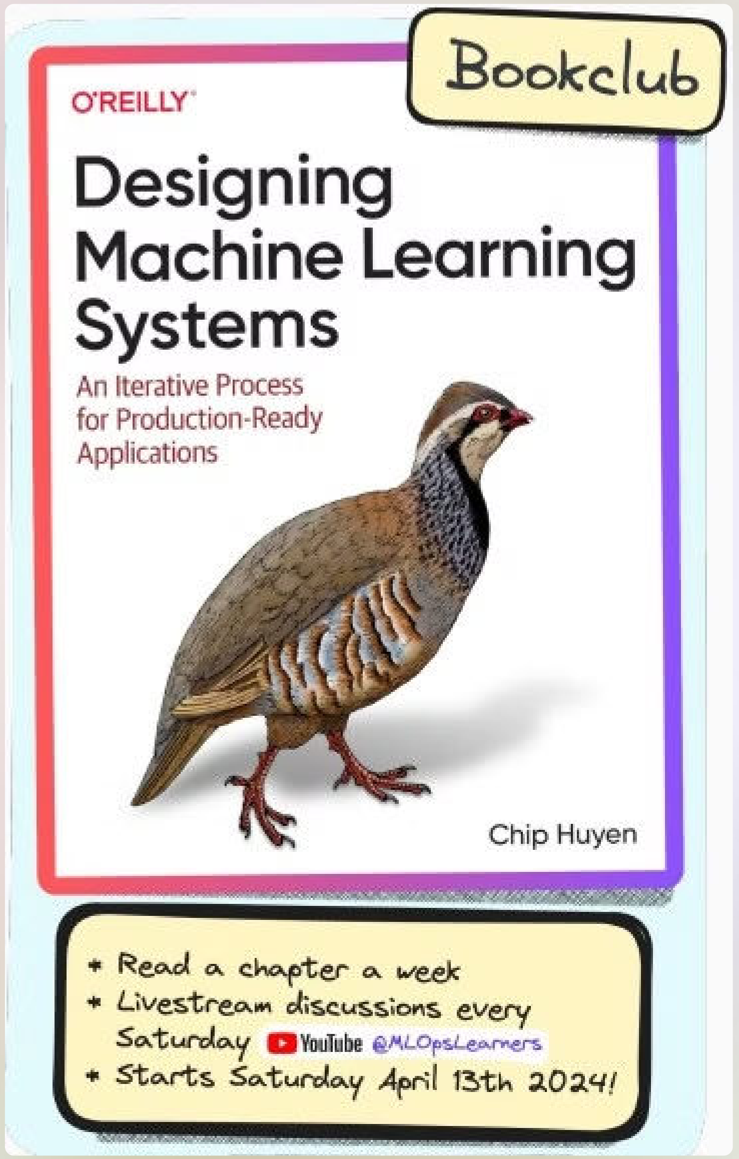

Bookclub
O'REILLY
Designing Machine Learning Systems
An Iterative Process for Production-Ready Applications
Chip Huyen
* Read a chapter a week
* Livestream discussions every Saturday YouTube @MLOpsLearners
* Starts Saturday April 13th 2024!

Chapter 7: Model Deployment

Group 9 B-Team

Saturday, May 25, 2024

YouTube

https://www.youtube.com/live/C_140NJfNtE?si=xSquXts-UXpy7p1T&t=232

Chapter 7: ML Model Deployment

Deploying machine learning models in production environments requires careful consideration of different prediction paradigms, model compression techniques, and the tradeoffs between cloud and edge computing. This section explores key factors in successfully deploying ML models at scale.

by Vern Liang

Backup Team (B-Team)

Vern Liang

https://www.linkedin.com/in/vernliang

Sr PM/SWE @ Intel, Staff Engineer @ Broadcom

BS EECS UC Berkeley, MS CS/ML Georgia Tech, MBA UIUC

Saketh Bachu

https://www.linkedin.com/in/sakethbachu

Deep Learning Researcher Mercedes-Benz

MS EE UC Riverside

Four Machine Learning Deployment Myths

Deploy Only 1-2 Models

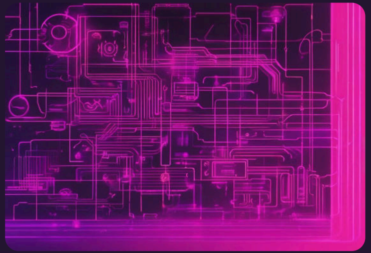
Deploying multiple models in production can improve performance and resilience.

