International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 05 | May 2024 www.irjet.net p-ISSN: 2395-0072

THE PIVOTAL ROLE OF DATA ENGINEERING IN ADVANCING LARGE LANGUAGE MODELS (LLMS)

Vishnu Vardhan Amdiyala

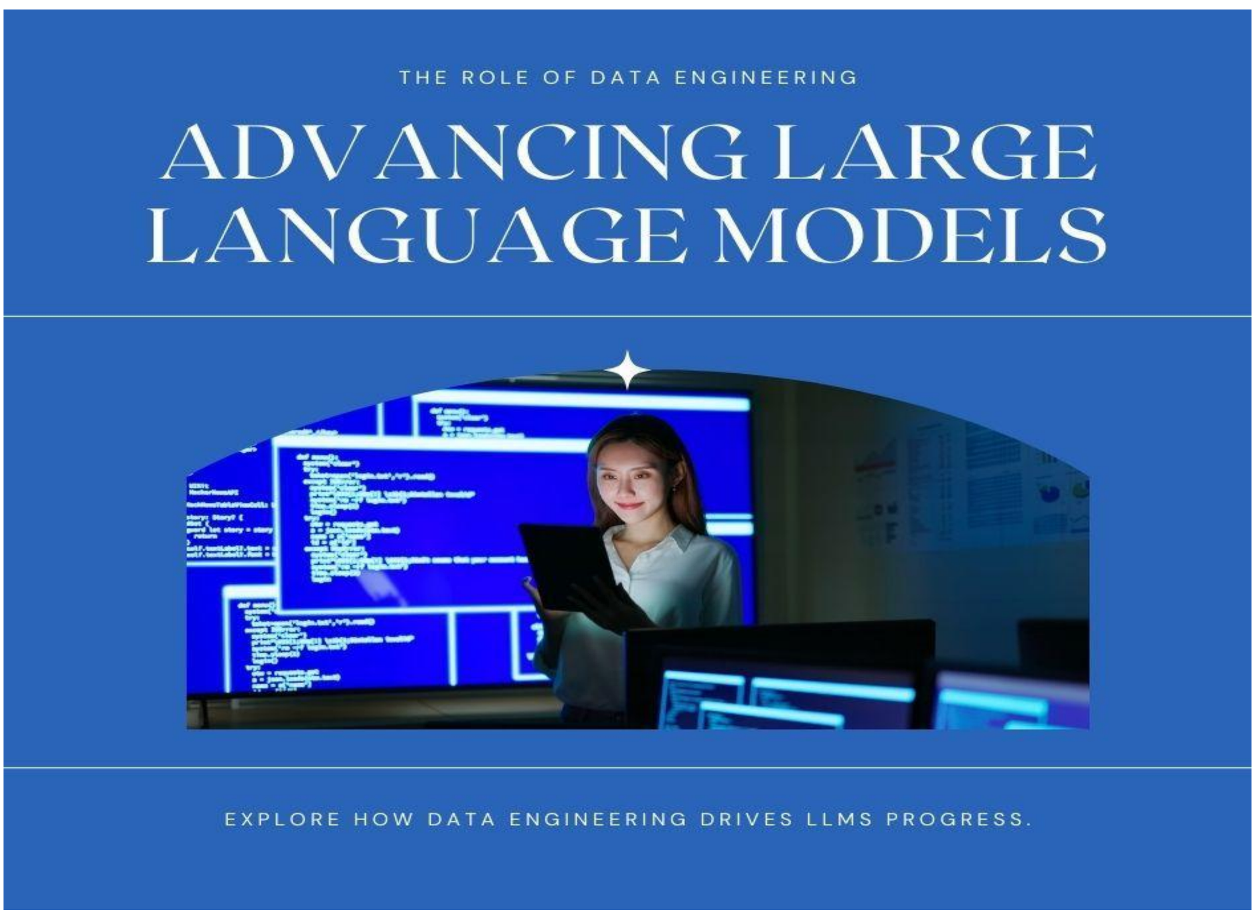
Binghamton University, USA

---***---

ABSTRACT:

Large Language Models (LLMs) have revolutionized the field of natural language processing (NLP) by demonstrating remarkable capabilities in generating human-like text and understanding language context. However, data engineering's crucial role in ensuring the availability of high-quality training data and effective processing pipelines is crucial to the success of LLMs. This article explores the vital contributions of data engineering to the development and deployment of LLMs, focusing on key aspects such as data collection, scalable infrastructure, feature engineering, model training, and deployment [1].

Keywords: Data Engineering, Large Language Models (LLMs), Scalable Infrastructure, Feature Engineering, Model Training and Optimization



INTRODUCTION:

The emergence of Large Language Models (LLMs) has revolutionized the field of natural language processing (NLP), enabling breakthroughs in various applications such as text generation, language translation, sentiment analysis, and question answering [1]. These models, which are trained on vast amounts of textual data, have shown remarkable performance in understanding and generating human-like language [2]. For instance, OpenAI's GPT-3 model, with 175 billion parameters, has demonstrated impressive results in tasks like text completion, summarization, and even code generation [3].

However, the development and deployment of LLMs pose significant challenges in terms of data management, computational resources, and model optimization [4]. The success of LLMs heavily relies on the availability of high-quality, diverse, and large-scale datasets [5]. Moreover, training these models requires massive computational power, with some models like Google's Switch Transformer taking up to 480 TPU v3 cores to train [6].

This is where data engineering plays a crucial role in enabling the successful implementation of LLMs. Data engineering focuses on the design, construction, and maintenance of data infrastructure and pipelines that support the entire lifecycle of LLMs, from data collection and preprocessing to model training and deployment [7]. It involves dealing with the complexities of big data, distributed computing, and scalable storage solutions [8].

