University of Idaho

Department of Computer Science

CS 404/504 Special Topics: Python Programming For Data Science

Dr. Alex Vakanski

Lecture 27

Introduction to Data Science Operations

Lecture Overview

- Introduction to Data Science Operations (DSOps)
- DS project life cycle
- DSOps levels of automation
- Model deployment
  - Production environment considerations
- Model serving
  - Batch prediction vs online prediction
  - Cloud computing vs edge computing
  - Model compression

Introduction to Data Science Operations

Introduction to Data Science Operations (DSOps)

- ***Data Science Operations (DSOps)*** is a set of practices for deployment of Data Science projects in production environment
- By the way Data Science is closely related and overlaps with Machine Learning, similarly, DSOps are related to ***Machine Learning Operations (MLOps)***
  - MLOPs is a set of practices for deployment of Machine Learning models in production environment
- DSOps practices are developed in collaboration between data scientists and operation engineers
  - DSOps provide means for:
    - Simplifying and automating the deployment
    - Assuring the quality of deployed projects
    - Meeting regulatory requirements
    - Aligning deployed projects with the business mission and objectives

Data Science Operations

Introduction to Data Science Operations (DSOps)

- Although originally DSOps was developed for managing project deployment and the operations phase, in recent years, the set of DSOps practices are increasingly applied for managing the entire life cycle of DS projects
- A depiction of the main phases in a DS project life cycle is shown below

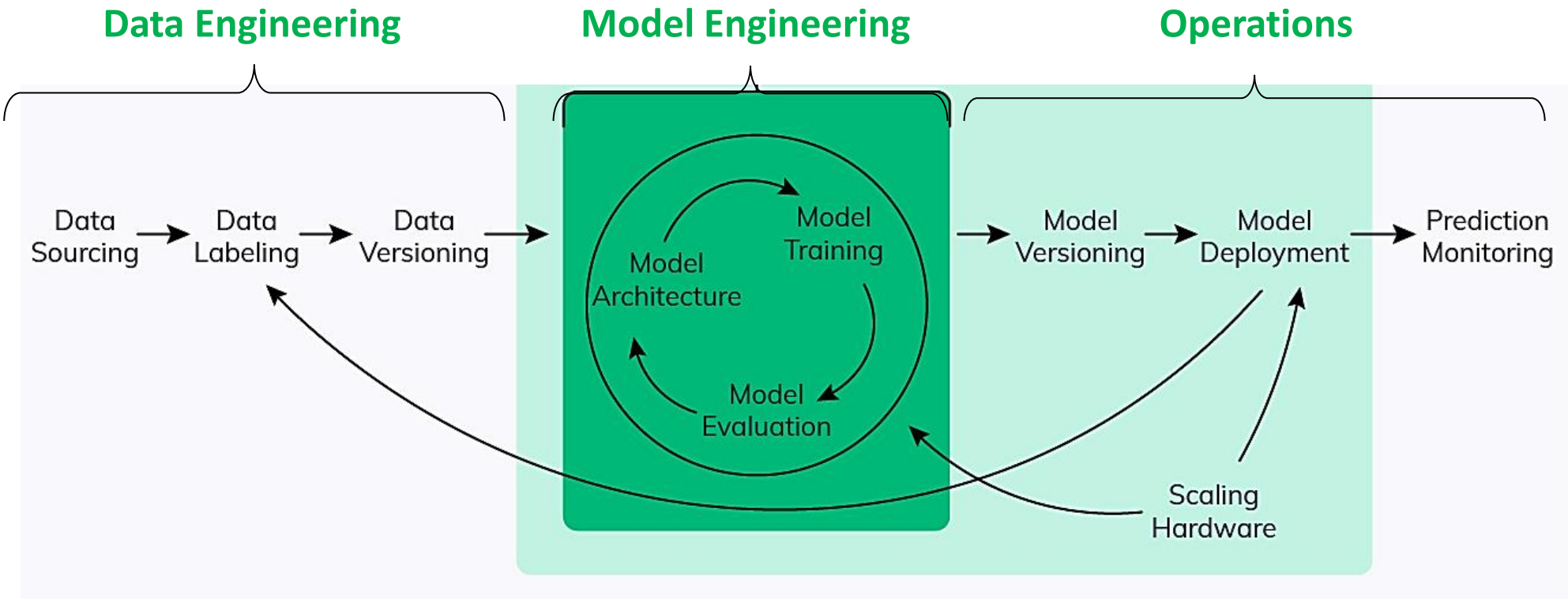


Figure from: https://neptune.ai/blog/mlops

DS Project Life Cycle

DS Project Life Cycle

- ***Data engineering phase*** (data preparation phase) in the life cycle of DS project can involve several steps, such as:
  - Data gathering / data sourcing
  - Data cleansing / labeling (if used for classification)
  - Data ingestion (preparing batches of data samples, or streams for real-time ingestion)
  - Data analysis and transformation (converting categorical variables, scaling numerical variables)
  - Data validation (missing values, duplicate features)
  - Feature engineering (selecting important features, dimensionality reduction with PCA)
  - Data splitting (train, test, validation datasets)

Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

DS Project Life Cycle

DS Project Life Cycle

- *Model engineering phase* (model creating phase) can involve:
  - Building a model
  - Training optimization (model search, hyperparameter tuning)
  - Model validation (evaluate on unseen test dataset)
  - Training at scale (train with new data, large datasets)
  - Select a final model to be deployed in production

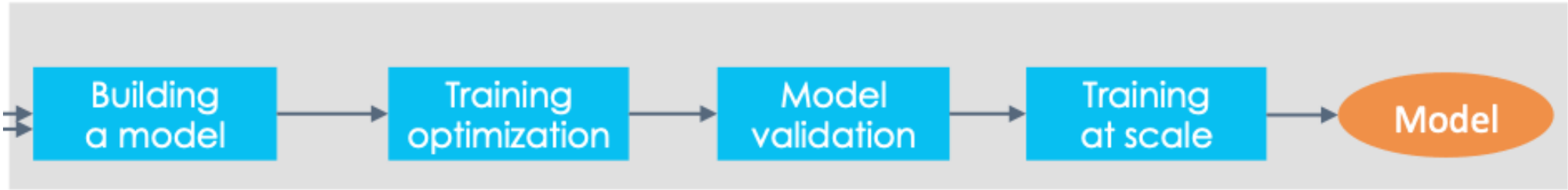


Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

DS Project Life Cycle

DS Project Life Cycle

- Model engineering phase typically require training multiple models with the goal of selecting the best performing model
  - An important part of model engineering phase is experiment tracking
- Experiment tracking includes collecting, organizing, and tracking model training information across multiple runs with different configurations (e.g., different data scaling, hyperparameters, data splits, parameters, etc.)
  - In practice, multiple models can be trained by different team members, different teams, or by different companies
- There are many available tools for experiment tracking, including:
  - CometML – to be covered in the next tutorial
  - Weights and Biases
  - TensorBoard
  - MLFlow
  - Neptune
  - Kubeflow