Model Perf Same over Time

Model performance can degrade over time, requiring frequent retraining and updates.

No Need to Freq Update Models

Regularly updating models is crucial to maintain accuracy and relevance.



MLE Don't Worry about Scale

Scaling machine learning to production requires careful design and engineering.

Batch Prediction vs. Online Prediction

Batch Prediction

Predictions are generated periodically or triggered. Only uses batch features.

Online Prediction

Predictions are generated and returned real-time. These can use batch features OR streaming features (streaming prediction).

Unifying Pipelines

Combining batch and streaming approaches can leverage the strengths of both, for a flexible, scalable solution.

Pros and Cons of Batch Prediction

High Efficiency

Batch prediction can efficiently process large volumes of data, leveraging the power of distributed computing in a data center.

Scheduled Updates

Batch models can be updated on a fixed schedule, allowing for more controlled model iterations and testing.

Delayed Responses

Batch prediction can have higher latency, as data must be aggregated before processing, leading to slower response times.

Pros and Cons of Online Prediction

Real-Time Responsiveness

Online prediction can provide immediate results, enabling quick decision-making and rapid response to changing conditions.

Resource-Intensive

Handling data in real-time requires significant computing power and infrastructure, which can be costly to maintain.

Continuous Learning

Online models can be updated incrementally, allowing them to adapt to evolving data patterns and user preferences.

Potential for Instability

Fluctuations in data streams or system failures can disrupt online prediction, leading to unreliable results.

Unifying Batch and Streaming Pipeline

To leverage the strengths of both batch and online prediction, a unified pipeline can be established. This approach combines the efficiency of batch processing with the responsiveness of real-time data handling, creating a flexible and scalable solution for diverse machine learning deployments.