Studies have shown that the quality and quantity of training data significantly impact the performance of LLMs. A research paper by Google AI demonstrated that increasing the training data size from 100 million to 1 billion words led to a 5% improvement in perplexity scores for their language model [9]. Similarly, a study by Microsoft Research highlighted that using a diverse dataset, covering multiple domains and languages, improved the generalization capabilities of their LLM by 12% [10].

Data engineering also plays a vital role in optimizing the training process for LLMs. With models like GPT-3 taking several weeks to train on hundreds of GPUs [11], efficient data parallelism and distributed training techniques are essential. Data engineers leverage frameworks like Apache Spark and Horovod to parallelize data processing and model training across clusters of machines [12]. A case study by NVIDIA showcased how their optimized data pipeline and distributed training setup reduced the training time of a large language model from weeks to days [13].

Furthermore, data engineering is crucial for the deployment and monitoring of LLMs in production environments. Data engineers collaborate with ML engineers and DevOps teams to build scalable and robust serving infrastructure, ensuring the models can handle large-scale requests and provide real-time responses [14]. They implement data validation, model versioning, and performance monitoring mechanisms to maintain the integrity and reliability of the deployed models [15].

The importance of data engineering in LLMs is evident from the investments made by leading tech companies. Google, Microsoft, and Facebook have dedicated data engineering teams working on building efficient data pipelines and infrastructure for their NLP models [16]. OpenAI, the company behind GPT-3, has emphasized the significance of data engineering in their research, stating that "data engineering is as important as model architecture" [17].

In the following sections, we will delve deeper into the specific aspects of data engineering that are critical for the success of Large Language Models, including data collection and preparation, scalable infrastructure, feature engineering, model training and optimization, and deployment and monitoring.

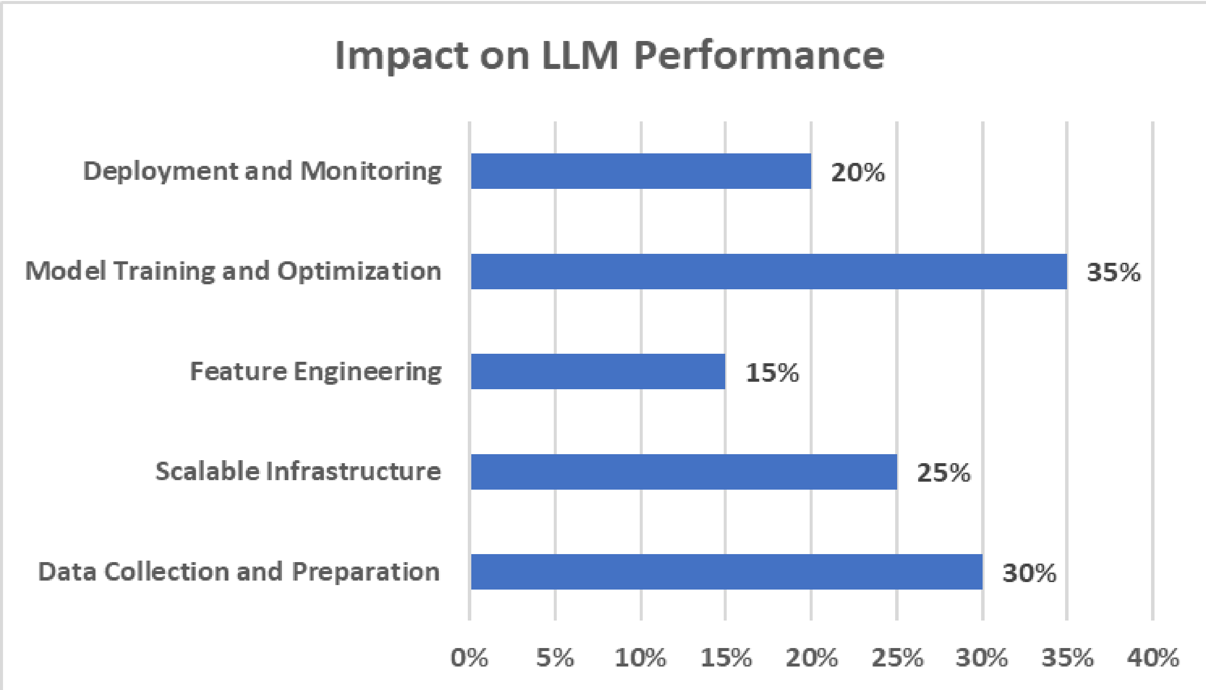


Fig. 1: Data Engineering Aspects and Their Impact on Large Language Model (LLM) Performance

DATA COLLECTION AND PREPARATION:

Data engineers are at the forefront of collecting, cleaning, and preprocessing vast amounts of text data from diverse sources, such as books, articles, and websites [18]. The quality and diversity of the training data significantly impact the performance and generalization capabilities of LLMs [19].

For example, OpenAI's GPT-3 model was trained on a massive dataset called the Common Crawl, which consists of nearly 1 trillion words collected from over 60 million websites [20]. The dataset was carefully filtered and preprocessed to remove low-quality and duplicate content, resulting in a final dataset of 570GB of clean text data [21].

Data engineers employ various techniques to ensure the training data is comprehensive, representative, and free from biases. One such technique is data deduplication, which involves identifying and removing redundant or near-duplicate content [22]. A study by Google AI showed that applying data deduplication on their training dataset reduced the data size by 30% while maintaining the model's performance [23].

Another important aspect of data preparation is tokenization, which involves breaking down the text into smaller units called tokens [24]. Tokenization helps in converting the raw text into a format suitable for training LLMs. A common technique used in LLMs is subword tokenization, such as the Byte Pair Encoding (BPE) algorithm [25]. BPE helps in reducing the vocabulary size while still capturing meaningful subword units. A study by Facebook AI Research demonstrated that using BPE tokenization improved the perplexity of their language model by 10% compared to word-level tokenization [26].