DS Project Life Cycle

DS Project Life Cycle

- *Model deployment phase* (operations phase) can involve:
  - Deploying the model for access by end-users (e.g., to the cloud, edge)
  - Serving the model (consuming the model by end-users)
  - Monitoring the deployed model and logging the performance
    - Checking for performance degradation
    - Providing explainability of the decision-making process by the model
  - Finetuning the model and improving the performance
    - Deployed models need to be updated periodically
  - Model versioning and data versioning
    - Keep track of deployed models and data used with different models

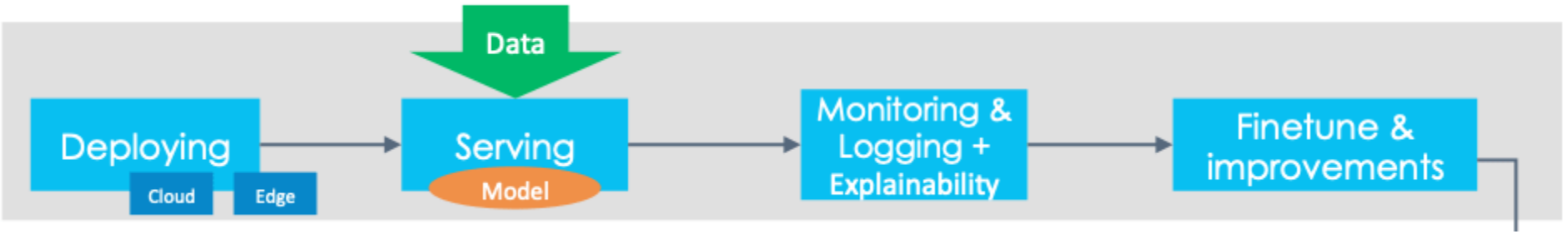


Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

Performance Degradation

DS Project Life Cycle

- Performance degradation in ML models can be caused by:
  - Discrepancy between the data processing techniques in the training and serving pipelines
  - Change in the data distribution between the model training and model serving phases
  - Using wrong objectives to optimize, that can result in collecting biased data for training the model
    - Afterward, new biased data can be collected that is used for re-training the future versions of the model, resulting in even more biased model
- Three main reasons for model performance degradation are:
  - Data drift, the data distribution is changed
    - E.g., a bug in the data pipeline, launch the model in a new country, new users from different demographics, malicious users feeding the model bad data
  - Concept drift, the relationship between the data and the output is changed
    - E.g., users' behavior changes
  - Domain shift, the data distribution is not sampled adequately
    - E.g., bias in the data sampling process, outliers (rare examples that are not present in the training data occur)

DS Project Life Cycle

DS Project Life Cycle

- DSOps practices are frequently used to streamline the entire life cycle of DS projects, from data preparation and model creation, to model deployment and monitoring
  - Application of DSOps allows to run and deploy DS project consistently from end to end

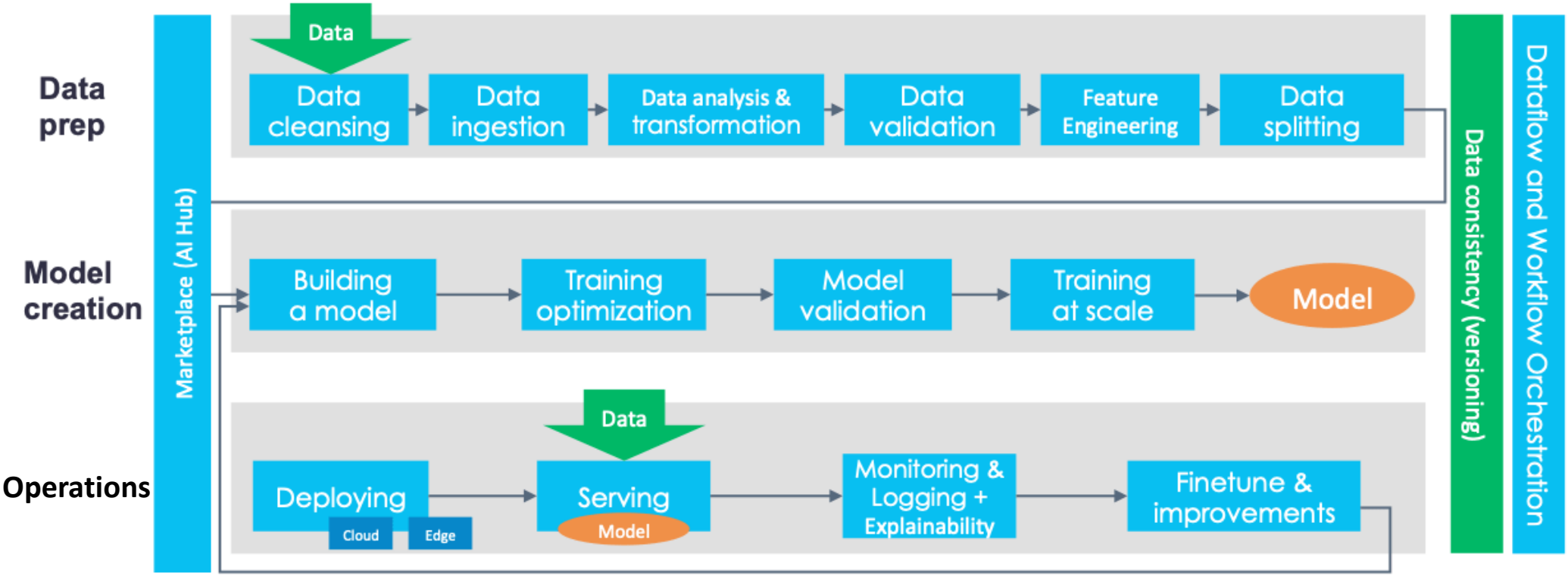


Figure from: https://www.kdnuggets.com/2020/07/tour-end-to-end-machine-learning-platforms.html

DS Project Life Cycle

DS Project Life Cycle

- Using consistent DSOps practices allows organizations to:
  - Automate data engineering, model engineering, and model deployment pipelines
  - Run reproducible projects by versioning data and models
  - Deploy projects in a consistent way that assures quality and provides reusability
  - Proactively address various concerns, such as regulatory compliance of deployed projects
  - Reduce costs by creating efficient workflows

