



MLOps Coursera - 2

202132033 염지현



2022.01.07.

good
slide

CONTENTS

MLOps[3] Machine Learning Modeling Pipelines in Production

- I Machine Learning Modeling Pipelines in Production
- II Model Resource Management Techniques
- III High-Performance Modeling
- IV Model Analysis
- V Interpretability

CONTENTS

MLOps[4]

Deploying Machine Learning Models in Production

- I Model Serving: Introduction Model Serving
- II Model Serving: Patterns and Infrastructure
- III Model Management and Delivery

MLOps[3]

Introduction to Machine Learning in Production

I

Machine Learning Modeling Pipelines in Production

1. AutoML
2. NAS

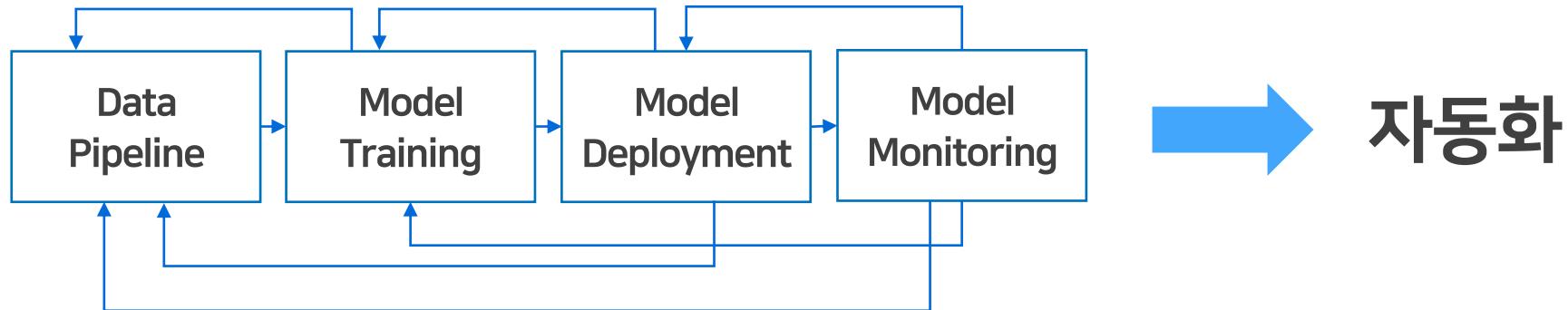
AutoML

AutoML 개념

AutoML

개념

- : 자동화된 ML 시스템
- : 데이터 전처리 과정부터 모델 훈련, 배포, 모니터링 단계에서 개발자의 개입을 최소화
- : ML 경험이 없는 개발자가 ML 모델 및 기술을 사용할 수 있도록 하는 것이 주목표



AutoML

AutoML workflow

• • •



Neural Architecture Search(NAS)

1) 개념

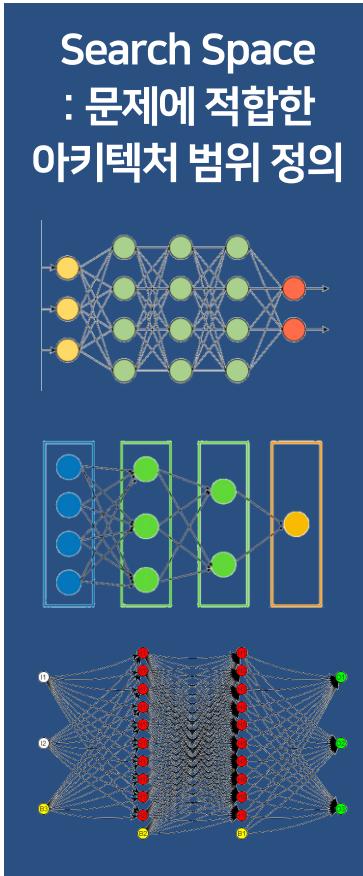
- : 인공신경망 설계를 자동화하기 위한 기술
- : 최적의 아키텍처를 찾는데 도움을 주는 기술
- : 거대한 Space를 통해 인공신경망 검색

2) 사용하는 이유

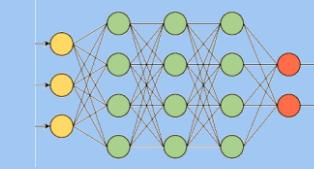
- : 가중치는 학습 시에 계속 업데이트 되지만 하이퍼파라미터는 지속적으로 수동 조정 필요
- : Activation function, weight initialization, learning rate 등 수동 조정의 어려움을 해결하기 위해

NAS

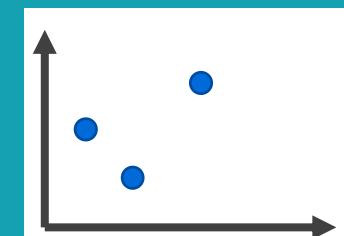
I NAS 프레임워크



Neural Architecture

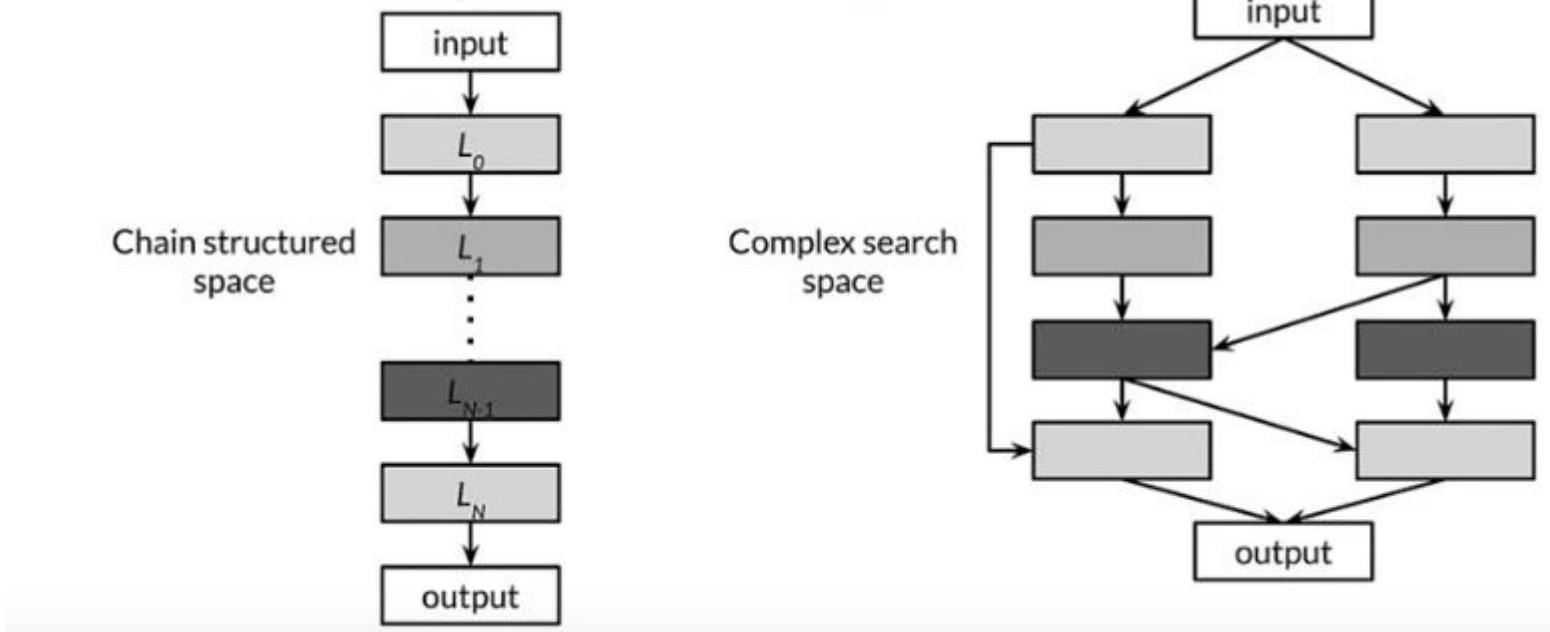


Performance Estimation
: 다양한 아키텍처 성능 비교



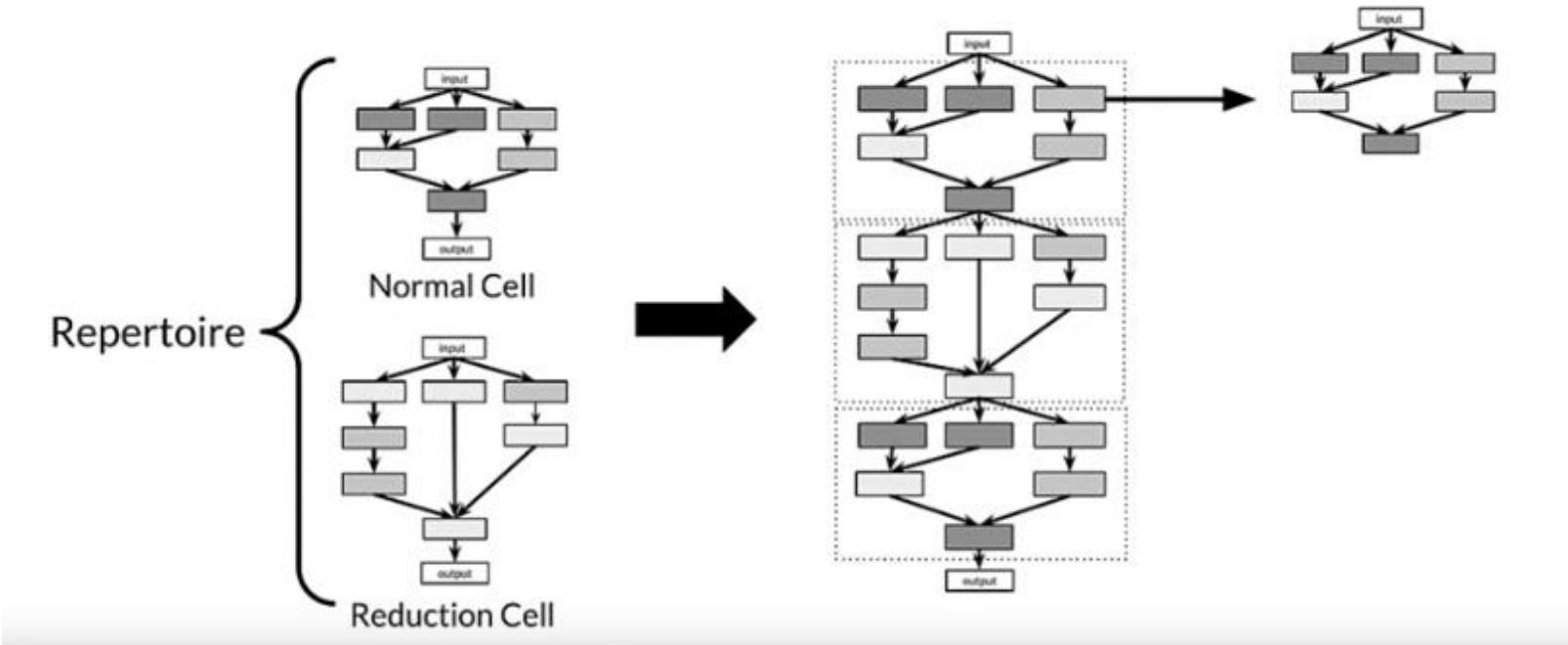
Search Space(1): Macro

Contains individual layers and connection types

**Macro**

- 최상의 모델을 검색하여 모델을 레이어별로 구축
- 복잡한 공간에서 체인 구조로 레이어 쌓기
- 여러 분기, 건너뛰기 연결을 사용하여 네트워크 구축