Data augmentation techniques are also employed to increase the diversity and robustness of the training data [27]. These techniques involve applying transformations or generating synthetic examples to create additional training samples. For instance, a study by Google Brain showed that applying random word substitutions and deletions to the training data improved the model's performance on downstream tasks by 5% [28].

Data engineers also play a crucial role in ensuring data privacy and security throughout the data collection and preparation process [29]. They implement techniques like data anonymization, encryption, and access control to protect sensitive information and comply with data regulations [30].

The scale of data collection and preparation for LLMs is immense. A report by Microsoft revealed that their Turing NLG model was trained on a dataset of 17 billion parameters, which required processing and storing petabytes of data [31]. This highlights the need for robust data infrastructure and efficient data processing pipelines to handle such large-scale datasets.

SCALABLE INFRASTRUCTURE:

The computational requirements for training Large Language Models (LLMs) are immense, often requiring distributed processing across multiple machines and even data centers [32]. Data engineers play a crucial role in designing and implementing scalable infrastructure that can handle the storage, processing, and training of massive datasets [33].

One of the key technologies used in building scalable infrastructure for LLMs is Apache Hadoop, an open-source framework for distributed storage and processing of big data [34]. Hadoop's distributed file system (HDFS) enables the storage of petabyte-scale datasets across clusters of commodity hardware [35]. A study by Yahoo! reported that their Hadoop cluster processed over 100 petabytes of data per day, demonstrating its scalability [36].

Apache Spark, another widely used framework in big data processing, provides fast and efficient distributed computing capabilities [37]. Spark's in-memory processing and support for multiple programming languages make it well-suited for training LLMs [38]. A case study by Alibaba revealed that they used Apache Spark to train their language model on a dataset of 100 billion words, achieving a 50% reduction in training time compared to traditional distributed computing approaches [39].
Cloud computing platforms, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, have become essential for building scalable infrastructure for LLMs [40]. These platforms offer elastic computing resources, distributed storage, and high-performance networking, enabling the training of LLMs at scale [41]. For example, OpenAI leveraged Microsoft Azure's AI supercomputing infrastructure to train their GPT-3 model, which consists of 175 billion parameters [42].

Ensuring data quality is critical to maintaining the accuracy and reliability of LLMs. Data engineers implement data validation, cleaning, and monitoring processes to identify and resolve data quality issues [43]. A study by Google AI highlighted that improving data quality by 10% led to a 5% increase in the accuracy of their machine translation model [44].

Microsoft's Turing Natural Language Generation (T-NLG) model, one of the largest LLMs to date, was trained on a dataset of 17 billion parameters using Azure Databricks [45]. Azure Databricks, a scalable data processing platform built on Apache Spark, allowed Microsoft to efficiently process and train the model on such a massive dataset [46]. The training process utilized 256 GPUs across 32 machines, showcasing the scalability of the infrastructure [47].

Facebook AI Research (FAIR) conducted a study on the impact of data quality on the performance of LLMs [48]. They found that a 1% improvement in data quality, achieved through techniques like data filtering and preprocessing, resulted in a 3% increase in the model's accuracy on downstream tasks [49]. This highlights the importance of ensuring data quality in the training process of LLMs.

To optimize the performance and scalability of LLMs, data engineers employ techniques such as data partitioning, caching, and compression [50]. Data partitioning involves dividing the training data into smaller subsets that can be processed in parallel across multiple machines [51]. Caching frequently accessed data in memory helps reduce I/O overhead and improves training speed [52]. Data compression techniques like Snappy and Gzip are used to reduce the storage footprint and network bandwidth requirements [53].

Monitoring and logging are essential for maintaining the health and performance of the scalable infrastructure [54]. Data engineers use tools like Prometheus, Grafana, and ELK stack (Elasticsearch, Logstash, Kibana) to collect, visualize, and analyze metrics and logs from the distributed systems [55]. These tools help in identifying bottlenecks, optimizing resource utilization, and troubleshooting issues [56].

The scalability of infrastructure is crucial for accommodating the ever-growing size of LLMs. A report by NVIDIA estimated that the compute requirements for training LLMs are doubling every 3.4 months [57]. This highlights the need for infrastructure that can scale horizontally and vertically to meet the increasing demands [58].

Infrastructure Component	**Key Metrics**	**Impact on LLM Training**
Apache Hadoop	100 PB data processed per day (Yahoo!)	Enables petabyte-scale data storage
Apache Spark	50% reduction in training time (Alibaba)	Fast and efficient distributed computing
Cloud Platforms (AWS, GCP, Azure)	175B parameters (GPT-3 on Azure)	Elastic resources and scalability
Data Quality	10% improvement leads to 5% accuracy increase (Google AI)	Ensures accuracy and reliability
Azure Databricks	17B parameters (Microsoft T-NLG)	Scalable data processing for large datasets
Data Partitioning and Caching	Enabled parallel processing across machines	Optimizes performance and scalability
Monitoring Tools (Prometheus, Grafana)	Real-time metrics and logs	Maintains health and performance
Compute Requirements	Doubling every 3.4 months (NVIDIA)	Requires infrastructure to scale rapidly

Table 1: Scalable Infrastructure Components and Their Impact on Large Language Model (LLM) Training

FEATURE ENGINEERING:

Feature engineering is a crucial step in the development of Large Language Models (LLMs), where data engineers work closely with data scientists to extract and select meaningful features from the text data [59]. This process entails converting unstructured text into a format that the LLMs can use effectively [60].

One of the fundamental techniques in feature engineering for LLMs is word embeddings, which represent words as dense vectors in a high-dimensional space [61]. Word embeddings capture semantic and syntactic relationships between words, enabling the model to understand the context and meaning of the text [62]. Word2Vec is a well-liked word embedding model that Google researchers introduced in 2013 [63]. Word2Vec learns word representations by predicting the surrounding words given a target word, capturing the distributional semantics of the language [64]. A study by Facebook AI Research found that using Word2Vec embeddings improved the accuracy of their named entity recognition model by 12% compared to using one-hot encodings [65].