DSOps, MLOps vs DevOps

DSOps, MLOps vs DevOps

- DSOps and MLOps were derived based on the *Development Operations (DevOps)* principles for software deployment and operations
  - DevOps were developed to shorten software development life cycle via consistent practices for software testing, versioning, continuous delivery, monitoring
- The differences between DSOPs/MLOps and DevOps include:
  - DSOps is more experimental, since it involves training and validation of multiple models
  - DSOps typically requires hybrid teams that consist of data scientists, ML researchers, software engineers, deployment engineers, etc.
  - DS projects are characterized with performance degradation over time, due to evolving data
    - Requires monitoring models in production (e.g., adopt metrics to be monitored), and re-training and updating the deployed models
  - Continuous Integration (CI) involves testing and validating data and models, not only code
  - Continuous Delivery (CD) requires to establish model training pipeline and model deployment pipeline

DSOps Levels of Automation

DSOps Levels of Automation

- The level of automation of DSOps determines the time with which new models can be trained and deployed
  - The objective of DSOps teams is to automate the complete life cycle and perform all steps without any manual intervention
- DSOps can be generally implemented at 3 levels of automation
  - DSOps Level 0 - manual process
  - DSOps Level 1 - ML pipeline automation
  - DSOps Level 2 - CI/CD pipeline automation

DSOps Level 0 – Manual Process

DSOps Levels of Automation

- ***DSOps Level 0***
  - This level is typical for companies that are just starting with DS and ML
- Characteristics:
  - Manual process: each step of the life cycle is completed manually
  - Disconnect between ML and operations: data scientists create the model, and hand it over to engineers who deploy and serve the model
  - Infrequent model iterations: the data science team manages a few models that don't change frequently (e.g., update models a few times per year)
  - No Continuous Integration (CI) or Continuous Delivery (CD): testing the models is part of the script or notebook used for training
  - Deployment: typically as a microservice with REST API
  - Lack of active performance monitoring

DSOps Level 1 – ML Pipeline Automation

DSOps Levels of Automation

- ***DSOps Level 1***
  - The goal is to automate the ML model pipeline to achieve continuous delivery of prediction service
- Characteristics:
  - Rapid experimentation: experiments are orchestrated and done automatically
  - Continuous training (CT): the model is automatically re-trained in production, using fresh data
    - Requires automated validation of new data, and validation of updated models
    - Requires pipeline triggers to retrain models with new data (e.g., based on a schedule, availability of new data, model performance degradation, data distribution shift)
  - Modularized code for pipelines: code is designed to be reusable and shareable across the ML pipelines (e.g., using containers)
  - Continuous delivery of models: the model deployment step that serves the trained and validated model as a prediction service is automated
  - Pipeline deployment: differently from Level 0 where a trained model is deployed as a prediction service, Level 1 deploys an entire ML model pipeline that automatically runs to serve the trained model as the prediction service

DSOps Level 2 – CI/CD Pipeline Automation

DSOps Levels of Automation

- ***DSOps Level 2***
  - The goal is to automate the CI/CD pipeline
- Level 2 is more suitable for tech-driven companies that need to:
  - Manage a large number of models
  - Retrain the models frequently (e.g., daily, or hourly)
  - Redeploy the models on many servers simultaneously
- Characteristics:
  - Development and experimentation: iteratively try new ML algorithms, and push the code to a source repository
  - Pipeline CI: build source code and run various tests, output are pipeline components to be deployed in a later stage
  - Pipeline CD: deploy the components from CI to the target environment, outputs is a deployed pipeline with an updated model
  - Automated triggering: the pipeline is automatically executed in production based on a schedule or in response to a trigger
  - Model CD: the updated model is served as a prediction service
  - Monitoring: monitor model performance based on live data

DSOps Level 2 – CI/CD Pipeline Automation

DSOps Levels of Automation

- DSOps Level 2 requires the following components
  - Source control: for versioning the code, data, and models
  - Test & build services: using CI tools for quality assurance and building packages and executables for pipelines
  - Deployment services: using CD tools for deploying pipelines to the target environment
  - Model registry: a registry for storing trained ML models
  - ML metadata store: tracking metadata of model training, such as model name, parameters, training data, test data, and metric results
  - Feature store: a repository to standardize the definition, storage, and access of features for training and serving
  - ML pipeline orchestrator: automating the steps of ML experiments

Model Deployment

Model Deployment

- Environments in ML model lifecycle can include:
  - Development environment, to run models during model development
  - Testing environment, to test the deployed model
  - Production environment, for consuming the deployed model by end-users
- ***Model deployment*** comprises activities for making the model accessible for making predictions (inference)
  - The model is run in a production environment, where it needs to provide reliable and fast prediction service to end-users
  - A report from 2021 shows that 41% of organizations with over 25,000 employees have more than 100 models in production
  - Some companies update their ML model every 10 minutes

Production Environment Considerations

Model Deployment

- Most people learn DS and ML concepts from courses, academic papers, or from research projects (e.g., in graduate school)
- Deploying ML models in production has important differences in comparison to ML practice in courses or research projects
  - The tables lists some of the main differences

	Courses/Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have	Important
Interpretability	Good to have	Important

Table from: Chip Huyen, Designing Machine Learning Systems

Production Environment Considerations

Model Deployment

- **Objectives**
  - Research: design an ML model that achieves state-of-the-art performance
  - Production: design an ML that satisfies the requirements of many stakeholders
    - E.g., managers, sales team, product managers, infrastructure engineers
- **Computational priority**
  - Latency versus throughput
    - Latency – time from receiving a query to returning a result
    - Throughput – number of processed query within a time period
  - Production environment prioritizes low latency
    - 53% phone users will leave a page that takes more than 3 seconds to load

Production Environment Considerations

Model Deployment

- **Data**
  - Research: clean, static, mostly historical data
  - Production: messy, constantly shifting, historical + streaming data

  - Figure: time spent on dataset preparation and model development, according to Andrej Karpathy