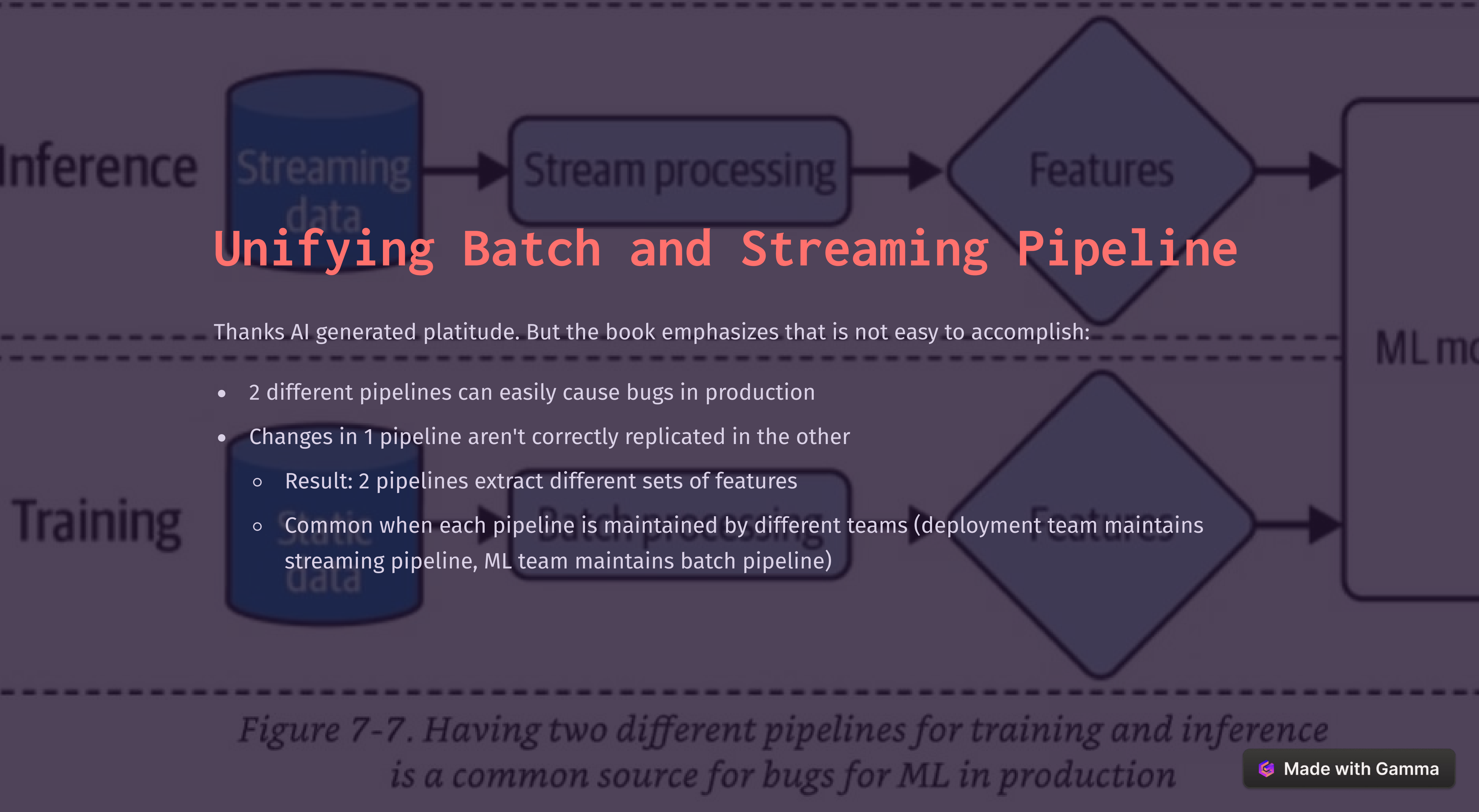


Figure 7-7. Having two different pipelines for training and inference is a common source for bugs for ML in production