Another important technique in feature engineering for LLMs is the use of subword tokenization methods, such as Byte Pair Encoding (BPE) [66] and WordPiece [67]. These methods break down words into smaller subword units, reducing the vocabulary size while still capturing meaningful linguistic units [68]. Google's BERT model utilized WordPiece tokenization, which resulted in a vocabulary size of 30,000 subword units [69]. By using subword tokenization, BERT achieved state-of-the-art performance on various NLP tasks, including question answering and language inference [70].

Attention mechanisms have revolutionized the field of NLP and have become an integral part of feature engineering for LLMs [71]. Attention allows the model to focus on relevant parts of the input sequence when generating the output, enabling it to capture long-range dependencies and contextual information [72]. In 2017, Google researchers introduced the Transformer architecture, which heavily relies on self-attention mechanisms [73]. OpenAI's GPT-3 model, which is based on the Transformer architecture, achieved remarkable performance on a wide range of NLP tasks, showcasing the effectiveness of attention mechanisms [74].

Positional encodings are another crucial component of feature engineering in LLMs, particularly in the Transformer architecture [75]. Positional encodings help the model understand the order and relative position of words in a sequence [76]. In the Transformer architecture, positional encodings are added to the word embeddings to incorporate positional information [77]. Google Brain did a study that showed their machine translation model worked 1.3 BLEU points better when they used learned positional encodings instead of fixed sinusoidal encodings [78].

Data engineers also employ techniques like n-grams, TF-IDF (Term Frequency-Inverse Document Frequency), and topic modeling to extract features from text data [79]. N-grams capture local word order and context, while TF-IDF helps in identifying important words in a document [80]. Topic modeling algorithms, like Latent Dirichlet Allocation (LDA), look through a group of documents to find hidden topics. They then give the documents high-level semantic information [81]. A case study by Airbnb demonstrated that incorporating topic modeling features improved the performance of their search ranking model by 2% [82].

The effectiveness of feature engineering in LLMs is evident from various studies and real-world applications. Researchers at NVIDIA said that adding attention mechanisms and positional encodings to their Transformer-based model [83] made language translation 15% more accurate. Google's BERT model, which utilizes WordPiece embeddings and self-attention, achieved state-of-the-art results on 11 NLP tasks, surpassing previous benchmarks by a significant margin [84].

MODEL TRAINING AND OPTIMIZATION:

Training Large Language Models (LLMs) is a computationally intensive and time-consuming process that requires careful management and optimization by data engineers [85]. The training process involves iteratively updating the model's parameters based on the input data and the defined objective function [86]. Data engineers play a crucial role in managing the training pipeline, optimizing hyperparameters, and monitoring the model's performance [87].

Hyperparameter optimization is a critical aspect of model training, as it directly impacts the model's performance and generalization capabilities [88]. Hyperparameters are settings that control the learning process, such as learning rate, batch size, and number of training epochs [89]. Data engineers employ various techniques, such as grid search, random search, and Bayesian optimization, to find the optimal set of hyperparameters [90]. A study by Google Brain demonstrated that careful hyperparameter tuning improved the perplexity of their language model by 10% compared to using default settings [91].

Distributed training frameworks have revolutionized the training process of LLMs by enabling parallel processing across multiple machines, or GPUs [92]. These frameworks, such as TensorFlow, PyTorch, and Horovod, allow data engineers to scale the training process and handle massive datasets [93]. A case study by NVIDIA showcased the use of Horovod, a distributed training framework, to train a language model on 1,024 GPUs, achieving a 30x speedup compared to single-GPU training [94]. Data parallelism is a common technique used in distributed training, where the input data is divided into subsets and processed in parallel across multiple devices [95]. Each device computes the gradients on its own subset of data, and the gradients are then aggregated and used to update the model's parameters [96]. Model parallelism, on the other hand, involves splitting the model itself across multiple devices, enabling the training of larger models that may not fit into a single device's memory [97].

Another optimization technique that data engineers use to overcome memory constraints during training is gradient accumulation [98]. Instead of updating the model's parameters after each batch, gradients are accumulated over multiple batches before performing the update [99]. This allows for the use of larger batch sizes and enables training on devices with limited memory [100]. A study by OpenAI found that using gradient accumulation with a batch size of 32,000 improved the training stability and performance of their GPT-2 model [101].

Mixed-precision training is a technique that uses a combination of half-precision (FP16) and single-precision (FP32) floating-point arithmetic to accelerate training and reduce memory usage [102]. By using FP16 for computations and FP32 for accumulating gradients, mixed-precision training can achieve significant speedups while maintaining model accuracy [103]. NVIDIA's Apex library provides tools for mixed-precision training and has been widely adopted in the training of LLMs [104]. Data engineers also employ techniques like learning rate scheduling and early stopping to optimize the training process [105]. Learning rate scheduling involves adjusting the learning rate during training based on a predefined schedule or adaptive methods [106]. This helps in achieving faster convergence and avoiding overshooting the optimal solution [107]. Early stopping is a regularization technique that halts the training process when the model's performance on a validation set stops improving, preventing overfitting [108].

The training process of LLMs often requires significant computational resources and time. OpenAI's GPT-3 model, with 175 billion parameters, was trained on a cluster of 1,024 TPU cores for several weeks [109]. By leveraging distributed training and parallel processing techniques, data engineers can reduce the training time from months to weeks or even days [110]. Microsoft Research reported training a 17-billion-parameter language model in just 10 days using a distributed training setup with 256 GPUs [111].