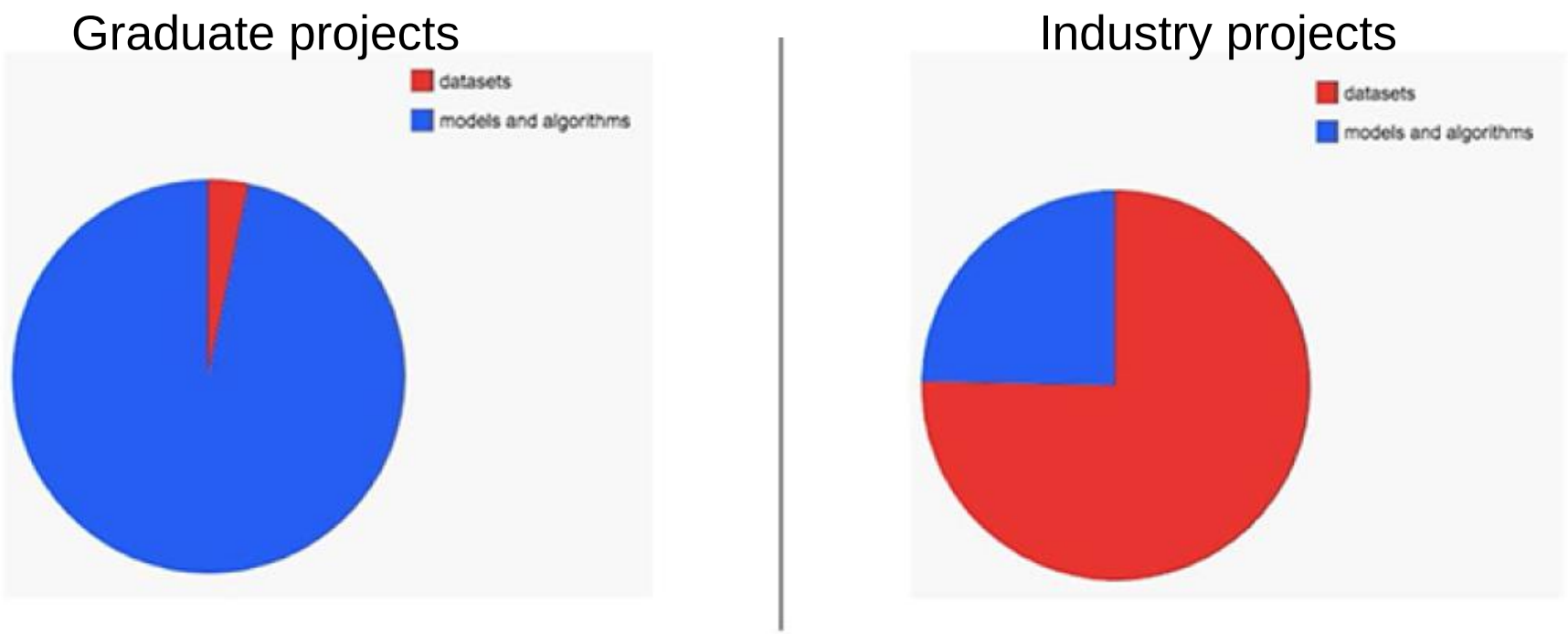


Figure from: Chip Huyen, Designing Machine Learning Systems

Production Environment Considerations

Model Deployment

- Other important considerations for deploying models in production include:
  - **Fairness**
    - The model is not biased against demographics groups
    - E.g., job applications are not rejected by a model because of the ZIP code
  - **Interpretability**
    - The users can interpret the decision-making process by the model
    - Interpretability is very important for the end-users to thrust the model
      - E.g., a model that is used for tumor classification needs to explain to clinicians how the prediction was made, to be trustworthy

Production Environment Considerations

Model Deployment

- Engineering challenges with large ML models
  - Too large to fit on consumer devices (e.g., cell phones)
  - Consume too much energy to work on consumer devices
  - Generating a prediction can take too long to be useful

Production Environment Considerations

Model Deployment

- Pace of software deployment and ML model deployment is accelerating
  - Performers who have established DevOps deploy 973 times more frequently, and have 6570 times faster lead time to deployment

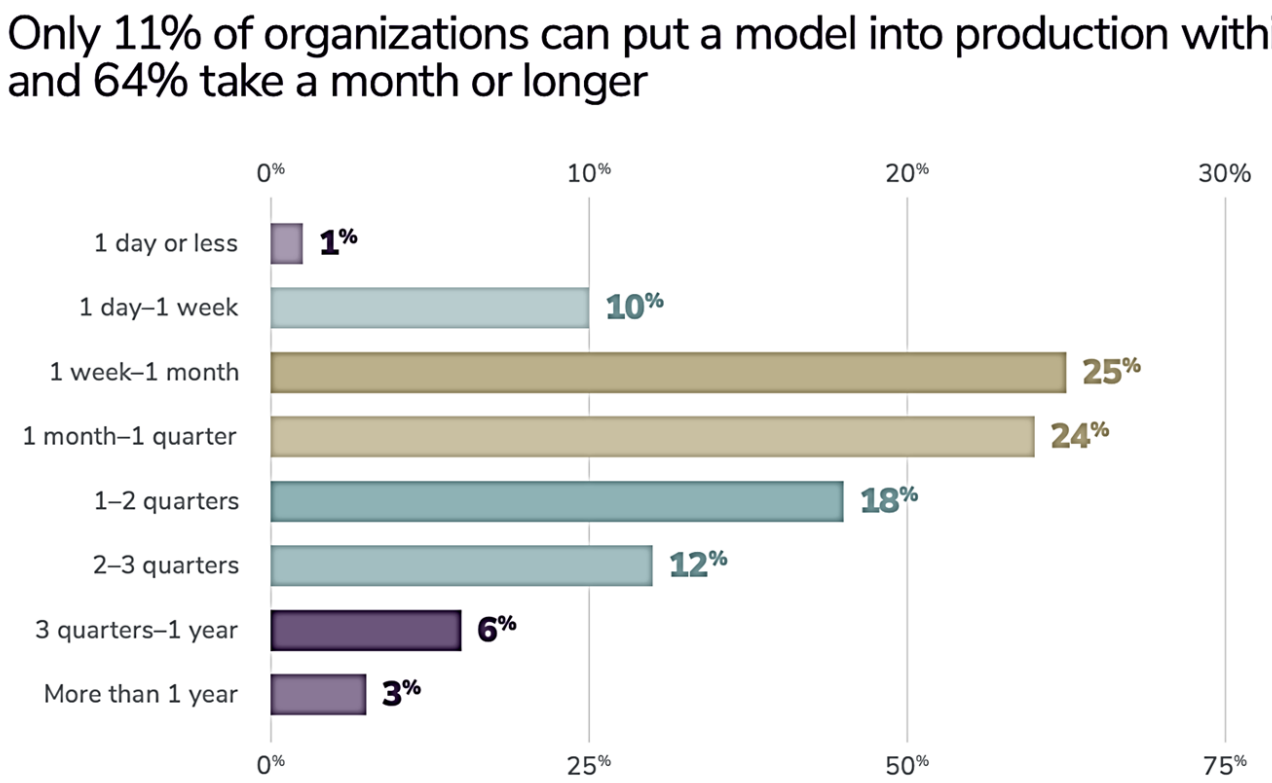


Figure from: Chip Huyen, Designing Machine Learning Systems

Online Prediction vs. Batch Prediction

Online Prediction vs. Batch Prediction

- During deployment, it is important to decide whether the predictions will be produced to the end-users as online prediction or batch prediction
- *Online prediction*
  - Generate predictions as soon as requests from end-users arrive
  - Predictions are returned as responses immediately to the end-users
  - It is also known as *synchronous prediction* (or *on-demand prediction*) since the predictions are generated synchronously with requests
- *Batch prediction*
  - Generate predictions periodically before requests arrive, store them (e.g., in SQL tables, or CSV files), and retrieve them when needed
    - E.g., movie recommendations are generated every 4 hours, and are shown to users when they log onto the website
  - It is also known as *asynchronous prediction*, since the predictions are generated asynchronously with requests