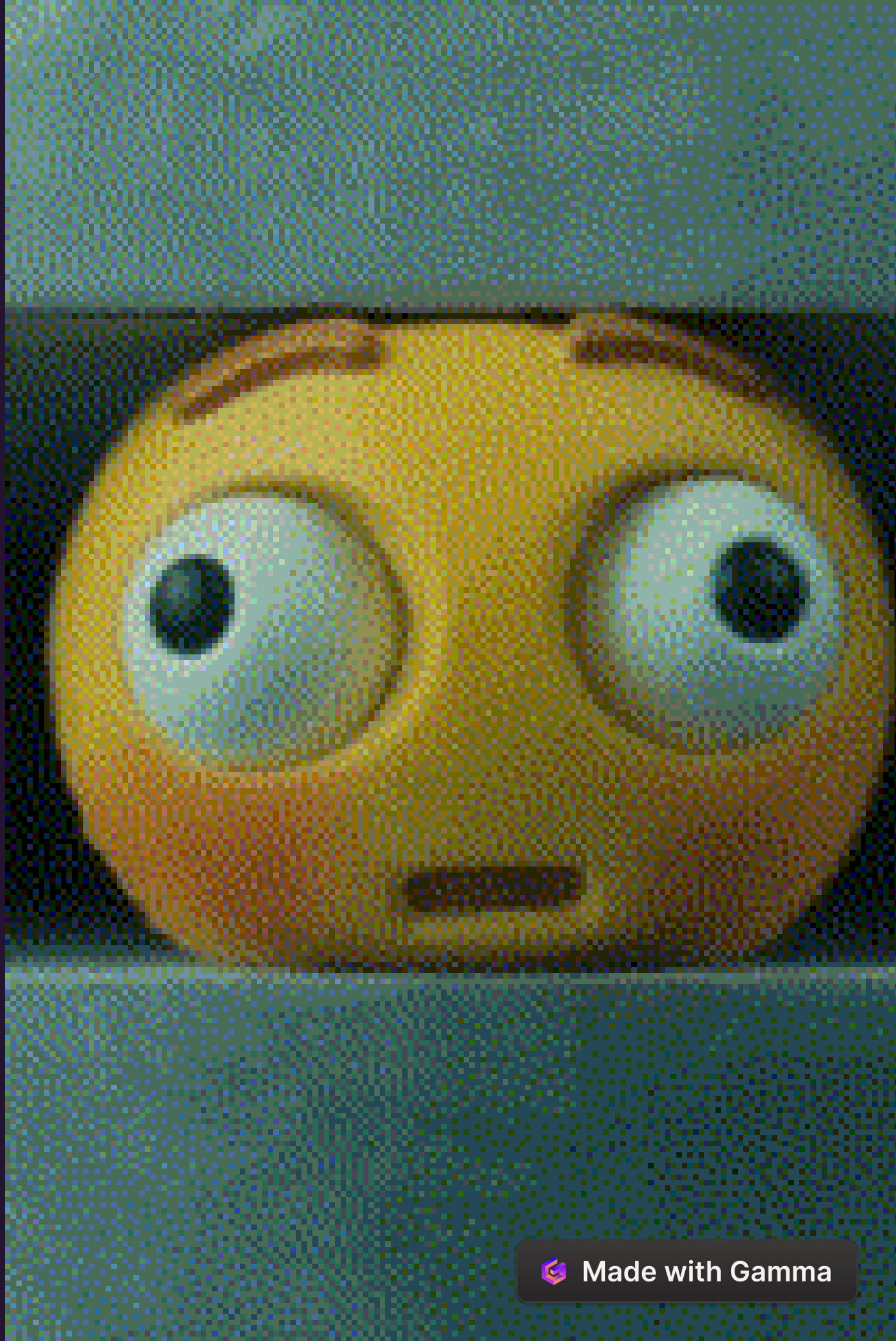
Unifying Batch and Streaming Pipeline

Thanks AI generated platitude. But the book emphasizes that is not easy to accomplish:

- 2 different pipelines can easily cause bugs in production
- Changes in 1 pipeline aren't correctly replicated in the other
  - Result: 2 pipelines extract different sets of features
  - Common when each pipeline is maintained by different teams (deployment team maintains streaming pipeline, ML team maintains batch pipeline)

Model Compression Techniques

To enable the deployment of machine learning models on resource-constrained devices, various compression techniques can be employed to reduce the model size and computational requirements without significantly sacrificing accuracy.