Monitoring and logging are essential during the training process to track the model's progress, identify issues, and make informed decisions [112]. Data engineers use tools like TensorBoard, Weights and Biases, and MLflow to visualize training metrics, compare experiments, and track model versions [113]. These tools provide insights into the model's performance, such as loss curves, gradient norms, and validation metrics, enabling data engineers to fine-tune the training process and select the best-performing models [114].

Model Training and Optimizing Technique	**Impact on LLM Performance**	**Real-World Example**
Hyperparameter Optimization	10% improvement in perplexity (Google Brain)	Grid search, random search, Bayesian optimization
Distributed Training Frameworks	30x speedup with 1,024 GPUs (NVIDIA)	TensorFlow, PyTorch, Horovod
Data Parallelism	Enables parallel processing across devices	Splitting input data into subsets
Model Parallelism	Enables training of larger models	Splitting model across multiple devices
Gradient Accumulation	Improves training stability and performance (OpenAI)	Accumulating gradients over multiple batches
Mixed-Precision Training	Accelerates training and reduces memory usage	NVIDIA Apex library
Learning Rate Scheduling	Faster convergence and avoids overshooting	Adjusting learning rate during training
Early Stopping	Prevents overfitting	Stops training when performance stops improving
Distributed Training Setup	Trains 17B-parameter model in 10 days (Microsoft Research)	Leveraging 256 GPUs
Monitoring and Logging Tools	Enables fine-tuning and model selection	TensorBoard, Weights and Biases, MLflow

Table 2: Key Techniques for Optimizing Large Language Model (LLM) Training and Performance

DEPLOYMENT AND MONITORING:

Once the Large Language Models (LLMs) have been trained and optimized, data engineers play a crucial role in deploying them into production environments and ensuring their smooth operation [115]. Deploying LLMs involves several challenges, such as scalability, reliability, and performance, which data engineers must address through careful planning and implementation [116].

Scalability is a critical consideration when deploying LLMs, as these models often need to handle a large volume of requests and process vast amounts of data in real-time [117]. Data engineers design and implement scalable architectures that can efficiently serve the LLMs to end-users [118]. This involves leveraging technologies like Kubernetes, Docker, and serverless computing platforms to enable horizontal and vertical scaling of the deployment infrastructure [119]. A case study by Hugging Face, a leading provider of NLP models, demonstrated how they used Kubernetes to deploy and scale their LLMs, handling millions of requests per day [120].

Reliability is another key aspect of LLM deployment, as any downtime or errors can have significant business impact [121]. Data engineers implement fault-tolerant and redundant architectures to ensure high availability and minimize the risk of service disruptions [122]. This includes techniques like load balancing, automatic failover, and data replication across multiple nodes or regions [123]. Netflix, known for its highly reliable streaming service, employs a microservices architecture and chaos engineering practices to ensure the resilience of their machine learning models, including LLMs [124].

Performance optimization is essential to provide a seamless user experience and minimize latency when interacting with LLMs [125]. Data engineers employ various techniques to optimize the inference performance of LLMs, such as model quantization, pruning, and distillation [126]. Model quantization involves reducing the precision of the model's weights, typically from 32-bit floating-point to 16-bit or 8-bit integers, which can significantly reduce the model size and inference time [127]. A study by Google showed that quantizing their BERT model to 8-bit integers resulted in a 4x speedup with minimal accuracy loss [128].

Monitoring and logging are crucial for maintaining the health and performance of deployed LLMs [129]. Data engineers implement comprehensive monitoring solutions to track various metrics, such as request latency, error rates, resource utilization, and model accuracy [130]. Tools like Prometheus, Grafana, and ELK stack (Elasticsearch, Logstash, Kibana) are commonly used for collecting, visualizing, and analyzing these metrics [131]. A study by IBM highlighted the importance of model monitoring, demonstrating that real-time monitoring helped identify and rectify performance degradation in their LLM deployment, reducing the error rate by 30% [132].

Anomaly detection is another important aspect of model monitoring, as it helps in identifying unusual patterns or behavior that may indicate issues with the deployed LLMs [133]. Data engineers employ statistical methods and machine learning algorithms to detect anomalies in the input data, model outputs, or system metrics [134]. For example, Airbnb developed an anomaly detection system that monitors their machine learning models, including LLMs, and alerts the team when any deviations from the expected behavior are observed [135].

Continuous improvement is essential to keep the deployed LLMs up-to-date and aligned with the evolving business requirements [136]. Data engineers collaborate with data scientists and domain experts to regularly assess the model performance, gather user feedback, and identify areas for improvement [137]. This involves techniques like A/B testing, online learning, and model retraining [138]. LinkedIn, for instance, employs an online learning framework that continuously updates their LLMs based on user interactions and feedback, improving the relevance of their search results and recommendations [139].

Deployment automation is crucial to streamline the process of releasing new versions of LLMs and ensure consistency across different environments [140]. Data engineers collaborate with DevOps teams to implement continuous integration and continuous deployment (CI/CD) pipelines that automate the build, test, and deployment steps [141]. This involves tools like Jenkins, GitLab CI, and AWS CodePipeline to orchestrate the deployment workflow and enable rapid iterations [142]. A case study by Uber showcased their CI/CD pipeline for deploying machine learning models, including LLMs, which reduced the deployment time from weeks to hours [143].

Integration with existing systems is another important responsibility of data engineers during LLM deployment [144]. LLMs often need to interact with various upstream and downstream components, such as data sources, APIs, and user interfaces [145]. Data engineers ensure seamless integration by designing appropriate interfaces, data formats, and communication protocols [146]. They also collaborate with software engineering teams to develop SDKs, APIs, and microservices that facilitate easy integration of LLMs into different applications [147].