Batch Prediction

Online Prediction vs. Batch Prediction

- In the figure depicting *Batch Prediction*, a Data Warehouse is used to store batch predictions
  - The Prediction Service periodically supplies predictions to the Data Warehouse (step 2), using Batch Features (step 1)
  - When the App requests a prediction (step 3), Precomputed Predictions are fetched from the Data Warehouse (step 4)
- Batch features are computed from historical data, and stored in data warehouses or databases
  - Data warehouses are repositories that integrate data from one or more sources, and store current and historical data in one single place

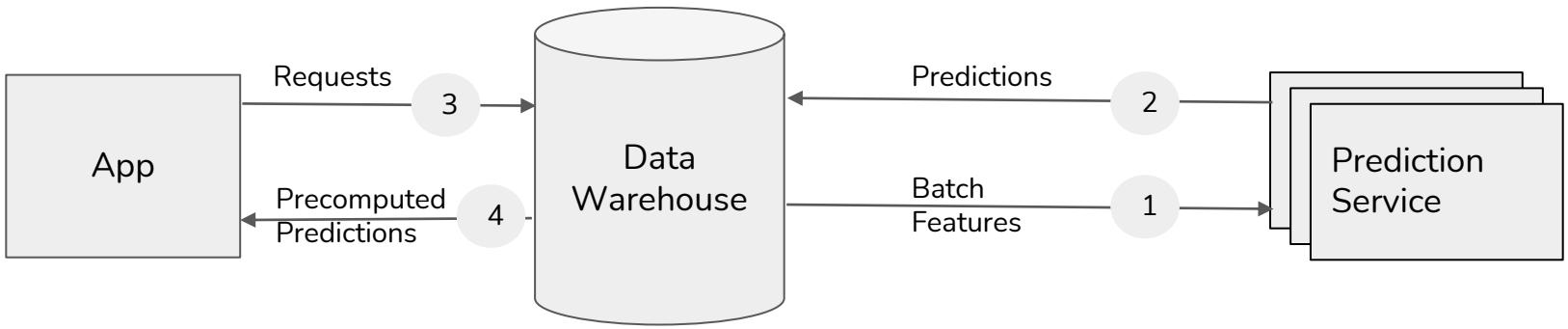


Figure from: Chip Huyen, Designing Machine Learning Systems

Online Prediction

Online Prediction vs. Batch Prediction

- In *Online Prediction*, when the App sends a request (step 1), the Prediction Service uses data features (step 2) to generate Predictions for the App (step 3)
- Online prediction can use both batch features (from historical data) or streaming features (from real-time data)
  - E.g., to estimate delivery time for an order on Doordash, ML model can use batch features (mean preparation time of this restaurant in the past) and streaming features (number of orders in real-time, number of delivery persons)

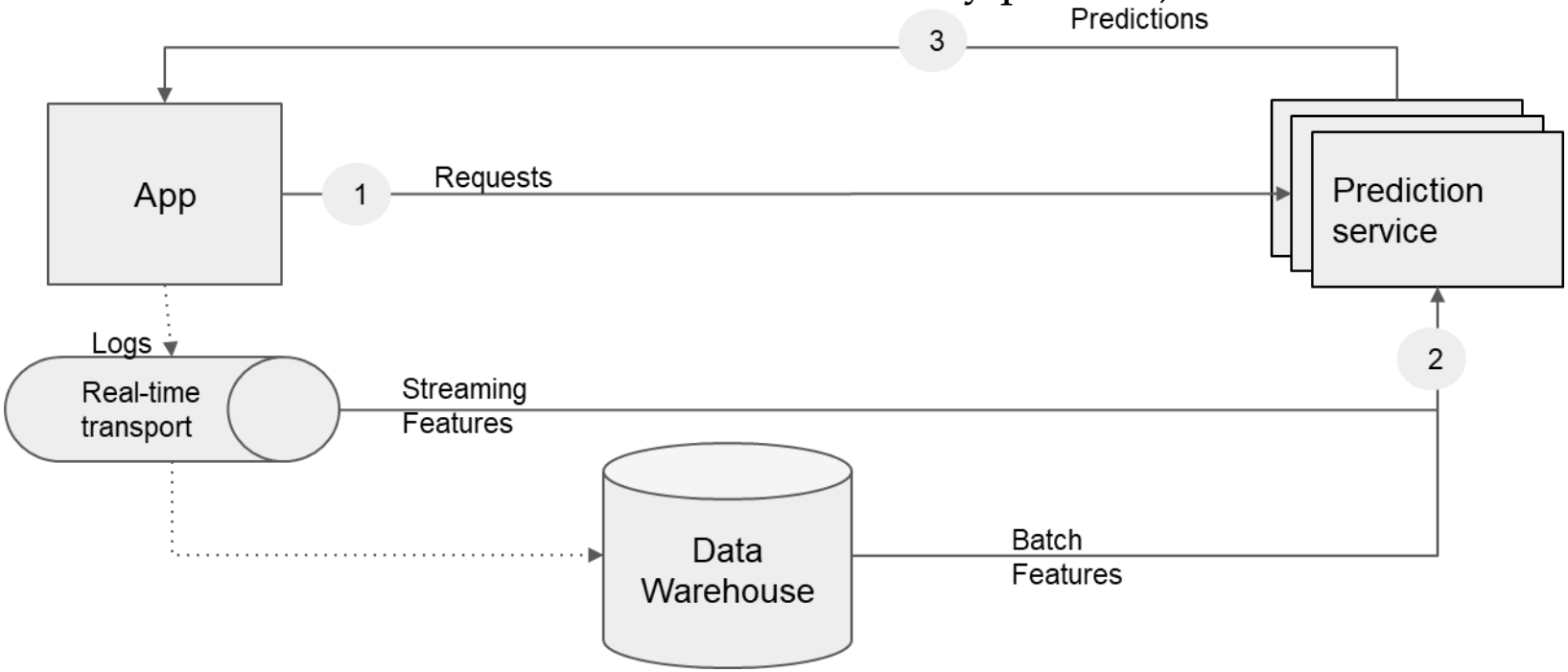


Figure from: Chip Huyen, Designing Machine Learning Systems

Online Prediction vs. Batch Prediction

Online Prediction vs. Batch Prediction

- Comparison between Online Prediction and Batch Prediction

	Online prediction (synchronous)	**Batch prediction (asynchronous)**
Frequency	As soon as requests come	Periodical (e.g., every 4 hours)
Useful for	When predictions are needed as soon as data sample is generated (e.g. fraud detection)	Processing accumulated data when you don't need immediate results (e.g. recommendation systems)
Optimized	Low latency	High throughput
Input space	Can be infinite	Finite: need to know how many predictions to generate
Examples	• Google Assistant speech recognition • Twitter feed	• TripAdvisor hotel ranking • Netflix recommendations