Search Space(1): Micro



Micro

- 더 작은 네트워크인 셀에서 신경망 구축
- 최종 네트워크 생성에서는 셀을 쌓아서 완성

I Search Strategies



Grid Search & Random Search

Grid Search & Random Search

1. Grid Search

- Search space에 있는 모든 아키텍처 검색

2. Random Search

- 무작위 검색

장점: Search space가 작다면 옳은 방법

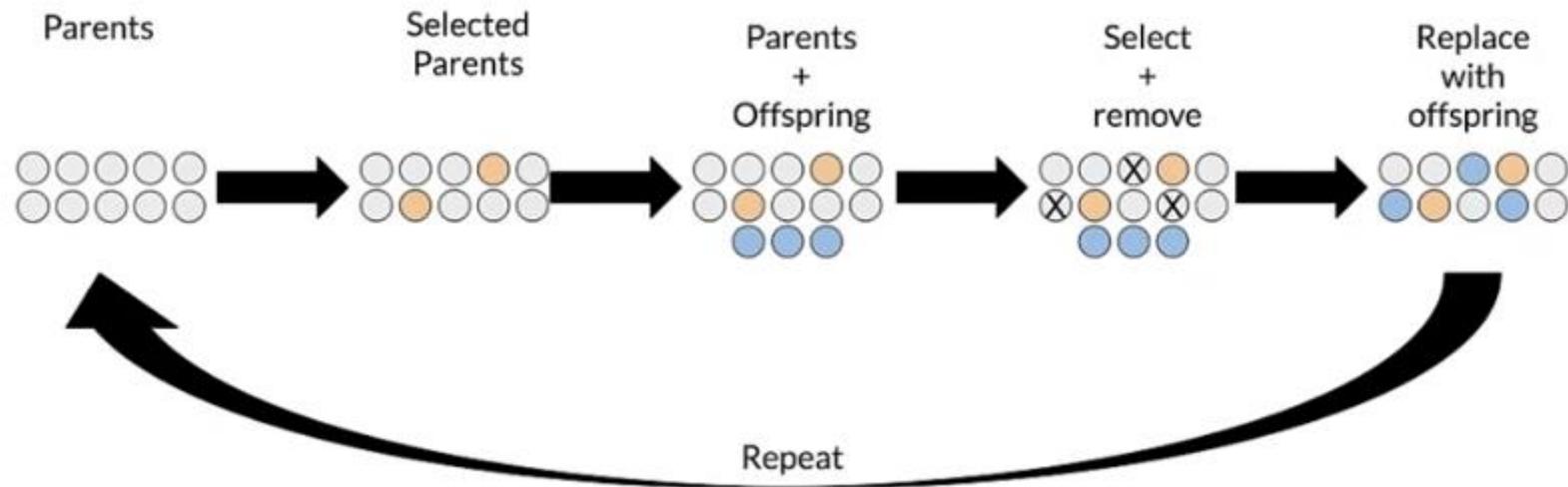
단점:

1. 빠르게 실패
2. Search space가 크면 성능이 좋지 못함

Bayesian Optimization

-
- 1. 성능을 기반으로 특정 확률 분포 가정
- 2. 아키텍처의 성능을 평가하고 다음 옵션 선택
- 3. 이러한 방식으로 유망한 아키텍처를 확률적으로 결정 및 테스트 가능

Evolutionary Algorithms



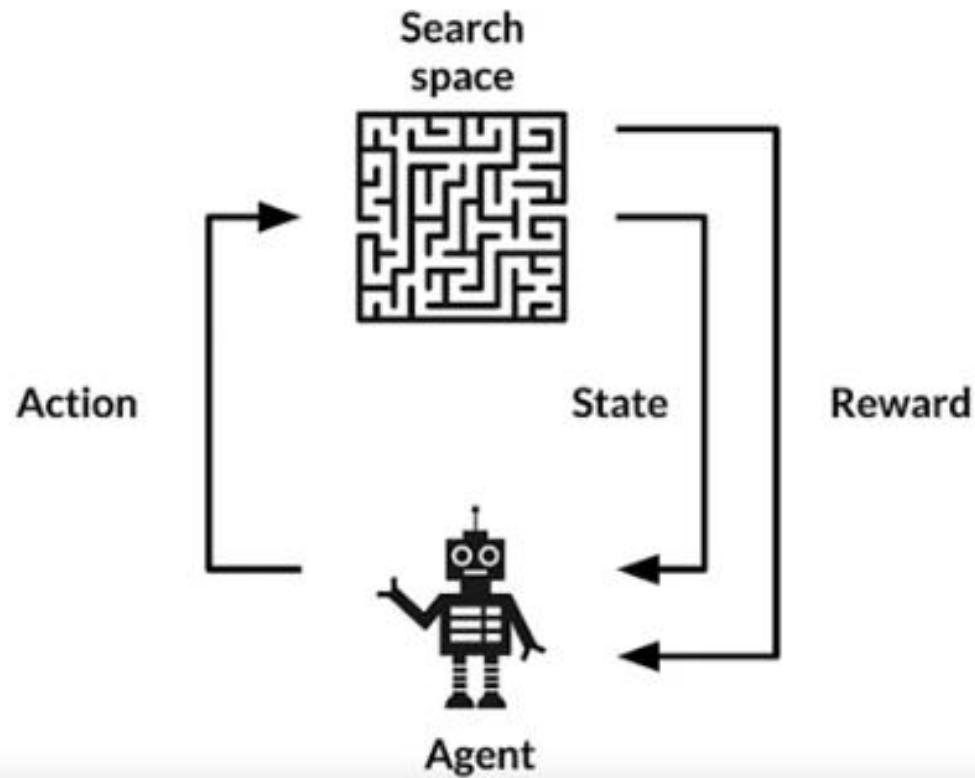
1. 서로 다른 모델 아키텍처의 초기 모집단 무작위 생성
2. 성능 평가 기반 부모 선택
3. 부모 + 자식 합침
4. 레이어 추가 또는 제거
5. 자손이 대체되고 1번부터 반복

Reinforcement Learning

1. Agent 목표: 보상 최대화

2. 가능한 옵션은 search space에서 선택

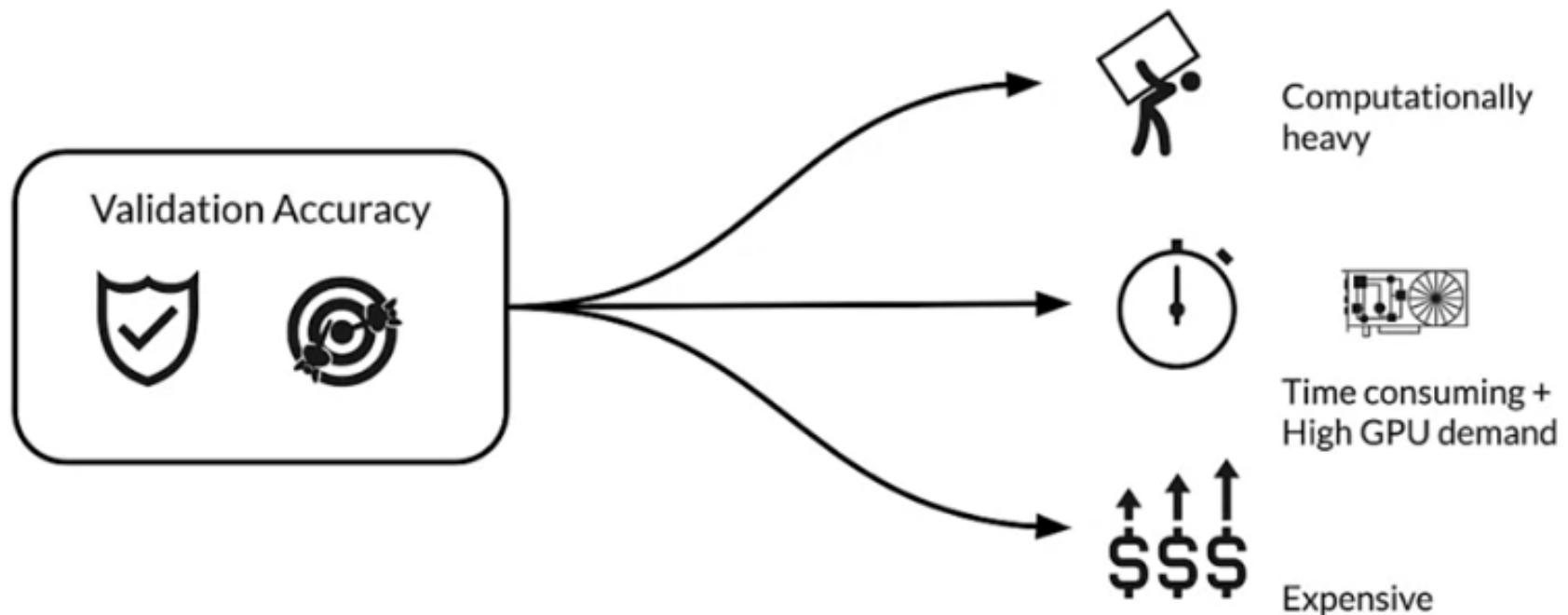
3. 성능 추정 전략: 보상으로 결정



성능 평가

성능 평가 전략

생성된 아키텍처의 성능을 추정하여 더 나은 아키텍처를 생성해야 함



MLOps[3]

Introduction to Machine Learning in Production

III

Model Resource Management — Techniques

1. 차원 축소
2. 양자화
3. 가지치기

차원 축소

II 고차원 데이터 단점

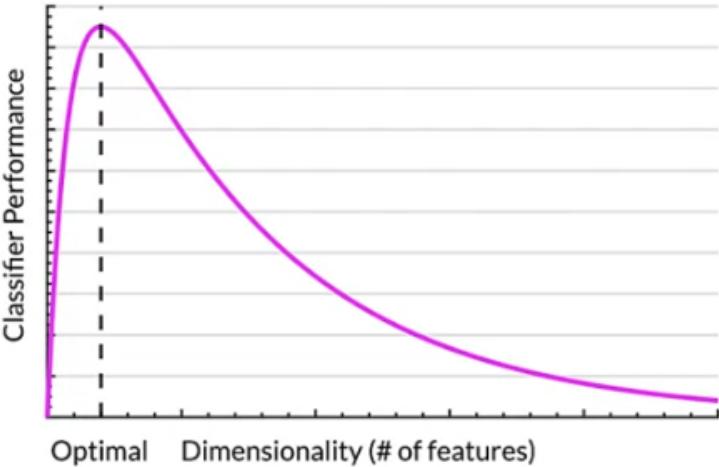
- 1. 고차원일수록 더 많은 feature가 필요
 - 2. 데이터 간의 거리가 비슷해지면서 클러스터링이 어려움
 - 3. 명확하게 나눠지지 않음
- 모바일, IoT 및 임베디드 애플리케이션 수요 증가에 따른 경량화 필요

차원 축소

II 많은 Feature 단점

- 중복되고 관계성이 높은 feature가 존재할 확률이 높다.
- Signal에 noise가 추가될 확률이 높다.
- 해석 및 시각화가 어렵다.
- 데이터 저장 및 처리가 어렵다.
- Feature 추가 시, 일정 수준까지 성능이 증가하다가 수준을 넘어서면 성능이 급격하게 감소한다.
 - 실행 시간과 시스템 메모리 요구사항 급증
 - Local minimum에 갇힐 위험성 증가
 - 상관관계가 높은 feature 일 가능성 증가
- 고차원일수록 데이터 간의 거리(Euclidean Distance)가 멀어진다. → feature space 밀도 감소 → 평균 거리를 유지하기 위해서는 더 많은 데이터가 필요하다.

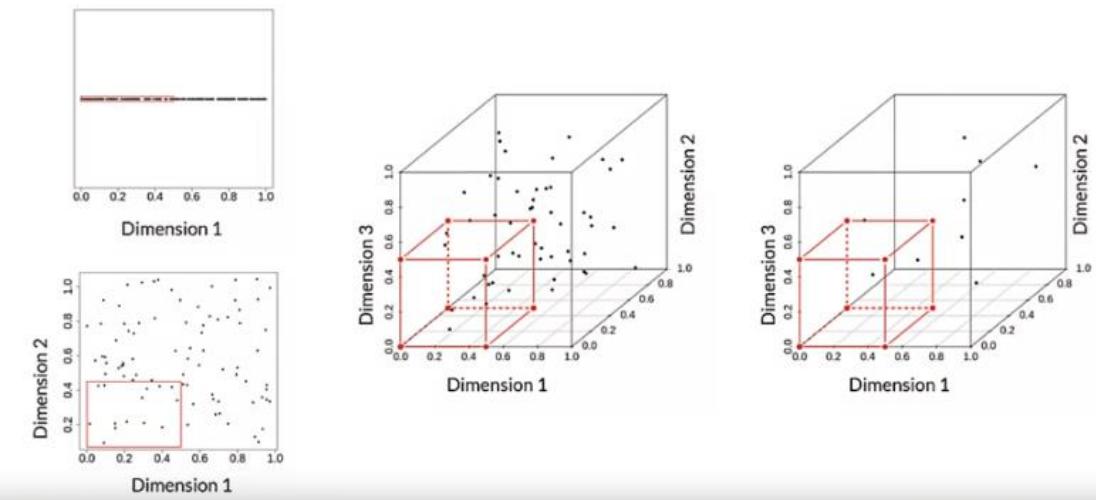
<Feature 수에 따른 성능 >



차원 축소

II 많은 Feature 단점

- 중복되고 관계성이 높은 feature가 존재한다.
- Signal에 noise가 추가될 확률이 높다
- 해석 및 시각화가 어렵다.
- 데이터 저장 및 처리가 어렵다.
- Feature 추가 시, 일정 수준까지 성능이 증가하다가 수준을 넘어서면 성능이 급격하게 감소한다.
 - 실행 시간과 시스템 메모리 요구사항 급증
 - Local minimum에 갇힐 위험성 증가
 - 상관관계가 높은 feature 일 가능성 증가
- 고차원일수록 데이터 간의 거리(Euclidean Distance)가 멀어진다. → feature space 밀도 감소 → 평균 거리를 유지하기 위해서는 더 많은 데이터가 필요하다.



