- Uber: Scaling Machine Learning at Uber with Michelangelo
- Spotify: Inside Spotify's Recommender System: A Complete Guide to Spotify Recommendation Algorithms
- Airbnb: Bighead: A Framework-Agnostic, End-to-End Machine Learning Platform

- Google: TFX - an end-to-end machine learning platform for TensorFlow
- LinkedIn: Scaling Machine Learning Productivity at LinkedIn
- Facebook: Introducing FBLearner Flow: Facebook's AI backbone
- Booking.com: 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com
- Twitter: Twitter's Recommendation Algorithm

2 Summary

In summary, this book has no content whatsoever.

References

Amershi, Saleema, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. "Software Engineering for Machine Learning: A Case Study." In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300. IEEE.

Gama, João, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. "A Survey on Concept Drift Adaptation." *ACM Computing Surveys (CSUR)* 46 (4): 1–37.

Huyen, Chip. 2022. *Designing Machine Learning Systems.* " O'Reilly Media, Inc.".

Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu. 2020. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems." *arXiv Preprint arXiv:2005.01643*.

Papernot, Nicolas, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. "Practical Black-Box Attacks Against Machine Learning." In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–19.

Polyzotis, Neoklis, Martin Zinkevich, Sudip Roy, Eric Breck, and Steven Whang. 2019. "Data Validation for Machine Learning." *Proceedings of Machine Learning and Systems* 1: 334–47.

Sculley, David, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. "Hidden Technical Debt in Machine Learning Systems." *Advances in Neural Information Processing Systems* 28.