Table from: Chip Huyen, Designing Machine Learning Systems

Deploying to the Cloud vs. Edge

Cloud Computing vs Edge Computing

- *Cloud computing* is performed on web servers
  - Examples:
    - Queries to Alexa, Siri, Google Assistant
    - Queries to Google Translate
  - Cloud providers offer many tools that facilitate model deployment
  - It is a preferred way for model deployment by many companies
- *Edge computing* is performed on edge devices, such as cell phones, tablets, laptops, embedded devices, smart watches, cars, etc.
  - Examples:
    - Unlocking devices with fingerprints, face recognition
  - Running models on edge devices is an appealing option, and many companies have been recently trying to push their models to edge devices

Benefits of Cloud Computing

Cloud Computing vs Edge Computing

- Benefits of cloud computing:
  - Access to structured environments (libraries, tools) for managing DS projects
  - Access to latest computational resources (no need to purchase hardware or software)
  - Pay for what you need only
  - Ability to quickly scale projects
  - Improved efficiency by relying on infrastructure hosted and managed by the cloud provider, and ML tools developed by the cloud provider
- Cloud computing companies have invested hundreds of billions of dollars in infrastructure and management
  - Spending reached $178 billion in 2021, a growth of 37% from 2020
  - Amazon Web Services (AWS), Microsoft, and Google Cloud accounted for half of the spending in 2020
- AWS Sagemaker, Microsoft Azure ML, Google Cloud AI provide many DSOps tools to facilitate the life cycle of DS projects

Limitations of Cloud Computing

Cloud Computing vs Edge Computing

- Many companies use a combination of on-premise infrastructure and cloud infrastructure
- The following figure is from a recent survey

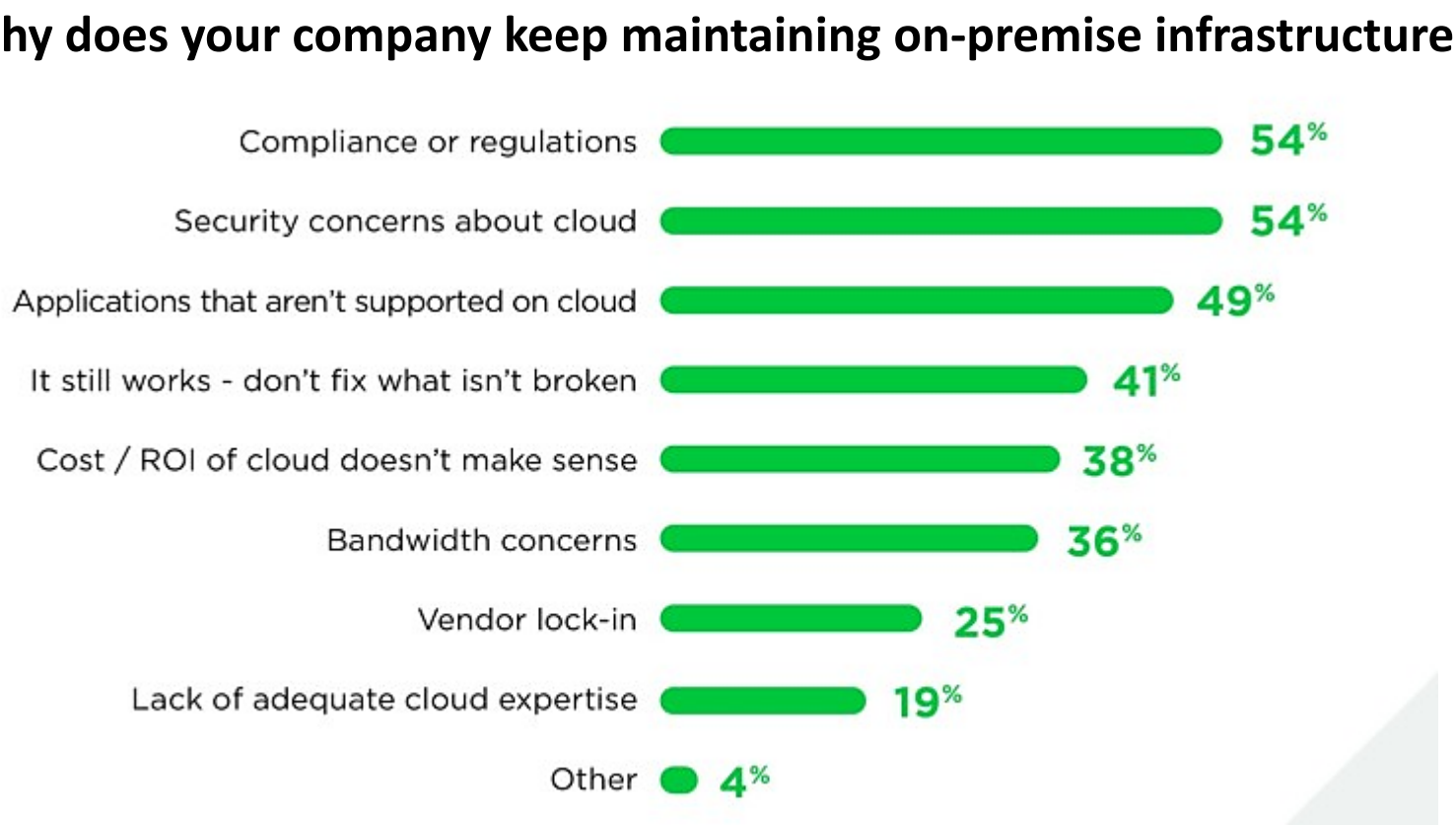


Figure from: Chip Huyen, Designing Machine Learning Systems

Benefits of Edge Computing

Cloud Computing vs Edge Computing

- Can work without Internet connection, or with unreliable connections (rural areas)
  - Many companies have strict no-Internet policy
  - Caveat: in some cases, apps may need external real-time information to work
- Do not need to worry about network latency
  - I.e., latency in sending data to the model on the Cloud, and afterward sending prediction back to the users
  - Large network latency can make some tasks impossible (e.g., text autocomplete)
  - Network latency might be a bigger problem than inference latency in some cases
    - Inference latency is the time for the model to make prediction
- Fewer privacy concerns
  - There is no need to send user data over networks (which can be intercepted)
  - Cloud database breaches can affect many people
    - "Nearly 80% of companies experienced a cloud data breach in past 18 months," Security Magazine
  - Easier to comply with regulations (e.g., GDPR by European Union)
  - Caveat: there is a risk of accessing user data by stealing the edge device