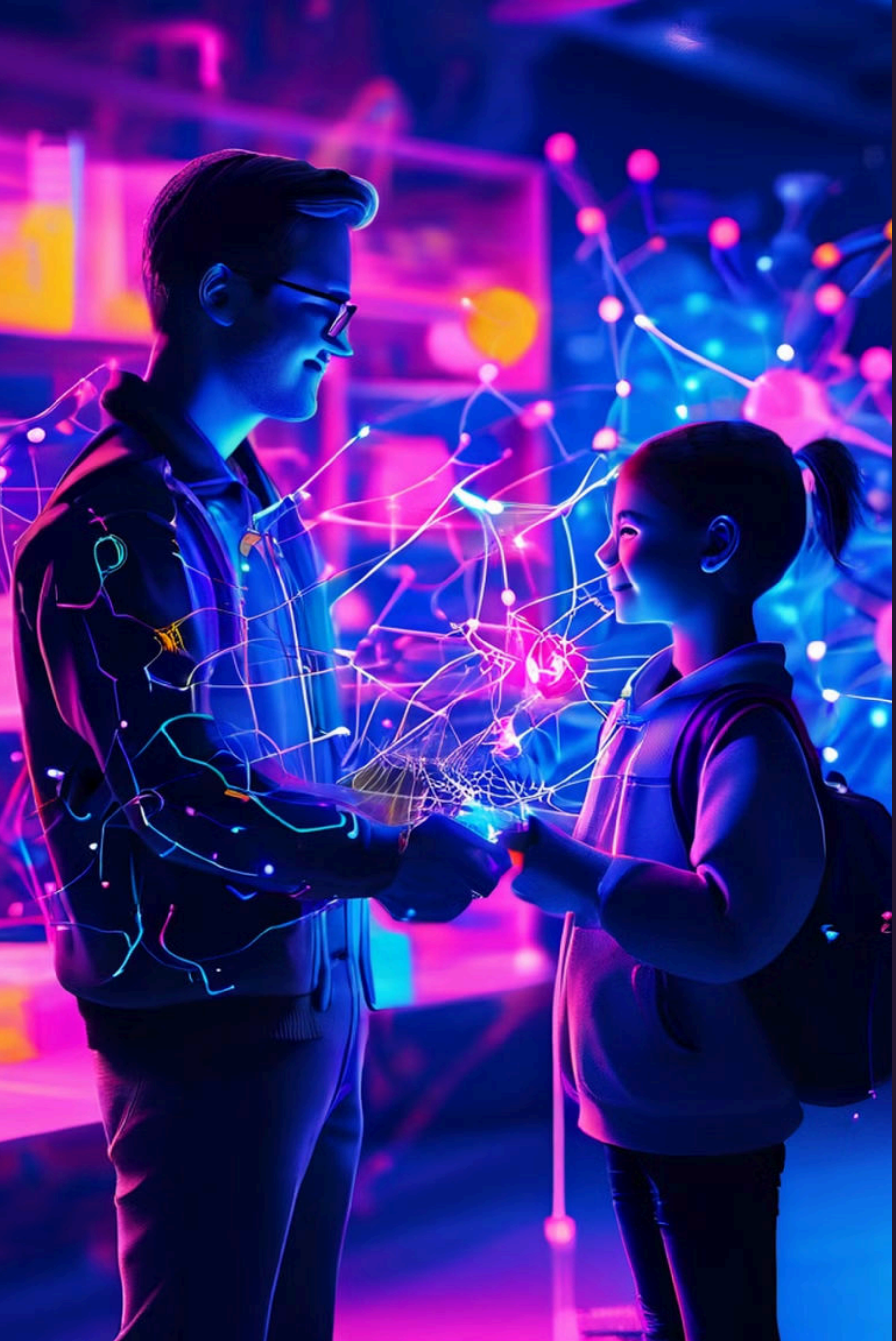


Low-rank Optimization

Low-rank optimization is a model compression technique that reduces the complexity of a neural network by approximating the weight matrices with lower-dimensional representations. This decreases the number of parameters, leading to smaller model sizes and faster inference.

E.g., SqueezeNets replaces 3×3 convolution with 1×1, achieving AlexNet-level accuracy on ImageNet with 50x fewer parameters.

By leveraging matrix factorization, low-rank optimization can effectively capture the most salient features while discarding redundant information, enabling efficient deployment on resource-constrained devices.



Knowledge Distillation

Knowledge distillation is a model compression technique that leverages a larger, more accurate "teacher" model to train a smaller "student" model.

E.g., DistilBERT (40% reduced size of BERT, 97% of language understanding, 60% faster inference time)

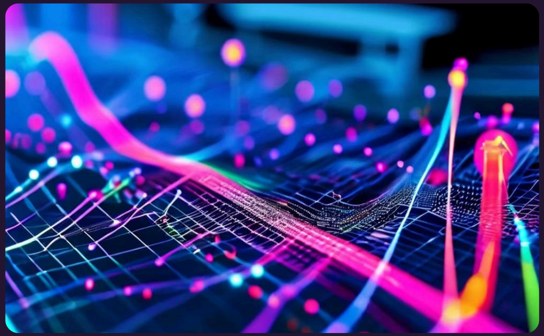
Advantages: This approach allows the student model to benefit from the teacher's expertise while being significantly more lightweight and efficient, enabling deployment on resource-constrained devices without sacrificing predictive performance. Also architecture independent (e.g., Transformer teacher, random forest student).

Disadvantages: Dependent on teacher model availability, Sensitive to applications

Pruning

Pruning (Weights)

Pruning is a model compression technique that in one form sets to 0 less important parameters in a neural network (leaving architecture intact), reducing its overall complexity (sparse neural network) without significantly impacting performance.



Pruning (Nodes)

In another form of pruning, identifying and selectively removing the least significant weights and connections, it can substantially decrease the model size (change architecture) and computational requirements, enabling efficient deployment on resource-constrained devices.