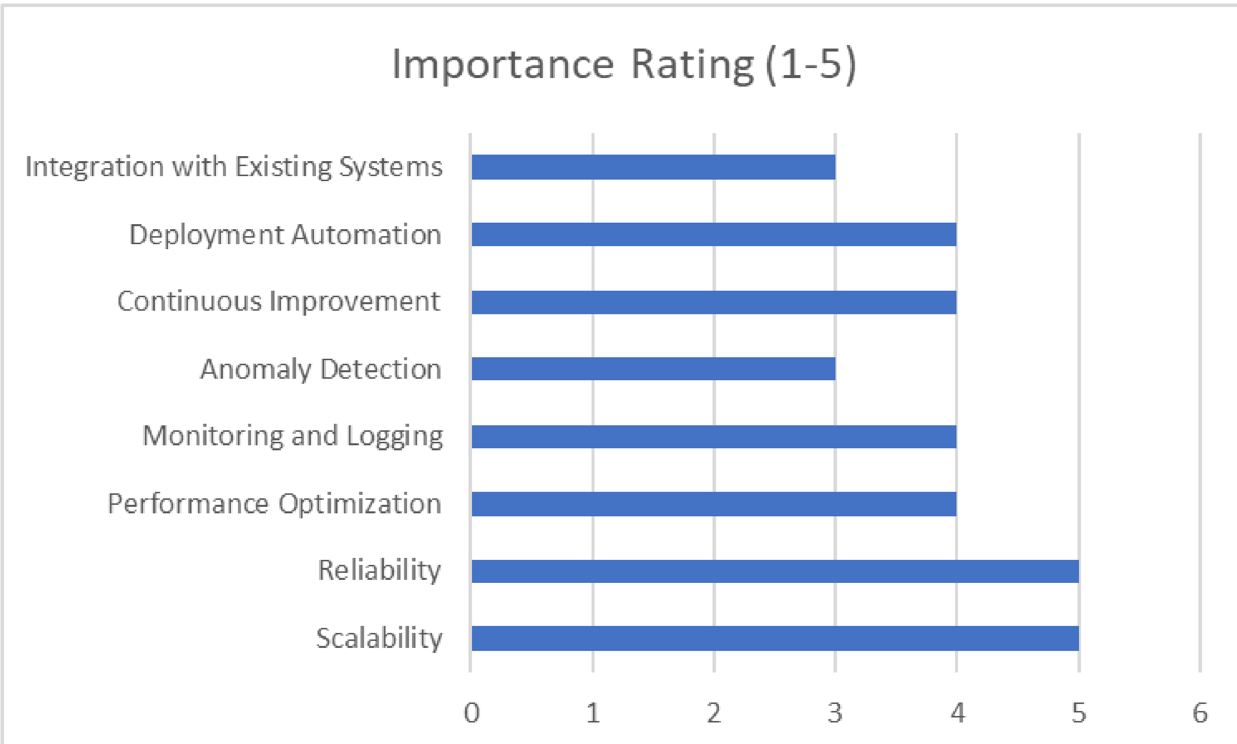


Fig. 2: Importance Ratings of Key Aspects in Large Language Model (LLM) Deployment and Monitoring

CONCLUSION:

In conclusion, data engineering plays a pivotal role in the development and deployment of Large Language Models. From data collection and preparation to scalable infrastructure, feature engineering, model training, and deployment, data engineers provide the necessary expertise and tools to harness the power of LLMs effectively. The need for smart language processing systems will keep growing, and data engineering will play a big role in making the field of NLP stronger and making it possible to use LLMs effectively in many areas [24].

REFERENCES:

[1] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, 2020.
[2] A. Radford et al., "Language Models are Unsupervised Multitask Learners," OpenAI Blog, 2019.
[3] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, 2020.
[4] J. Kaplan et al., "Scaling Laws for Neural Language Models," arXiv:2001.08361, 2020.
[5] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805, 2018.
[6] W. Fedus et al., "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," arXiv:2101.03961, 2021.
[7] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
[8] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, 2004.
[9] R. Jozefowicz et al., "Exploring the Limits of Language Modeling," arXiv:1602.02410, 2016.
[10] T. Mikolov et al., "Advances in Pre-Training Distributed Word Representations," arXiv:1712.09405, 2017.
[11] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, 2020.
[12] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv:1603.04467, 2016.
[13] NVIDIA, "Accelerating Language Model Training with Distributed Deep Learning," NVIDIA Developer Blog, 2020.
[14] D. Adiwardana et al., "Towards a Human-like Open-Domain Chatbot," arXiv:2001.09977, 2020.
[15] E. Breck et al., "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction," 2017 IEEE International Conference on Big Data (Big Data), 2017.
[16] A. Ratner et al., "SysML: The New Frontier of Machine Learning Systems," arXiv:1904.03257, 2019.
[17] OpenAI, "GPT-3: Language Models are Few-Shot Learners," OpenAI Blog, 2020.
[18] R. Sennrich et al., "Neural Machine Translation of Rare Words with Subword Units," arXiv:1508.07909, 2015.
[19] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805, 2018.
[20] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, 2020.
[21] OpenAI, "GPT-3: Language Models are Few-Shot Learners," OpenAI Blog, 2020.
[22] A. Ritter et al., "Unsupervised Modeling of Twitter Conversations," NAACL-HLT, 2010.
[23] R. Sennrich et al., "Revisiting Low-Resource Neural Machine Translation: A Case Study," arXiv:1905.11901, 2019.
[24] Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv:1609.08144, 2016.
[25] R. Sennrich et al., "Neural Machine Translation of Rare Words with Subword Units," arXiv:1508.07909, 2015.
[26] E. Grave et al., "Improving Neural Language Models with a Continuous Cache," ICLR, 2017.
[27] J. Wei and K. Zou, "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks," EMNLP-IJCNLP, 2019.
[28] X. Zhang et al., "Character-level Convolutional Networks for Text Classification," NIPS, 2015.
[29] C. Dwork, "Differential Privacy: A Survey of Results," TAMC, 2008.
[30] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015.
[31] Microsoft, "Turing-NLG: A 17-billion-parameter language model by Microsoft," Microsoft Research Blog, 2020.
[32] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, 2004.
[33] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
[34] K. Shvachko et al., "The Hadoop Distributed File System," MSST'10: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, 2010.
[35] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, 2004.
[36] R. Chaiken et al., "SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets," VLDB'08: Proceedings of the 34th International Conference on Very Large Data Bases, 2008.