차원 축소

차원의 저주

Curse of dimensionality

- : 데이터의 **직관적이지 않은 속성**이 고차원 공간에서 **관찰**되는 상황
- : 인간은 **여러 차원에 걸쳐있는 데이터 패턴**을 분석하는 데 **익숙/능숙하지 않음**

차원 축소

차원의 저주

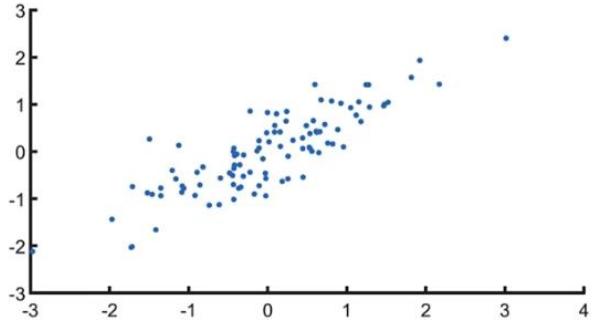
Curse of dimensionality

- : 데이터의 **직관적이지 않은 속성**이 고차원 공간에서 **관찰**되는 상황
- : 인간은 **여러 차원에 걸쳐있는 데이터 패턴**을 분석하는 데 **익숙/능숙하지 않음**

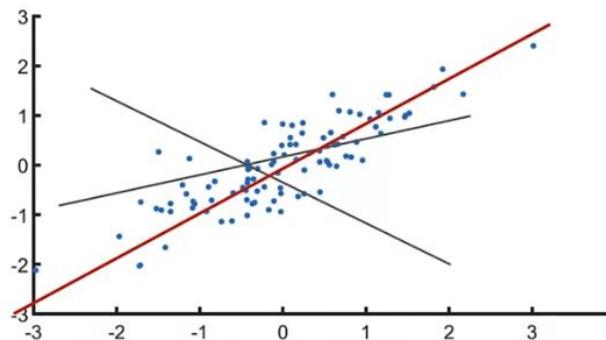


- 저장에 용이하다.
- 빠르고 효율적인 계산이 가능하다.
- 일관성이 있다.
- 효율적인 시각화가 가능하다.

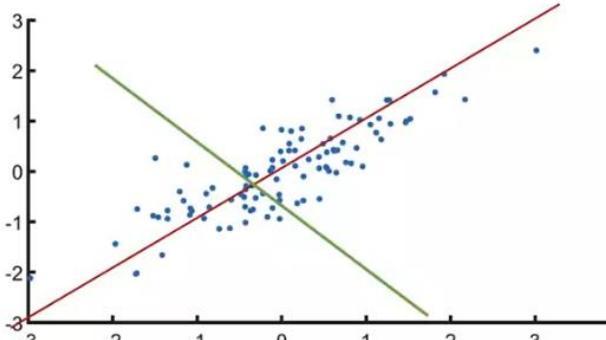
방법 II PCA



Step0. 데이터 준비

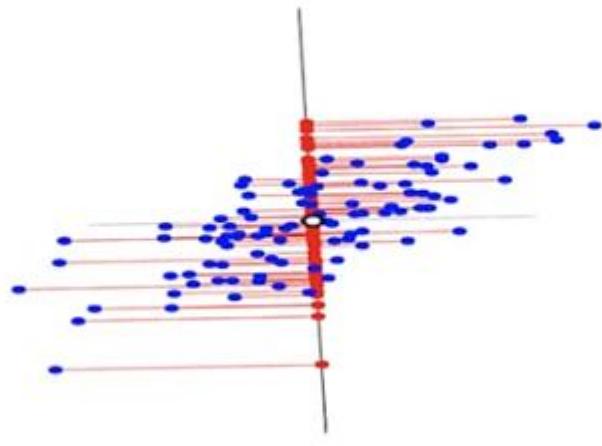


Step1. 첫번째 주성분은 데이터의 분산을 최대화하는 vector 찾기

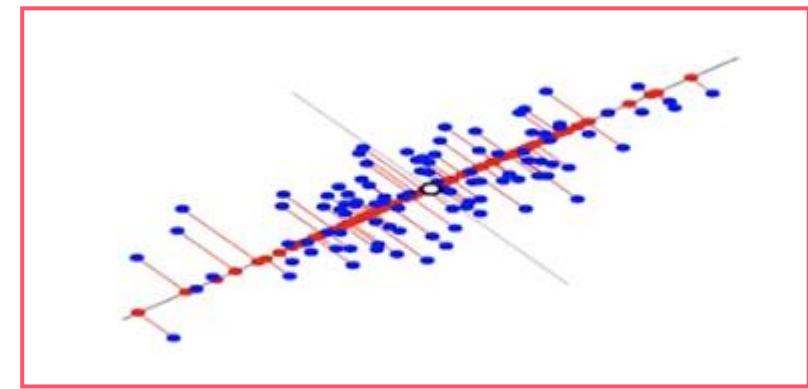


Step2. 두번째 주성분은 첫번째 주성분과 직교 + 투영된 데이터의 나머지 분산을 최대화하는 vector 찾기

방법 II PCA



VS



두 주성분을 비교했을 때 오른쪽이 데이터의 분산을 더 잘 표현함

- 빨간 점의 표현
- 파란색 데이터와 주성분 위에 투영되는 데이터 간의 거리

장점

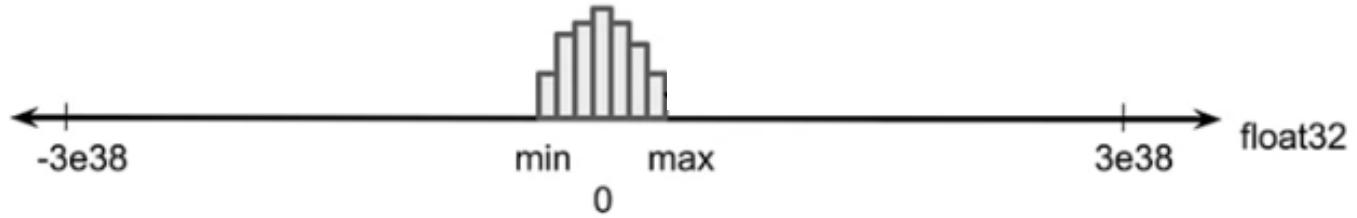
- 실제로 잘 작동하는 유연한 기술
- 빠르고 간단한 기술
- 변형 및 확장이 제공됨

단점

- 결과 해석이 어려움
- 임계값 설정 필요

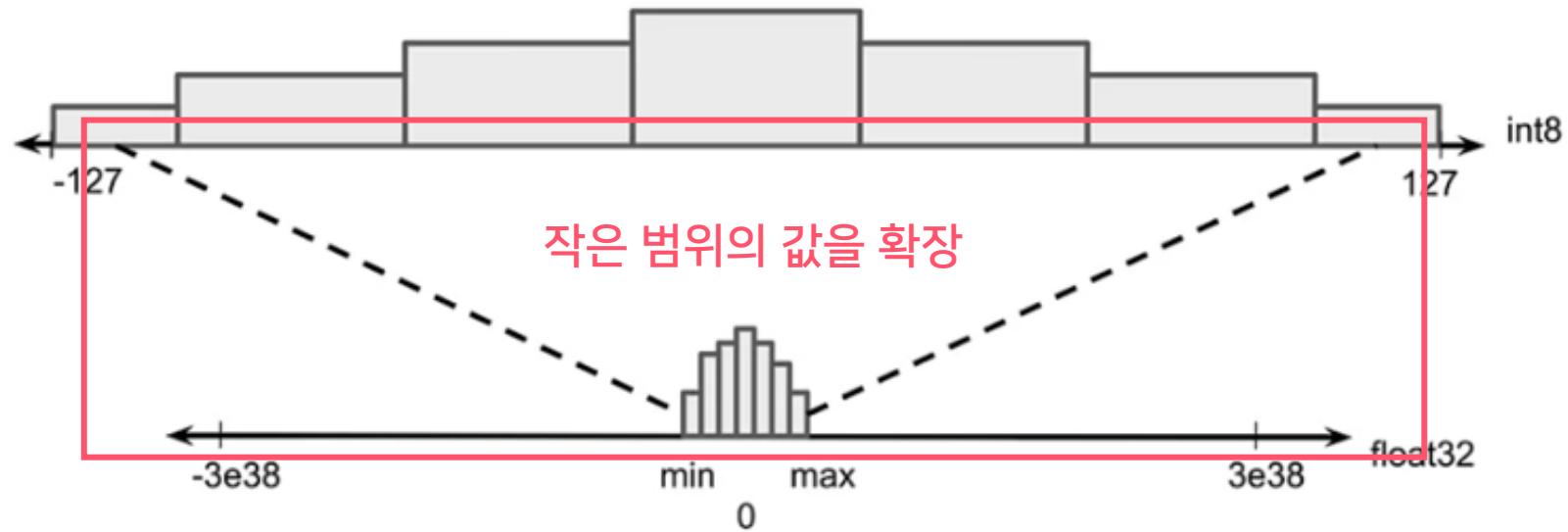
양자화

양자화 개념



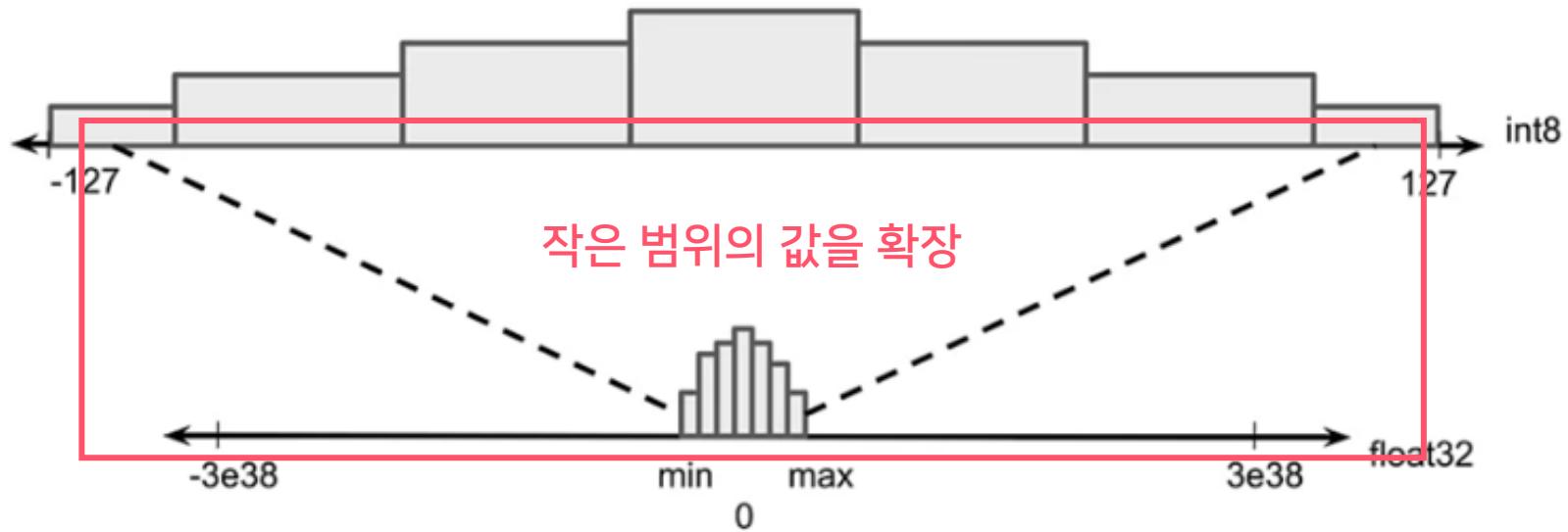
양자화

양자화 개념



양자화

양자화 개념



- 빠른 연산 가능
- 계산 리소스 축소
- 저전력
- CPU/DSP/NPU를 통한 정수 연산

약간의 정밀도 손실이 있으나 좋은 결과 추출
모델 정확도와 복잡성 간의 균형 필요

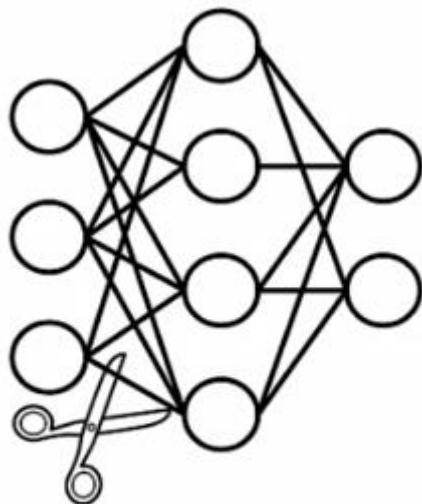
가지치기

가지치기 개념

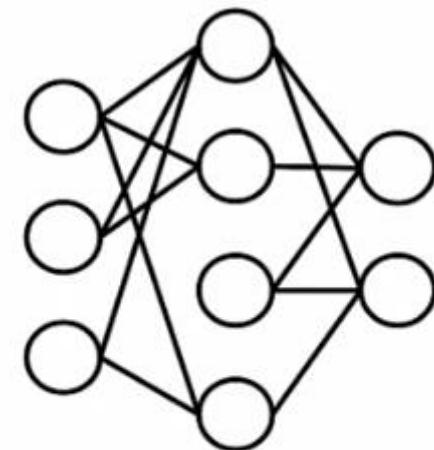
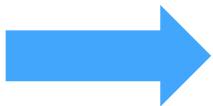
가지치기(Pruning)