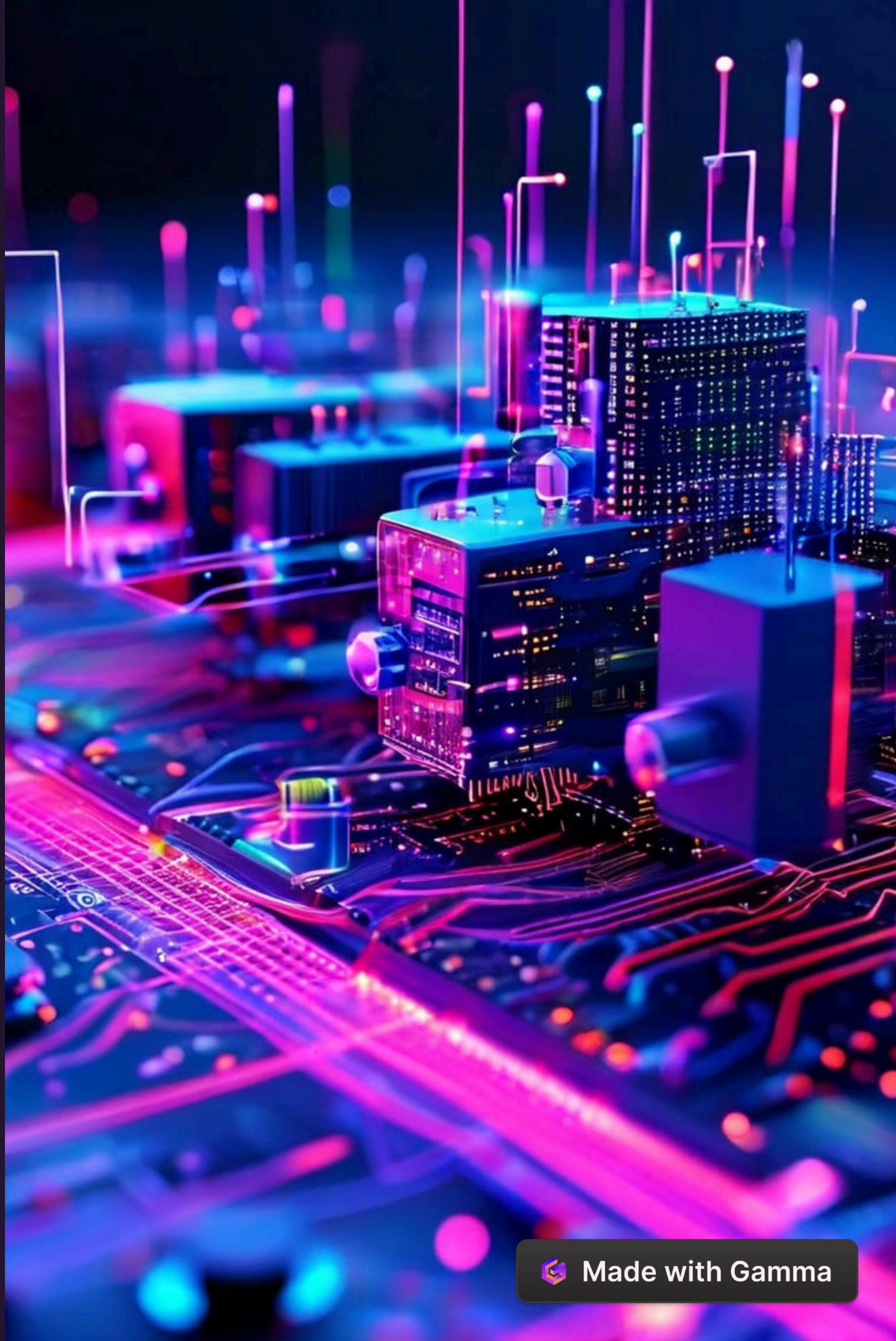
Balancing Performance

Pruning requires careful optimization to find the right balance between model size reduction and maintaining predictive accuracy. This can introduce biases into the model.

Quantization

Quantization is a model compression technique that reduces the precision of a neural network's weights and activations, converting them from floating-point to lower-bit representations, such as integers.

By leveraging reduced numerical precision, quantized models can significantly decrease memory footprint and computational requirements, enabling efficient deployment on resource-constrained devices like mobile phones and edge devices.



ML On the Cloud and Edge

Cloud Easy to Start/Scale

Managed Cloud Services (e.g., AWS, GCP, etc)

Easy to get up and running quickly and scale up/down

Edge Advantages over Server

Runs on client device, no $ to AWS or GCP

Don't worry (as much) on network latency

Don't pass sensitive user info over network

Cloud Compute Cost

Very expensive

Large orgs: Intuit, Pinterest spend $100M+/yr

SMBs: $50k to $2M/yr (easy to go broke)

[A few years ago my startup was killed by a AWS mistake that ran overnight.| Hacker News]

Edge Compute Needs Power

Needs enough battery power

Needs enough processing power to run models

ML On Edge Devices

Problems

Limited Power, Latency, Trade-off between Size and Accuracy, Heterogeneous Hardware

Conventional Solutions

(In the book): Model Compression, Efficient Design (MobileNets), Quantization, Sparsity, Distillation