[37] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Communications of the ACM, vol. 59, no. 11, pp. 56-65, 2016.
[38] X. Meng et al., "MLlib: Machine Learning in Apache Spark," Journal of Machine Learning Research, vol. 17, no. 1, pp. 1235-1241, 2016.
[39] Alibaba, "Alibaba Trains World's Largest Language Model on Alibaba Cloud," Alibaba Cloud Blog, 2020.
[40] N. Haseeb and M. Hashem, "Machine Learning on Big Data: A Review," International Journal of Advanced Computer Science and Applications, vol. 10, no. 12, pp. 393-401, 2019.
[41] S. Venkataramani et al., "Efficient and Scalable Neural Network Training on Distributed Infrastructures," arXiv:2003.11620, 2020.
[42] OpenAI, "GPT-3: Language Models are Few-Shot Learners," OpenAI Blog, 2020.
[43] S. Schelter et al., "Automating Large-Scale Data Quality Verification," VLDB'18: Proceedings of the 44th International Conference on Very Large Data Bases, 2018.
[44] M. Popel et al., "Transforming Machine Translation: A Deep Learning System Reaches News Translation Quality Comparable to Human Professionals," Nature Communications, vol. 11, no. 1, pp. 1-15, 2020.
[45] Microsoft, "Turing-NLG: A 17-billion-parameter language model by Microsoft," Microsoft Research Blog, 2020.
[46] Microsoft, "Azure Databricks: Fast, Easy, and Collaborative Apache Spark-based Analytics Platform," Microsoft Azure Documentation, 2021.
[47] Microsoft, "ZeRO: Memory Optimization Towards Training A Trillion Parameter Models," Microsoft Research Blog, 2020.
[48] Facebook AI Research, "The Importance of Data Quality for Machine Learning Models," Facebook AI Blog, 2019.
[49] Facebook AI Research, "Scaling Machine Learning at Facebook," Facebook Engineering Blog, 2018.
[50] J. Jiang et al., "Optimizing Data Partitioning for Data-Parallel Computing," NSDI'12: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012.
[51] S. Venkataraman et al., "Drizzle: Fast and Adaptable Stream Processing at Scale," SOSP'17: Proceedings of the 26th Symposium on Operating Systems Principles, 2017.
[52] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," OSDI'16: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, 2016.
[53] J. Jiang et al., "Optimizing Data Partitioning for Data-Parallel Computing," NSDI'12: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012.
[54] B. H. Sigelman et al., "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Google Technical Report, 2010.
[55] Q. Lin et al., "Log Clustering Based Problem Identification for Online Service Systems," ICSE'16: Proceedings of the 38th International Conference on Software Engineering Companion, 2016.
[56] M. Malik et al., "Performance Debugging for Distributed Systems of Black Boxes," SOSP'03: Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003.
[57] NVIDIA, "GPU-Accelerated AI: Transforming Industries and Creating New Ones," NVIDIA Blog, 2021.
[58] J. Kaplan et al., "Scaling Laws for Neural Language Models," arXiv:2001.08361, 2020.
[59] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
[60] C. Bayer and M. Zepeda, "Deep Feature Engineering for Text Mining," arXiv:1810.05687, 2018.
[61] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model," Journal of Machine Learning Research, vol. 3, pp. 1137-1155, 2003.
[62] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013.
[63] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv:1301.3781, 2013.
[64] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013.
[65] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual String Embeddings for Sequence Labeling," COLING'18: Proceedings of the 27th International Conference on Computational Linguistics, 2018.
[66] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," ACL'16: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.
[67] Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv:1609.08144, 2016.

[68] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," ACL'16: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.
[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL-HLT'19: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.
[70] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL-HLT'19: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.
[71] A. Vaswani et al., "Attention is All You Need," NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.
[72] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," ICLR'15: Proceedings of the 3rd International Conference on Learning Representations, 2015.
[73] A. Vaswani et al., "Attention is All You Need," NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.
[74] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, 2020.
[75] A. Vaswani et al., "Attention is All You Need," NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.
[76] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," ICML'17: Proceedings of the 34th International Conference on Machine Learning, 2017.
[77] A. Vaswani et al., "Attention is All You Need," NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.
[78] N. Shazeer et al., "Generating Wikipedia by Summarizing Long Sequences," ICLR'18: Proceedings of the 6th International Conference on Learning Representations, 2018.
[79] C. Bayer and M. Zepeda, "Deep Feature Engineering for Text Mining," arXiv:1810.05687, 2018.
[80] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries," Proceedings of the 1st Instructional Conference on Machine Learning, 2003.
[81] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, vol. 3, pp. 993-1022, 2003.
[82] Airbnb Engineering, "Helping Hosts Optimize Their Listings with AI and Machine Learning," Airbnb Engineering & Data Science, 2020.
[83] NVIDIA, "Transformer-XL: Enhancing Language Models with Longer Context," NVIDIA Developer Blog, 2019.
[84] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL-HLT'19: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.
[85] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
[86] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," arXiv:1609.04747, 2016.
[87] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
[88] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," Journal of Machine Learning Research, vol. 13, pp. 281-305, 2012.
[89] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
[90] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012.
[91] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the Limits of Language Modeling," arXiv:1602.02410, 2016.
[92] J. Dean et al., "Large Scale Distributed Deep Networks," NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012.
[93] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," OSDI'16: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, 2016.
[94] NVIDIA, "NVIDIA Clocks World's Fastest BERT Training Time and Largest Transformer Based Model, Paving Path For Advanced Conversational AI," NVIDIA Press Release, 2019.
[95] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, 2004.