개념

- : 모델의 효율성을 높이기 위해 결과 생성에 기여하지 않는 부분 모델에서 제거
- : 연결을 제거하여 예측 생성과 관련된 매개변수 수를 줄이는 것이 목표



Before pruning

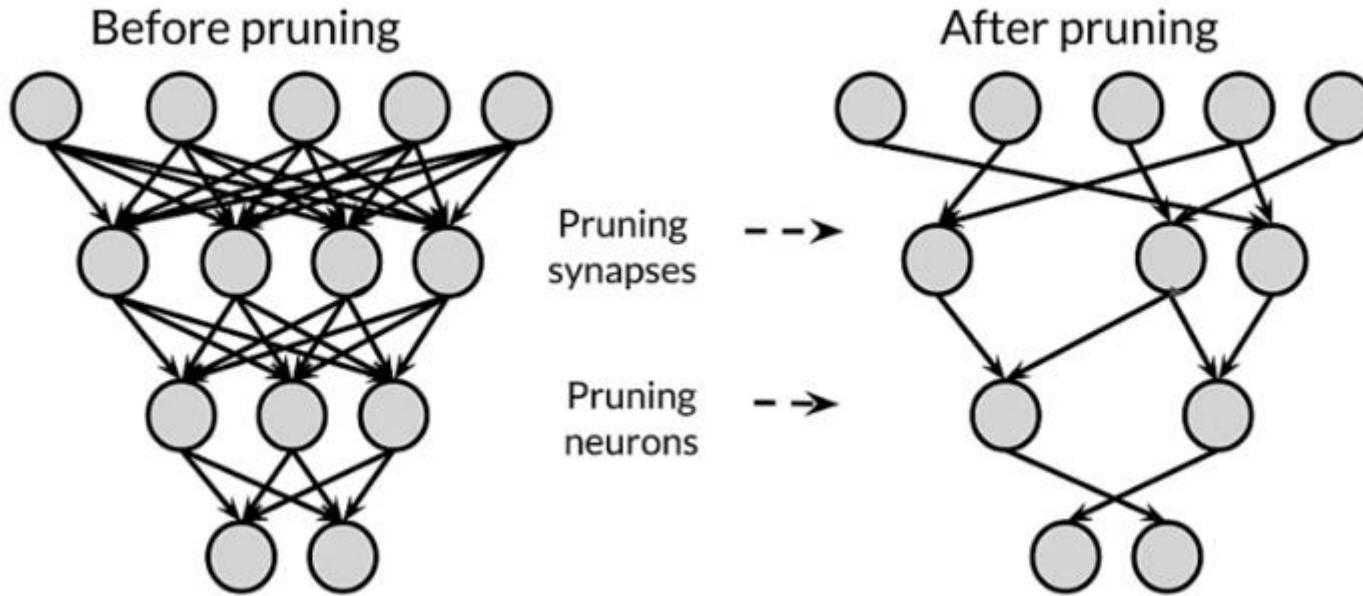


After pruning

가지치기

II

가지치기 한계점



반복적인 Pruning을 진행 시, 재훈련이 훨씬 어려움



가중치 초기화 방법이 아닌, 기존 가중치를 사용하여 네트워크를 재훈련

가지치기

가지치기 특징

저장 및 전송을 위해 모델의 크기 축소 가능

CPU 및 가속기(GPU/TPU) 속도 향상

가지치기 + 양자화를 통해 복합적인 이점 취득

압축을 통해 런타임 메모리 효율이 증가하여 성능 향상 기대 가능

MLOps[3]

Introduction to Machine Learning in Production

III

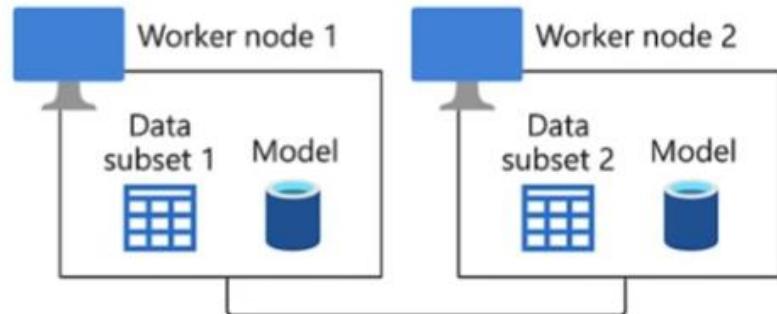
High-Performance Modeling

1. Distributed training
2. Input pipeline
3. Knowledge Distillation techniques

Distributed training

Distributed training

Worker nodes



- 각 workers 에는 전체 모델을 올리기
- 데이터를 하위 집합으로 분류하여 각 데이터에 대해 학습
- 오류를 기반으로 모델 업데이트를 할 수 있도록 변경사항을 다른 workers에게 전달 → 일관된 모델 훈련 가능

Model parallelism

- 동일한 데이터를 사용함
- 모델을 workers 간 분할
 - 각 layer의 복잡성 분석하여 공평하게 작업 분할
- 각 workers는 정방향/역방향 프로세스 수행
- 프로세스 완료
- 업데이트된 모델 가중치를 다른 장치와 동기화