New Solution - Hybrid Cloud Edge

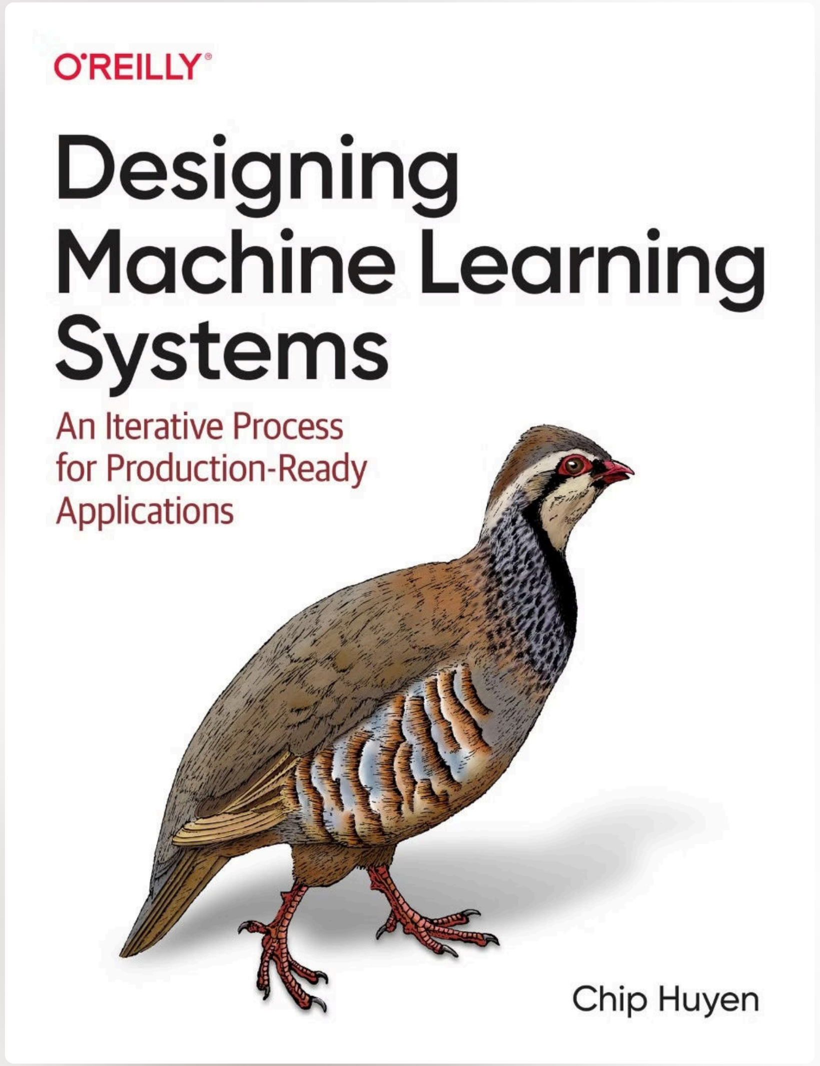
Push not so important data to cloud and pull the predictions to the edge once they are ready

New Solution - Collaborative Federated Learning

Private information on the edge and "not-so-private" information on the cloud and train models collaboratively

Transformers on Edge Devices

1. **Problem**:
   Too much computation (Dot-product Attention) → Restricted by on-device memory and battery support
2. **Exciting Solutions**:
   Reduce the quadratic complexity to linear complexity of the attention mechanism (using approximations by lowering precisions, quantizations)
3. **New Hardware**:
   SwiftTron, an open source hardware accelerator for Transformers. Port operations to integers (reduced FLOPs) and use scaling factors for corrections


O'REILLY

Designing Machine Learning Systems

An Iterative Process for Production-Ready Applications

Chip Huyen

Didn't Cover (Buy the Book!) in Chapter 7:

Compiling and Optimizing Models for Edge Devices

Model Optimization for Edge Devices

Using ML to Optimize ML Models

Running ML Models in Internet Browsers

https://huyenchip.com/books (also free online book for ML Job Interviews)

https://tomslist.com (Medium Post covering GenAI Presentation Slide Tools)

Made with Gamma

Discussion Questions

1. What strategies can be employed to reduce the energy consumption of transformer-based applications on edge devices?
2. What edge devices are currently used in the industry for deploying ML models and why?
3. What experiences do you have in deploying a model in production? Industry/Personal Projects (with AWS Sagemaker, GCP Vertex AI, Azure ML)?
4. Can you share any professional experiences with model compression?
5. Can you share any professional experiences with batch and streaming pipelines?