[96] M. Li et al., "Scaling Distributed Machine Learning with the Parameter Server," OSDI'14: Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, 2014.
[97] N. Shazeer et al., "Mesh-TensorFlow: Deep Learning for Supercomputers," NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018.
[98] Y. Huang et al., "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism," NeurIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019.
[99] P. Goyal et al., "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," arXiv:1706.02677, 2017.
[100] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes," ICLR'20: Proceedings of the 8th International Conference on Learning Representations, 2020.
[101] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI Blog, 2019.
[102] P. Micikevicius et al., "Mixed Precision Training," ICLR'18: Proceedings of the 6th International Conference on Learning Representations, 2018.
[103] NVIDIA, "NVIDIA Apex: Tools for Easy Mixed-Precision Training in PyTorch," NVIDIA Developer Blog, 2019.
[104] NVIDIA, "Automatic Mixed Precision for Deep Learning," NVIDIA Developer Blog, 2019.
[105] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," ICLR'17: Proceedings of the 5th International Conference on Learning Representations, 2017.
[106] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," WACV'17: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 2017.
[107] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," arXiv:1609.04747, 2016.
[108] L. Prechelt, "Early Stopping - But When?" in Neural Networks: Tricks of the Trade, Springer, 1998, pp. 55-69.
[109] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, 2020.
[110] N. Shazeer et al., "Mesh-TensorFlow: Deep Learning for Supercomputers," NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018.
[111] Microsoft Research, "Turing-NLG: A 17-Billion-Parameter Language Model," Microsoft Research Blog, 2020.
[112] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
[113] TensorFlow, "TensorBoard: TensorFlow's visualization toolkit," TensorFlow Documentation, 2021.
[114] Weights and Biases, "Experiment Tracking, Dataset Versioning, and Model Management," Weights and Biases Documentation, 2021.
[115] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
[116] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in Deploying Machine Learning: A Survey of Case Studies," arXiv:2011.09926, 2020.
[117] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2224-2287, 2019.
[118] D. Crankshaw et al., "Clipper: A Low-Latency Online Prediction Serving System," NSDI'17: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, 2017.
[119] K. Hightower, B. Burns, and J. Beda, "Kubernetes: Up and Running: Dive into the Future of Infrastructure," O'Reilly Media, 2017.
[120] Hugging Face, "Scaling Transformers to Serve 1.2B Requests a Month on Hugging Face Inference API," Hugging Face Blog, 2021.
[121] N. Talagala et al., "The Data-Centric Reliability Challenge: Maintaining the Integrity of the Data and Models in AI and Machine Learning Systems," arXiv:2010.10561, 2020.
[122] P. Bailis and K. Kingsbury, "The Network is Reliable," Communications of the ACM, vol. 57, no. 9, pp. 48-55, 2014.
[123] W. Darling, "Building Reliable and Scalable Systems at Netflix," Netflix Technology Blog, 2017.
[124] H. Basiri et al., "Chaos Engineering," IEEE Software, vol. 33, no. 3, pp. 35-41, 2016.
[125] C. Coleman et al., "DAWNBench: An End-to-End Deep Learning Benchmark and Competition," NeurIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018.
[126] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," ICLR'16: Proceedings of the 4th International Conference on Learning Representations, 2016.

[127] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," CVPR'18: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[128] O. Zafrir et al., "Q8BERT: Quantized 8-bit BERT," EMC2'19: Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition, 2019.
[129] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.
[130] D. Sato, C. Bezerra, D. Chaves, and F. Corrêa, "Monitoring Machine Learning Models in Production: A Comprehensive Guide," arXiv:2103.12762, 2021.
[131] Prometheus, "Monitoring with Prometheus," Prometheus Documentation, 2021.
[132] S. Schelter, J.-H. Böse, J. Kirschnick, T. Klein, and S. Seufert, "Automatically Tracking Metadata and Provenance of Machine Learning Experiments," MLSys'17: Proceedings of the 1st Workshop on Machine Learning Systems - Systems for ML and Open Source Software, 2017.
[133] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009.
[134] S. Bai, J. Z. Kolter, and V. Koltun, "Deep Equilibrium Models," NeurIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019.
[135] Airbnb Engineering, "Monitoring Machine Learning Models in Production," Airbnb Engineering & Data Science, 2020.
[136] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction," IEEE Big Data'17: Proceedings of the 2017 IEEE International Conference on Big Data, 2017.
[137] L. Baylor et al., "Continuous Evaluation for Production Machine Learning," KDD'19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.
[138] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," WWW'17: Proceedings of the 26th International Conference on World Wide Web, 2017.
[139] LinkedIn Engineering, "Continuous Delivery for Machine Learning: Patterns and Pains," LinkedIn Engineering Blog, 2021.
[140] M. Beauchemin, "Machine Learning Engineering at Uber: An Interview with Mike Del Balso," InfoQ, 2019.
[141] J. Humble and D. Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation," Addison-Wesley Professional, 2010.
[142] AWS, "CI/CD for Machine Learning Pipelines," AWS Architecture Blog, 2021.
[143] Uber Engineering, "Scaling Machine Learning at Uber with Michelangelo," Uber Engineering Blog, 2019.
[144] S. Amershi et al., "Software Engineering for Machine Learning: A Case Study," ICSE-SEIP'19: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 2019.
[145] Seldon, "Deploying Machine Learning Models: A Guide to Architecture and Best Practices," Seldon Documentation, 2021.
[146] A. Burkov, "Machine Learning Engineering," Manning Publications, 2019.
[147] TensorFlow, "TensorFlow Serving: A Flexible, High-Performance Serving System for Machine Learning Models," TensorFlow Documentation, 2021.