Input pipeline

비효율적인 ETL 프로세스

Extract
추출



Transform
변환

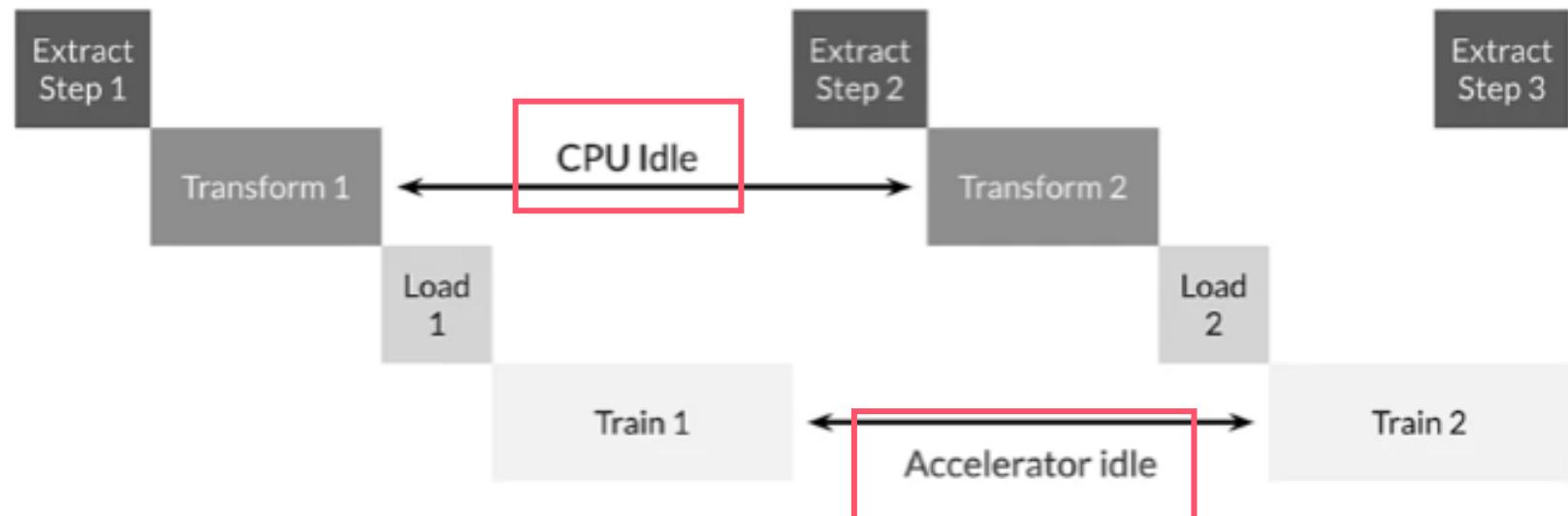


Load
적재



Input pipeline

비효율적인 ETL 프로세스



Input pipeline

III 개선된 ETL 프로세스



컴퓨터 리소스 효율성 향상



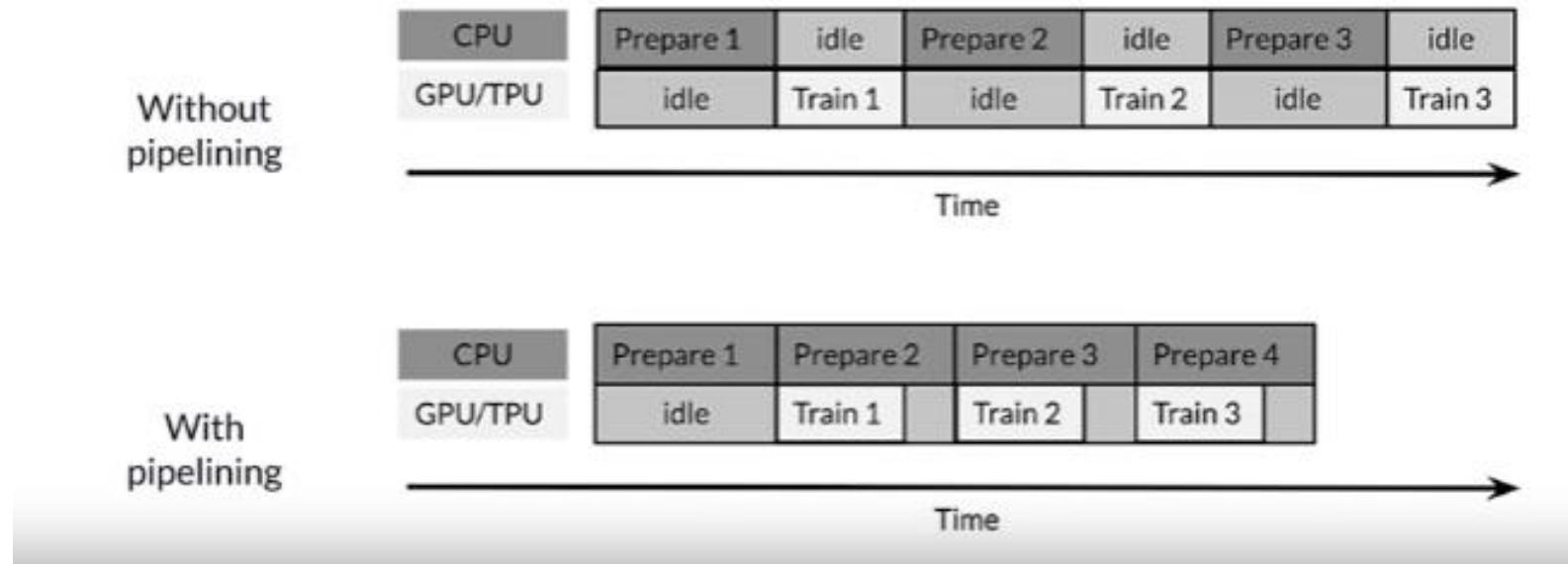
교육 속도 향상

Input pipeline

III

파이프라이닝을 통한 시간 단축

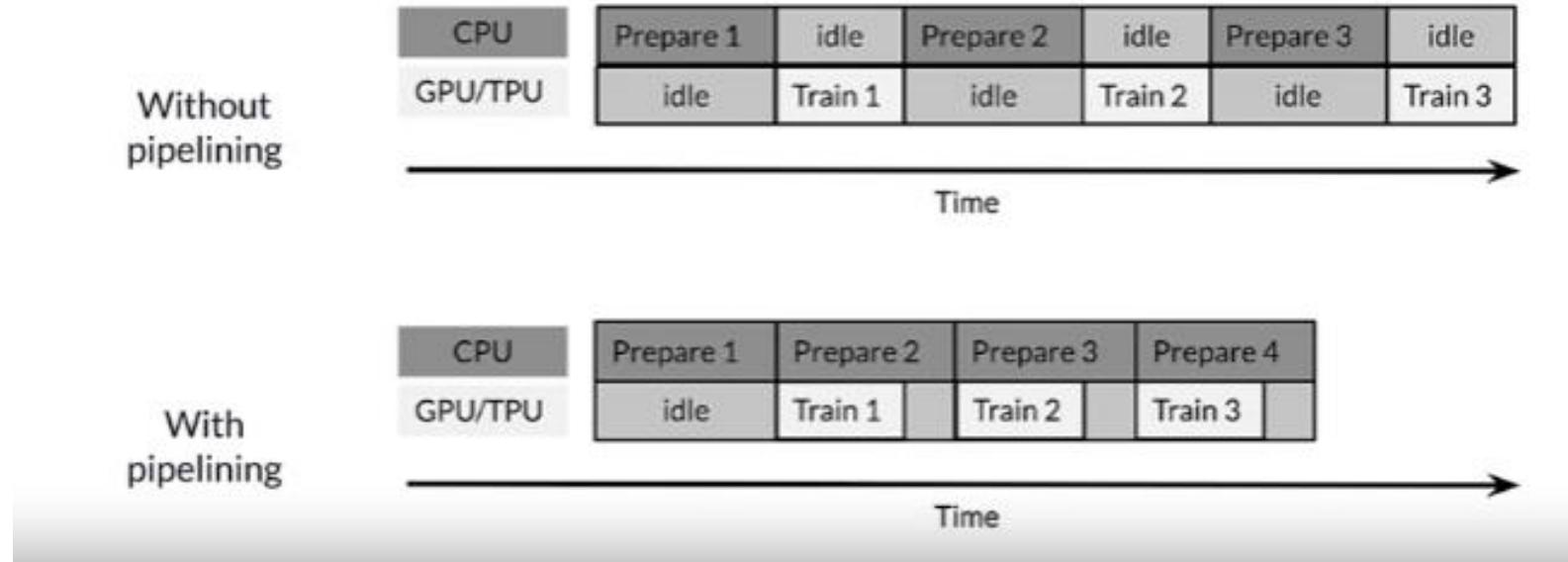
Pipelining: 하나의 명령어가 실행되는 도중에 다른 명령어를 실행 → 동시에 여러 명령어를 실행 및 처리



Input pipeline

III 파이프라인 성능 최적화 방법

Pipelining: 하나의 명령어가 실행되는 도중에 다른 명령어를 실행 → 동시에 여러 명령어를 실행 및 처리



Prefetching

하드 쿠어 → CPU

Parallelize data extraction and transformation

Caching

Reduce memory

Knowledge Distillation techniques

지식 증류 방법

Knowledge Distillation

1) 개념

: 간결한 모델에서 학습한 지식을 캡처하거나 추출하려고 시도 하는 것

2) 방법론



Teacher



Student

Knowledge Distillation techniques

지식 증류 방법

Knowledge Distillation

1) 개념

: 간결한 모델에서 학습한 지식을 캡처하거나 추출하려고 시도 하는 것

2) 방법론



모델의 정확도를 최대화하려는 목적
“난 선생님이니까 정답을 알려줘야 돼.”

Knowledge Distillation techniques

지식 증류 방법

Knowledge Distillation

1) 개념

: 간결한 모델에서 학습한 지식을 캡처하거나 추출하려고 시도 하는 것

2) 방법론



Teacher의 예측 분포를 일치시키려는 목적
“난 학생이니까 선생님의 가르침을 배워야지.”

Knowledge Distillation techniques

지식 증류 방법

Knowledge Distillation

1) 개념

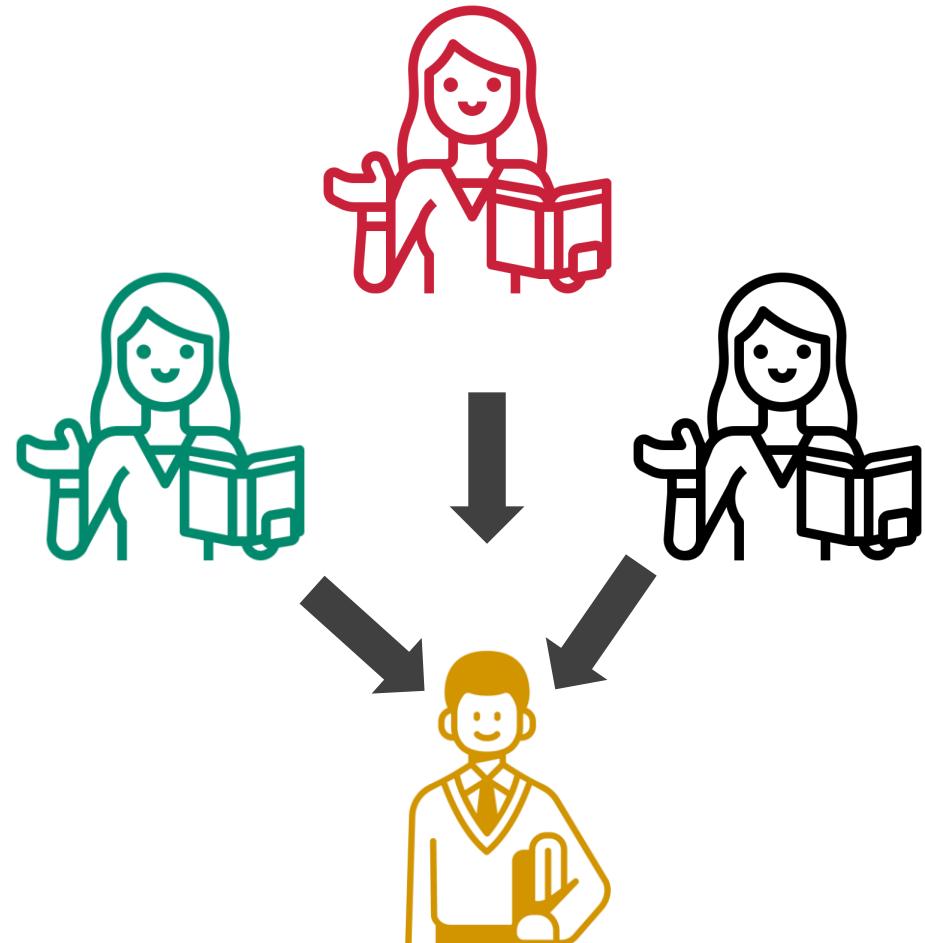
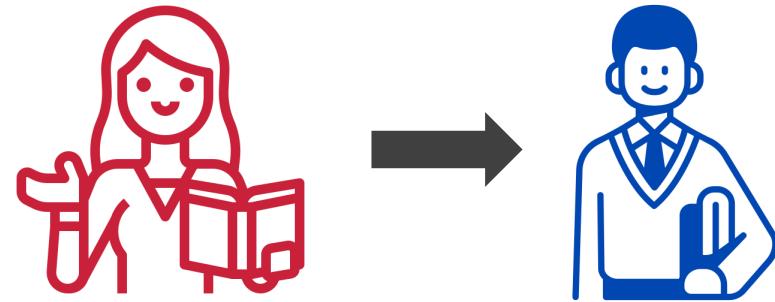
: 간결한 모델에서 학습한 지식을 캡처하거나 추출하려고 시도 하는 것

2) 방법론



Knowledge Distillation techniques

지식 증류 방법



MLOps[3]

Introduction to Machine Learning in Production

IV

Model Analysis

1. Model performance
2. Model debugging

Model performance

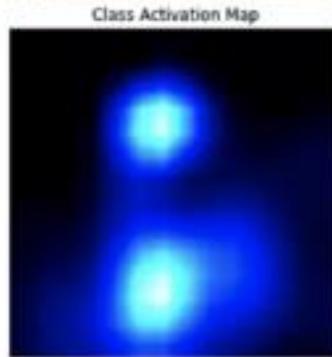
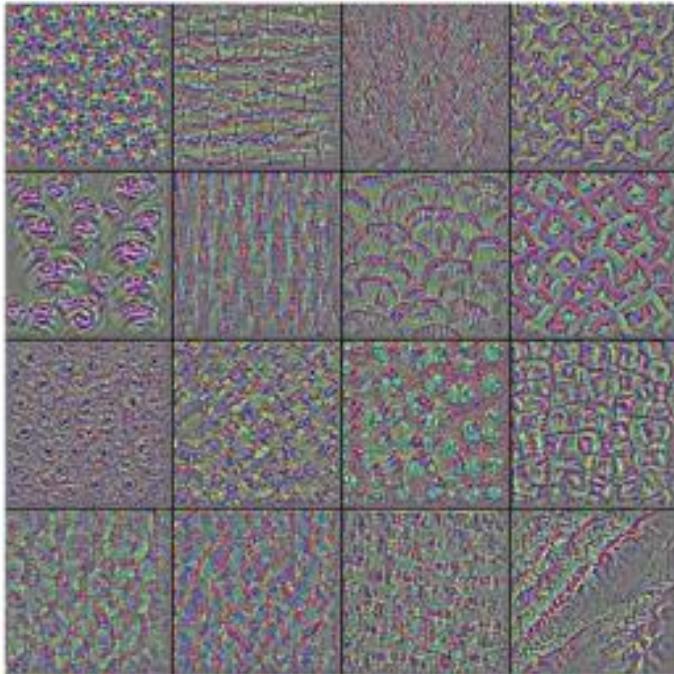
모델 성능 분석 방법

- 블랙 박스 평가
 - 모델 내부구조 고려 X
 - Metric, loss를 통해 모델 성능 수량화
- 모델 내부 검사
 - 내부적으로 작동하는 방식에 초점
 - 데이터 내부의 흐름 파악

Model performance

모델 성능 분석 방법

Model introspection



Model performance

Slice

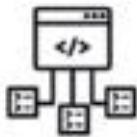
Slice

데이터 하위 집합과 관련된 문제를 찾고자,
데이터를 Slice 하여
모델 성능에 대한 심층 분석을 수행한다.

Model performance

TFMA 진행 파이프라인

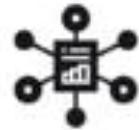
TensorFlow Model Analysis(TFMA)



Scalable
framework



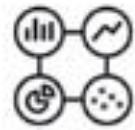
Open source
library



Ensures models
meet required
quality
thresholds



Used to compute
and visualize
evaluation
metrics



Inspect model's
performance
against different
slices of data

Model debugging

Robustness

Robustness

- 모델의 견고성
- 하나 이상의 feature가 크게 변해도 일관된 결과 생성
- 데이터가 변경됨에 따라 다른 결과를 생성

Model debugging

모델 디버깅

Robustness



Model's Robustness 향상

(모델 투명성, 사회적 차별, 공격, 개인정보보호, 데이터 분포 변경 영향)

Model debugging

- 모델에서 문제를 찾고 수정하여 모델의 Robustness 향상
- 모델 위험 관리, 기존 모델 진단 및 소프트웨어 테스트에서 다양한 방식 차용
- ML 시스템 정확성, 공정성, 보안 및 기타 문제 감지 후 수정

Model debugging

모델 디버깅 기술

Benchmark models	<ul style="list-style-type: none"> 같은 문제를 다루는 모델과 비교 → baseline 제시 벤치마크 모델 성능과의 비교 <ul style="list-style-type: none"> 개발한 모델의 성능이 낮을 경우 → 모델 문제, 데이터 모델링 확인 개발한 모델의 성능이 동일할 경우 → 벤치마크 모델이 견고한 디버깅 도구로 사용
Sensitivity analysis	<ul style="list-style-type: none"> 각 feature가 예측에 미치는 영향 조사 다른 feature를 일정하게 유지하면서 단일 feature 값을 변경하여 실험하고 모델 결과를 관찰
Random attacks	<ul style="list-style-type: none"> 무작위 입력 데이터를 생성하고 모델 출력 테스트 모든 종류의 예상치 못한 SW 및 버그 드러낼 수 있음 시작점을 설정하기 어려울 경우 사용하면 좋은 방법
Partial dependence plots	<ul style="list-style-type: none"> 하나 또는 두 개의 feature 한계 효과, 모델 결과에 미치는 영향 보여줌 레이블과 특정 feature 간의 관계가 선형 단조인지 더 복잡한지 여부 표시 가능
Residual analysis	<ul style="list-style-type: none"> 모델이 예측한 결과와 GT 간 차이 측정 무작위 분포 error에 좋음 상관관계 존재(예측 정보를 캡처하지 못한 것) → 모델이 개선될 수 있다는 신호

Model debugging

모델 공격

모델이 공격에 얼마나 취약한가?