Benefits of Edge Computing

Cloud Computing vs Edge Computing

- Cheaper
  - Training ML models can take long time and can results in high Cloud service costs
  - A mistake in estimating Cloud computing costs can bankrupt your startup
  - Companies are reconsidering using edge devices

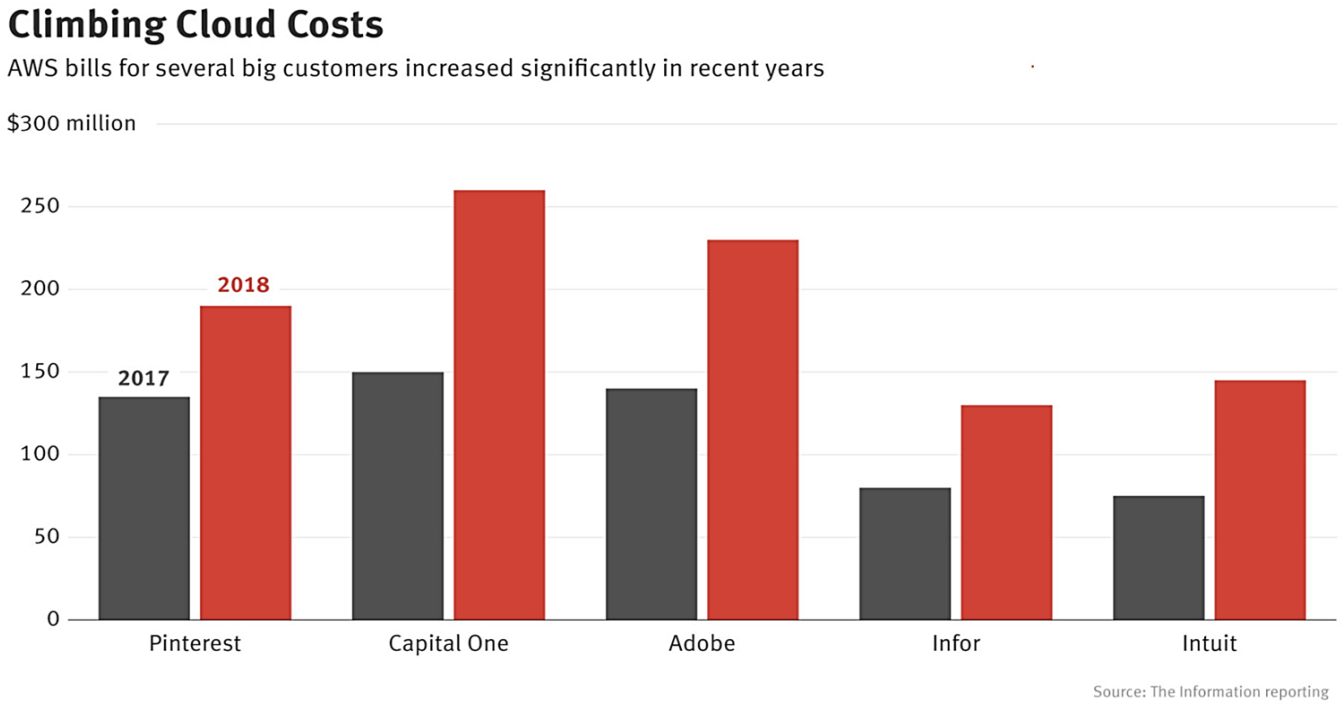


Figure from: Chip Huyen, Designing Machine Learning Systems

Disadvantages of Edge Computing

Cloud Computing vs Edge Computing

- The main disadvange of edge computing is that the devices may not be powerful enough to run models (especially high-performing DL models)
  - Computational power constraint
    - Large deep learning models require GPU
  - Memory constraint
    - To store the model and load it into memory
  - Energy constraint
    - Battery power to run an app for long periods of time

Potential Solutions

Cloud Computing vs Edge Computing

- Hybrid cloud-edge computing
  - Predictions for frequent inputs are precomputed on the Cloud and stored on the Edge device
  - Local data centers (e.g., each warehouse has its own on-premises server)

- Improve the capabilities of edge devices
  - Hardware - manufacture more powerful hardware
  - Model compression – use smaller models
  - Model optimization – implement vectorization, parallelization

Model Compression

Model Compression

- Current trend is toward larger models with billions of parameters
  - However, large models require powerful hardware and have large inference latency
- ***Model compression*** is a set of techniques for reducing the size of models
  - Compressed models have lower inference latency and can run on edge devices
- Model compression techniques
  - Pruning
  - Quantization
  - Knowledge distillation
  - Low-rank factorization

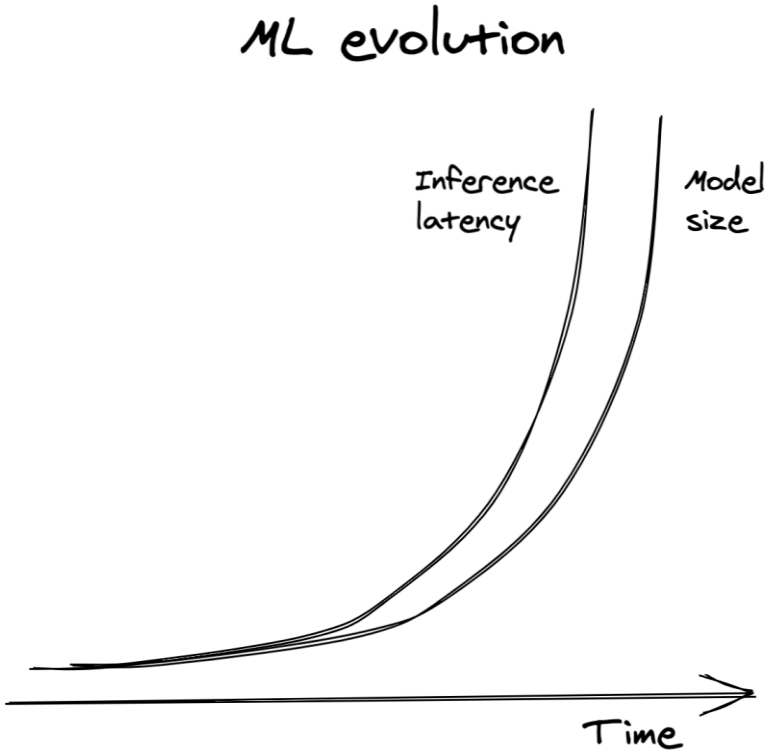


Figure from: Chip Huyen, Designing Machine Learning Systems

Model Compression: Pruning

Model Compression

- Pruning was originally used for decision trees to remove uncritical and redundant sections
  - It was widely adopted for pruning neural networks afterward
- Neural networks
  1. Remove least useful neurons
  2. Set least useful neurons to 0
     - Number of parameters remains the same
- Pruning makes models more sparse
  - Lower memory footprint
  - Increased inference speed

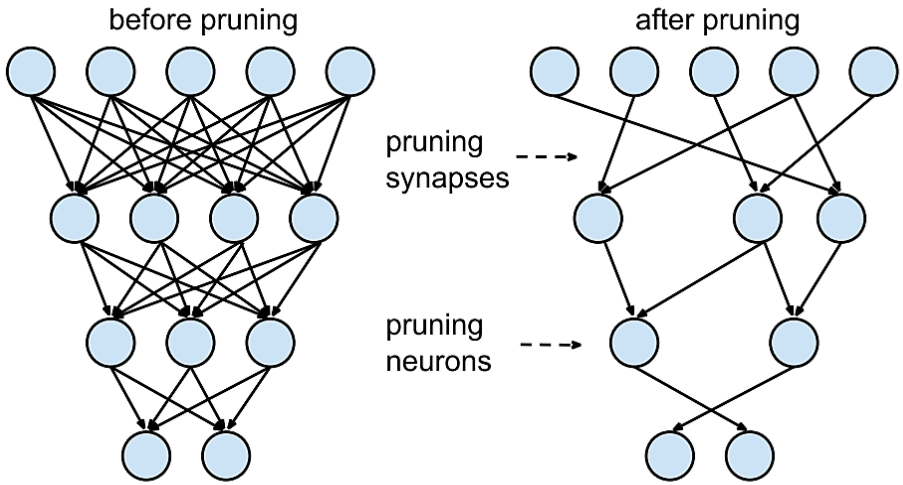
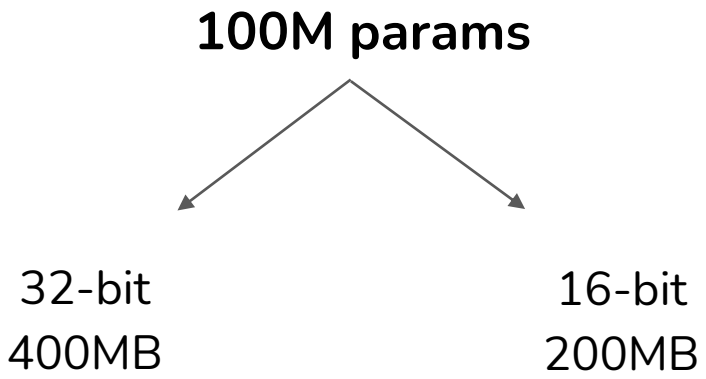


Figure from: Chip Huyen, Designing Machine Learning Systems

Model Compression: Quantization

Model Compression

- Quantization reduces the size of a model by using fewer bits to represent parameter values
  - E.g., instead of full-precision (32-bit) for each float number parameter, use half-precision (16-bit) or integer (8-bit)



- Pros
  - Reduce memory footprint
  - Increase computation speed
    - Computation on 16 bits is faster than on 32 bits
- Cons
  - Smaller range of values
  - Values rounded to 0
    - Need efficient rounding/scaling techniques

Model Compression: Knowledge distillation

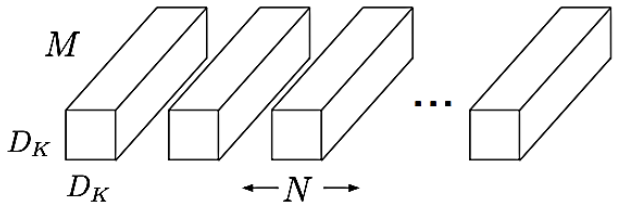
Model Compression

- Knowledge distillation refers to training a small model (student) to mimic the results of a larger model (teacher)
  - First pretrain the teacher, and afterward train the student to mimic the teacher
  - E.g., DistillBERT reduces size of BERT by 40%, and increases inference speed by 60%, while retaining 97% language understanding
- Pros:
  - Student can be fast to train
  - Teacher and student can have completely different architectures
- Cons:
  - It may need more data and time to first train teacher
  - The success depends on the application

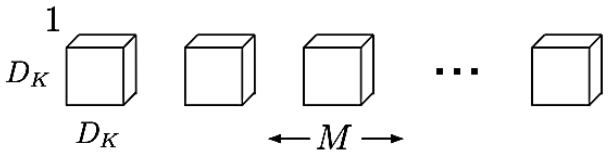
Model Compression: Factorization

Model Compression

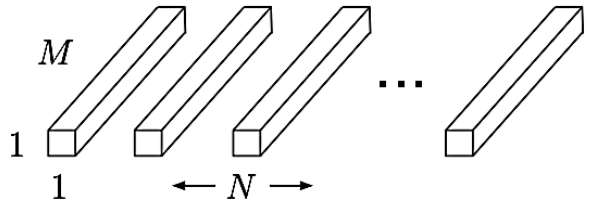
- Low-rank factorization is replacing high-dimensional tensors with lower-dimensional tensors
- E.g., a 3 x 3 matrix can be written as a product of 3 x 1 and 1 x 1
  - 6 params instead of 9
- This approach is used with some Conv Nets
  - MobileNet decomposes a covolutional filter of size $K \times K \times C$ into a depthwise convolution ($K \times K \times 1$) and pointwsie convolution ($1 \times 1 \times C$)
  - Use $K^2 + C$ parameters, instead of K^2C
  - MobileNets can achieve slighlty lower accuracy with much lower number of parameters, in comparison to other Conv Nets



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

References

1. Chip Huyen, Designing Machine Learning Systems, O'Reilly Media, 2022.
2. MLOps: What It Is, Why It Matters, and How to Implement It, by Prince Canuma, available at: https://neptune.ai/blog/mlops