- 공격의 목표: 모델을 바보로 만들자!
- 성공적인 공격은 매우 치명적
- 적대적 예제 테스트
- 모델 강화

Model debugging

모델 공격

Information Harms

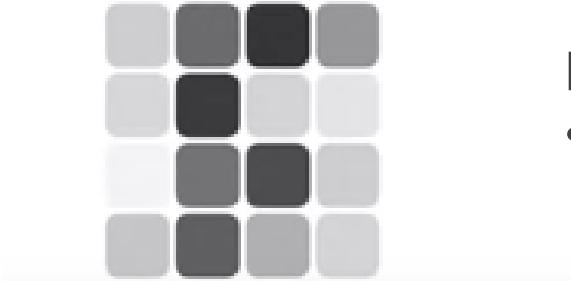
- 의도하지 않거나 예상하지 못한 정보 유출
- Membership inference: 결과를 통해 개인 정보 추측
- Model inversion: 모델 출력을 사용하여 훈련 데이터 재생성 → 개인 얼굴 이미지 재구성
- Model extraction: 모델 출력을 사용하여 모델 자체 재생성 → 개인 정보 보호 및 보안은 물론 모델 자체의 재산 손상

Behavioral Harm

- 모델 자체의 동작을 조작하여 예측/결과에 부정적인 영향을 미침
- Poisoning: 모델 행동을 변경하기 위해 악성 데이터를 훈련 데이터에 삽입하여 발생 시킴
- Evasion: 모델이 데이터를 잘못 분류하도록 유도하는 입력 데이터

Model debugging

모델 취약성 측정



Cleverenans

- 모델을 벤치마킹하여 적대적 사례에 대한 취약성 측정

Foolbox

- 심층 신경망과 같은 기계 학습 모델에 대한 적대적 공격을 수행하여 취약성 측정

MLOps[3]
Introduction to Machine Learning in Production



v _ Interpretability

1. Explainable AI
2. Feature가 모델에 미치는 영향

Explainable AI

V Explainable AI 개념

Explainable AI: 설명가능한 AI

- 복잡해지는 모델에 따라 해석 가능성에 대한 중요도 상승
- 공정성 보장
- 훈련 데이터
- 규제 및 법률
- 편향된 문제 찾기
- 최상의 결과를 생성하기 위해 최적화하는 모델 내부 연구

Explainable AI

설명가능성의 중요성

- 모델이 내린 결과를 설명해야 하기 때문에
- 공정성
- 평판과 브랜딩
- 법적 및 규제적 문제
- 실제 y 값과 유사하지 않은 결과를 생성할 때

MLOps[4]

Deploying Machine Learning Models in Production

I

Model Serving: Introduction

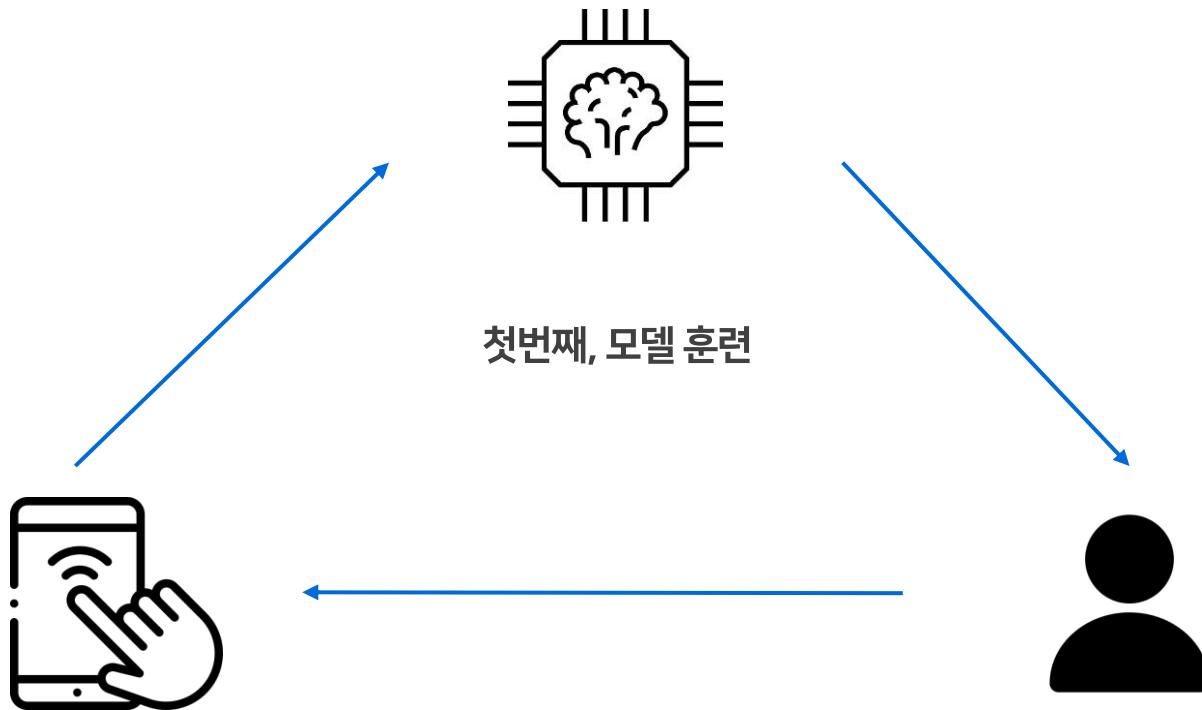
— Model Serving

1. Model Serving
2. Model Serving Patterns
3. Model Serving 3요소
4. 배포 환경

Model Serving

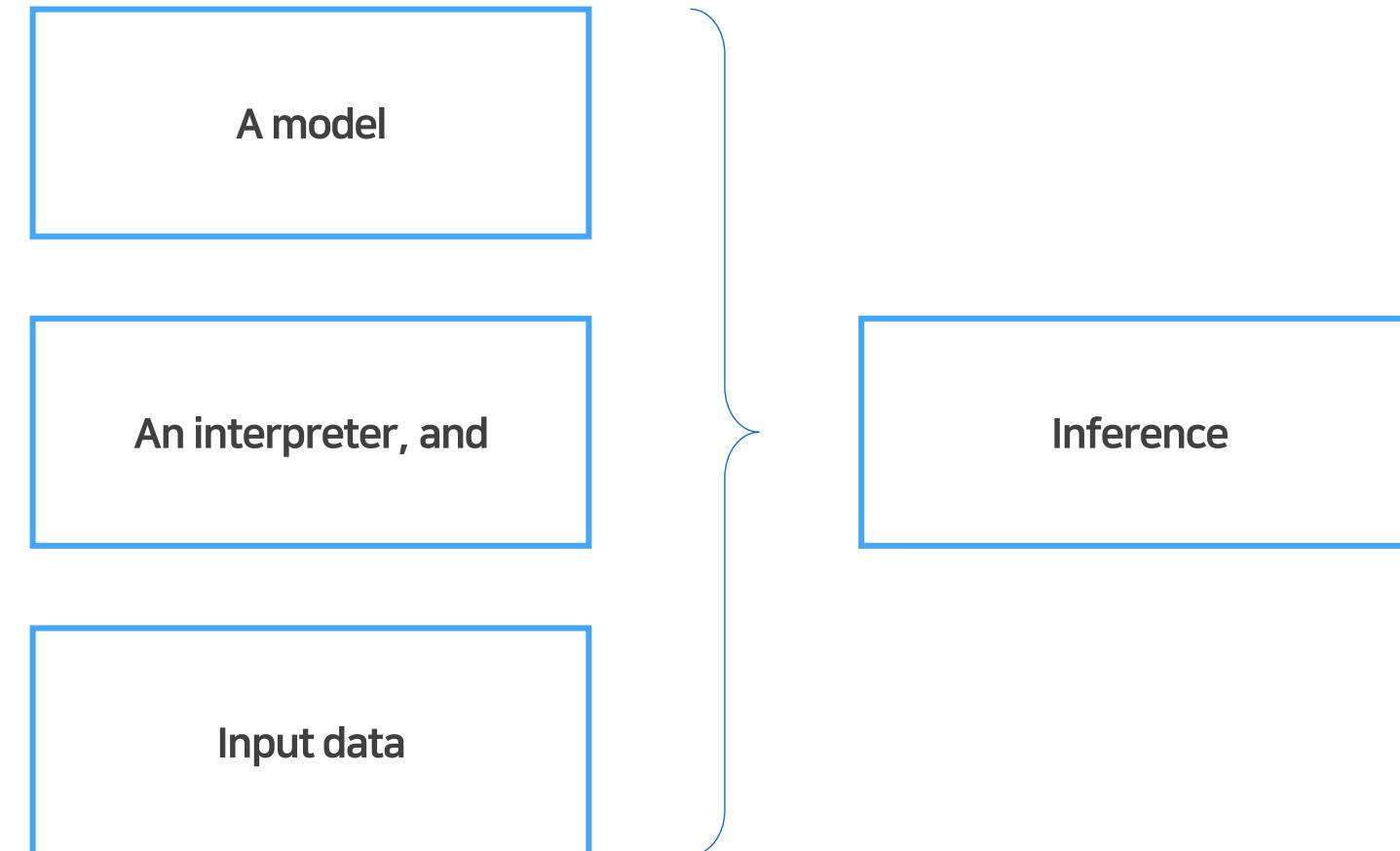
Model Serving 개념

Model Serving



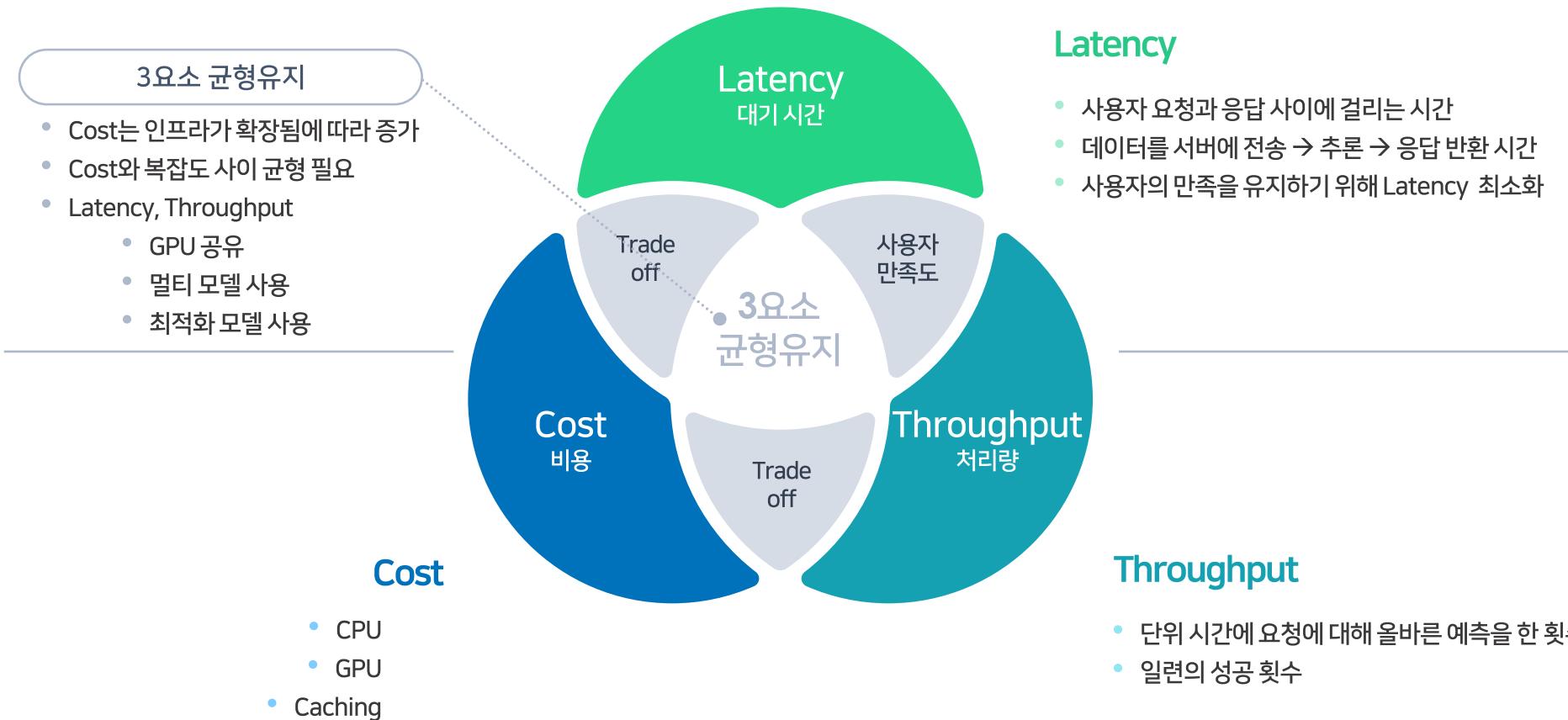
Model Serving

Model Serving Patterns



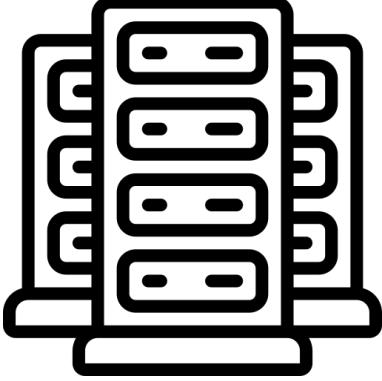
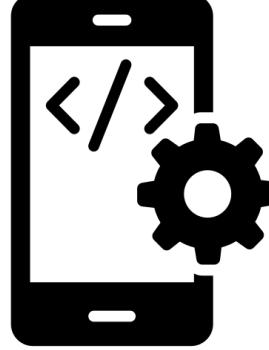
Model Serving

Model Serving 3요소



Model Serving

배포 환경

Huge data center	Embedded devices
	
<ul style="list-style-type: none">원격 호출을 통해 접근리소스가 비교적 보장되지만 비용과 효율성 중요리소스 활용도 개선 및 비용 절감 방법 모색	<ul style="list-style-type: none">로컬에서 사용저장 공간을 많이 차지하는 애플리케이션이라면 사용자는 설치하지 않을 것최적화된 모델 축소 중요

MLOps[4]

Deploying Machine Learning Models in Production

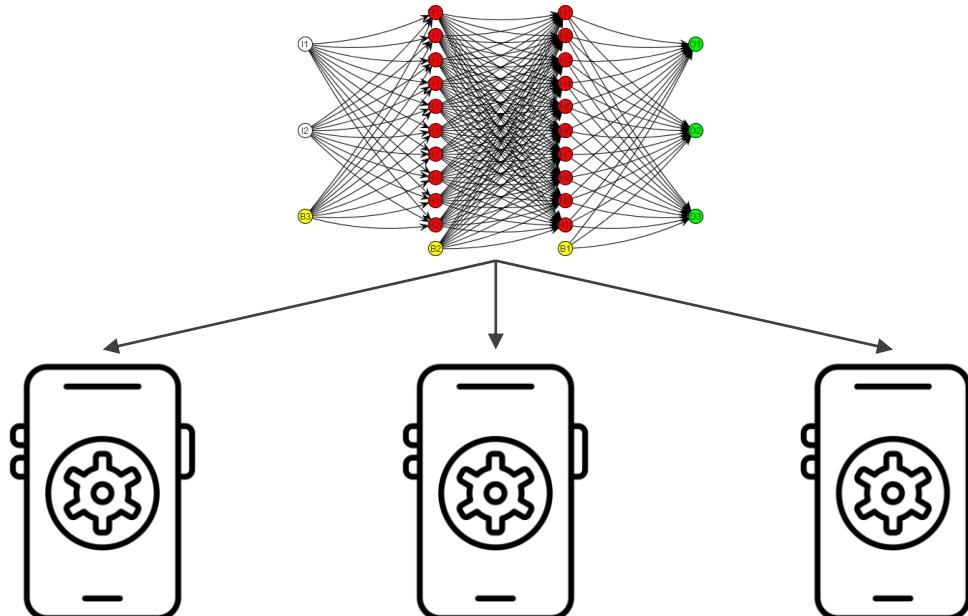
III

Model Serving: Patterns and Infrastructure

1. Scaling infrastructure

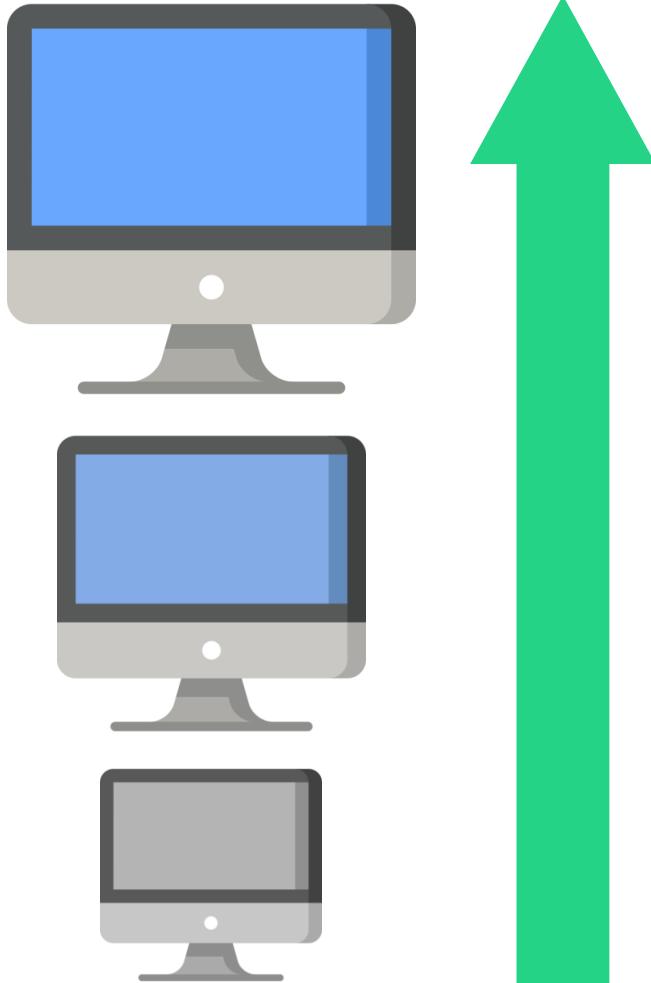
확장의 중요성

- 거대한 데이터 세트 훈련은 시간/비용이 많이 소요
- 하드웨어 확장 후 분할하여 훈련시켜 효율성 증진



Scaling Infrastructure

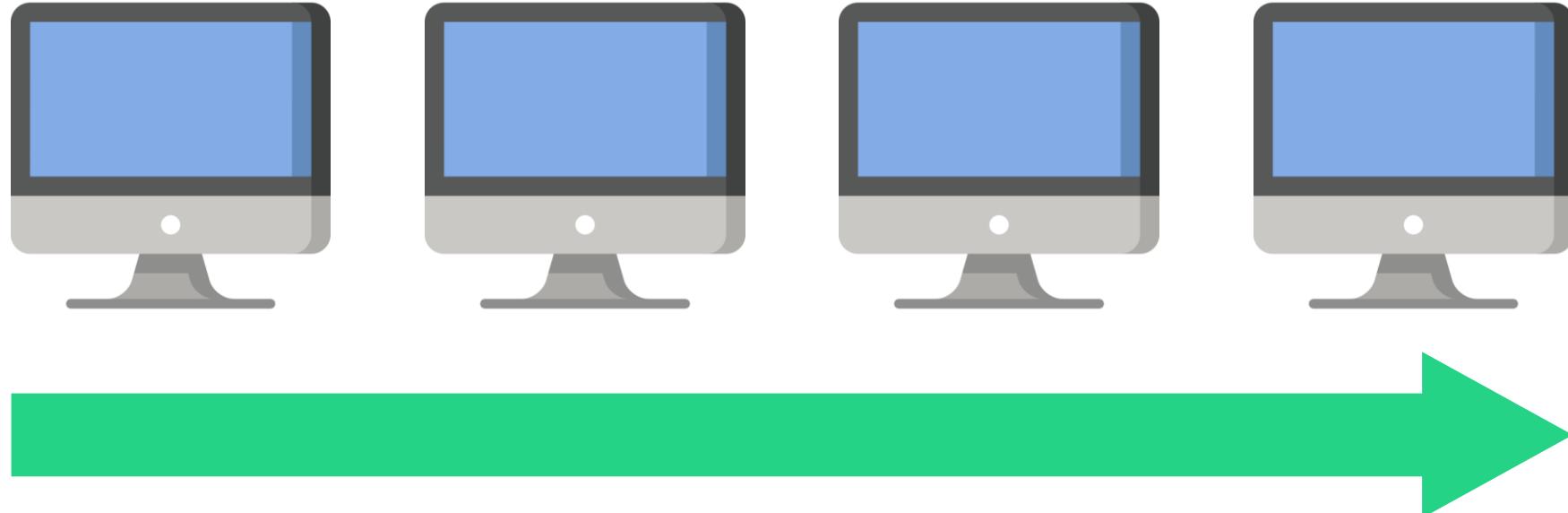
확장 방법(1): 수직 확장



- 더 크고 강력한 하드웨어 사용
- 최신 GPU
- CPU 업그레이드
- RAM 추가
- 빠른 저장소
- 차의 성능을 업그레이드
 - 기존에 5명이 탑승할 수 있는 자동차가 있음
 - 100명을 운송하기 위해서
 - 차량 탑승 인원은 변하지 않고 자동차만 업그레이드

Scaling Infrastructure

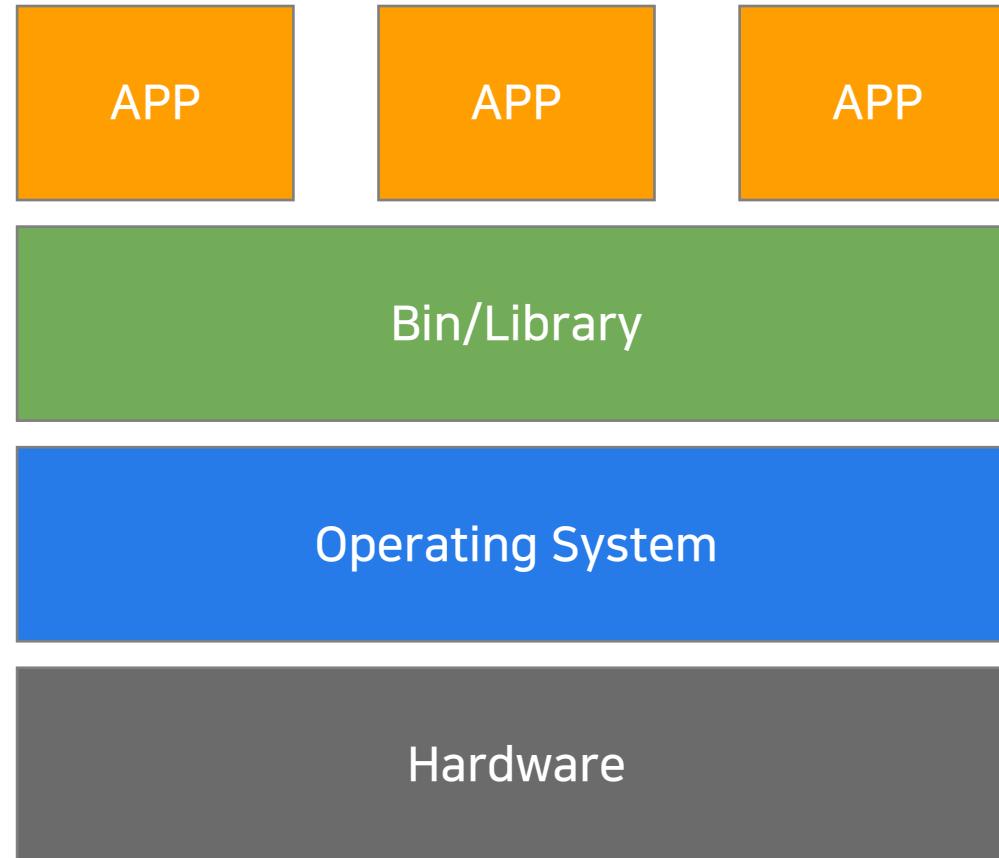
확장 방법(2): 수평 확장



- 더 많은 장치 추가
- 규모에 따라 up/down
- 일반적으로 수평 확장 권장
 - 비교적 안정적
 - 한계가 없음
- 차량 수를 업그레이드
 - 기존에 5명이 탑승할 수 있는 자동차가 있음
 - 100명을 운송하기 위해서
 - 차량 대수를 업그레이드 하여 10대 준비

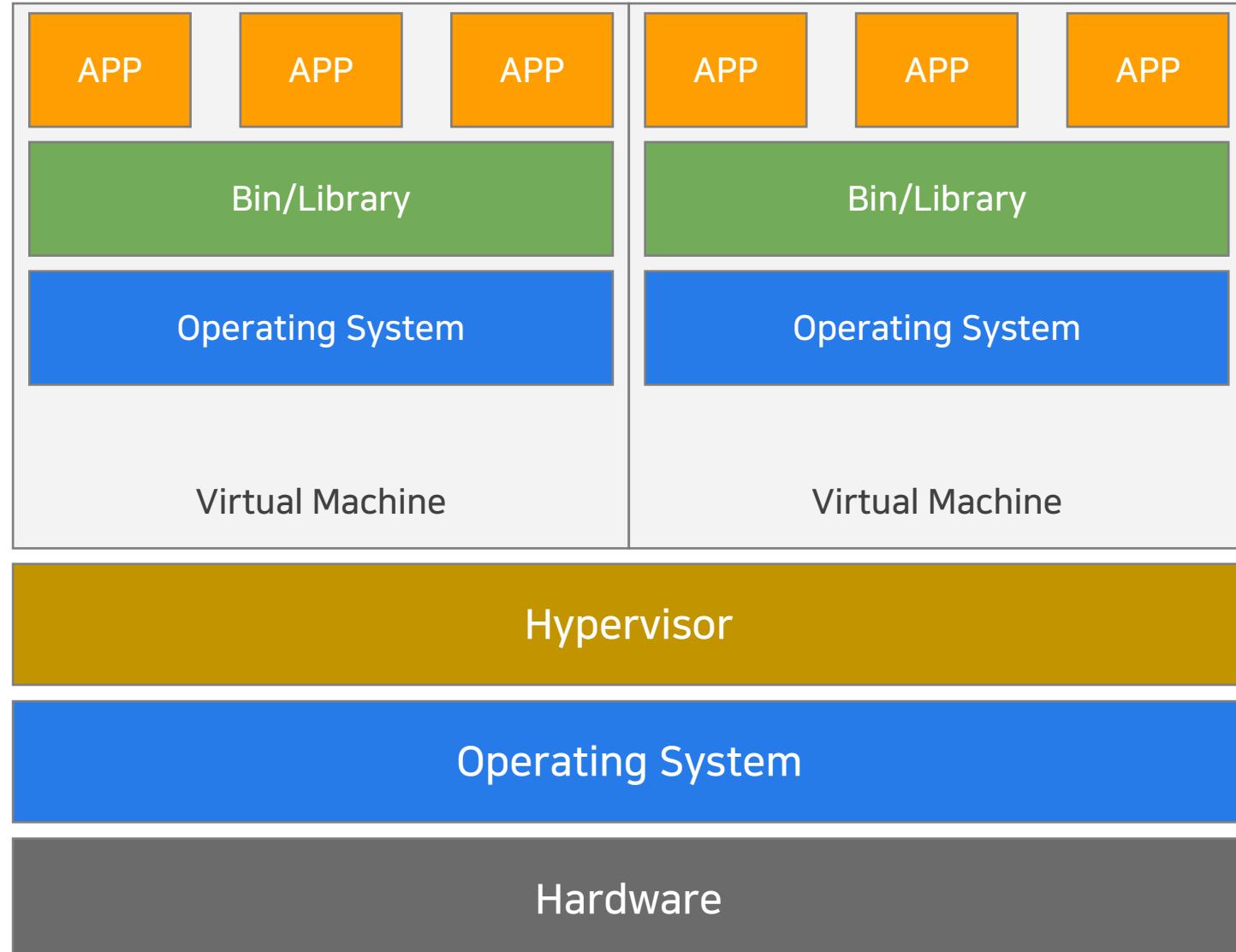
Scaling Infrastructure

II Typical System Architecture



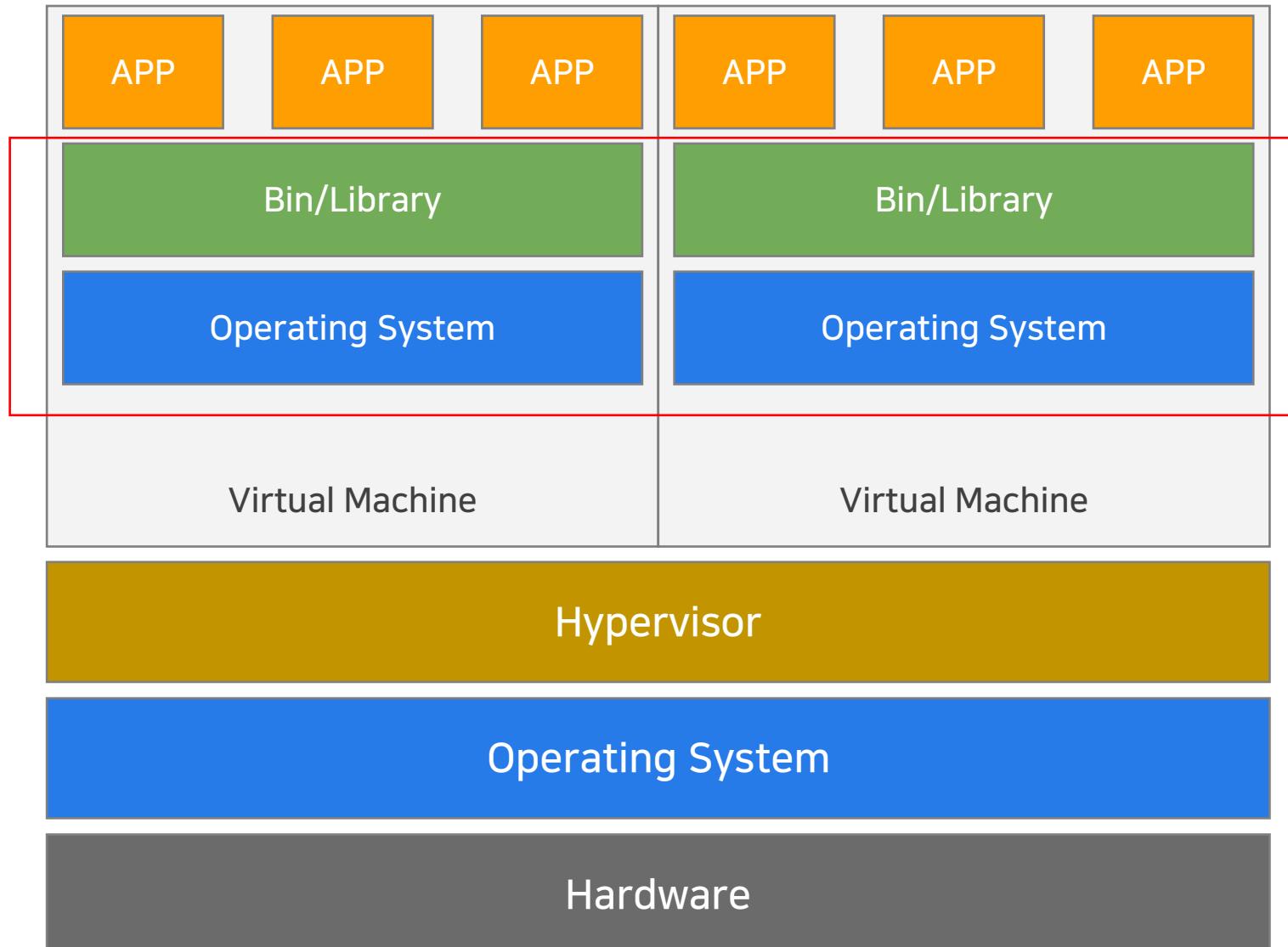
Scaling Infrastructure

VM Architecture



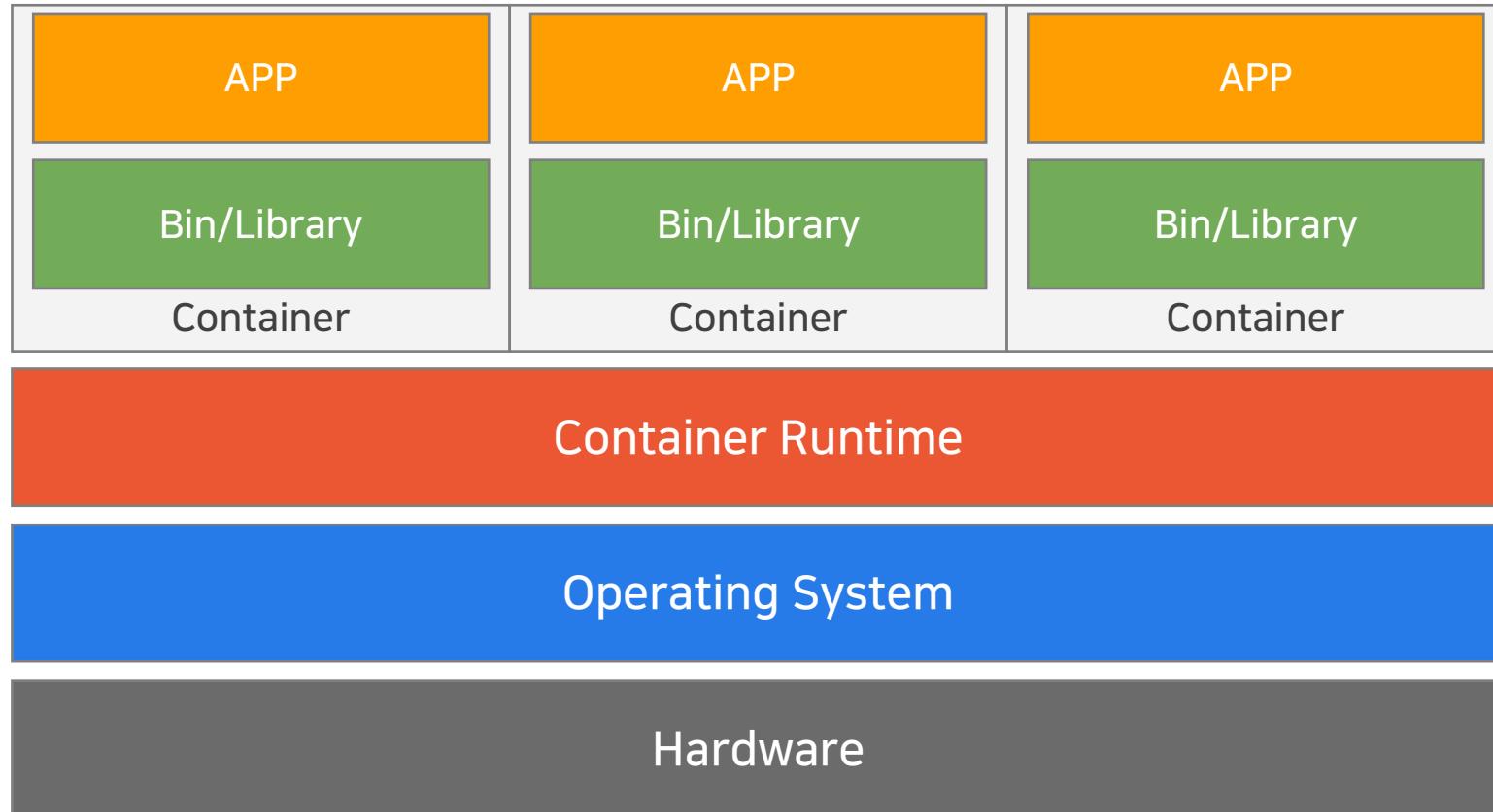
Scaling Infrastructure

VM Architecture



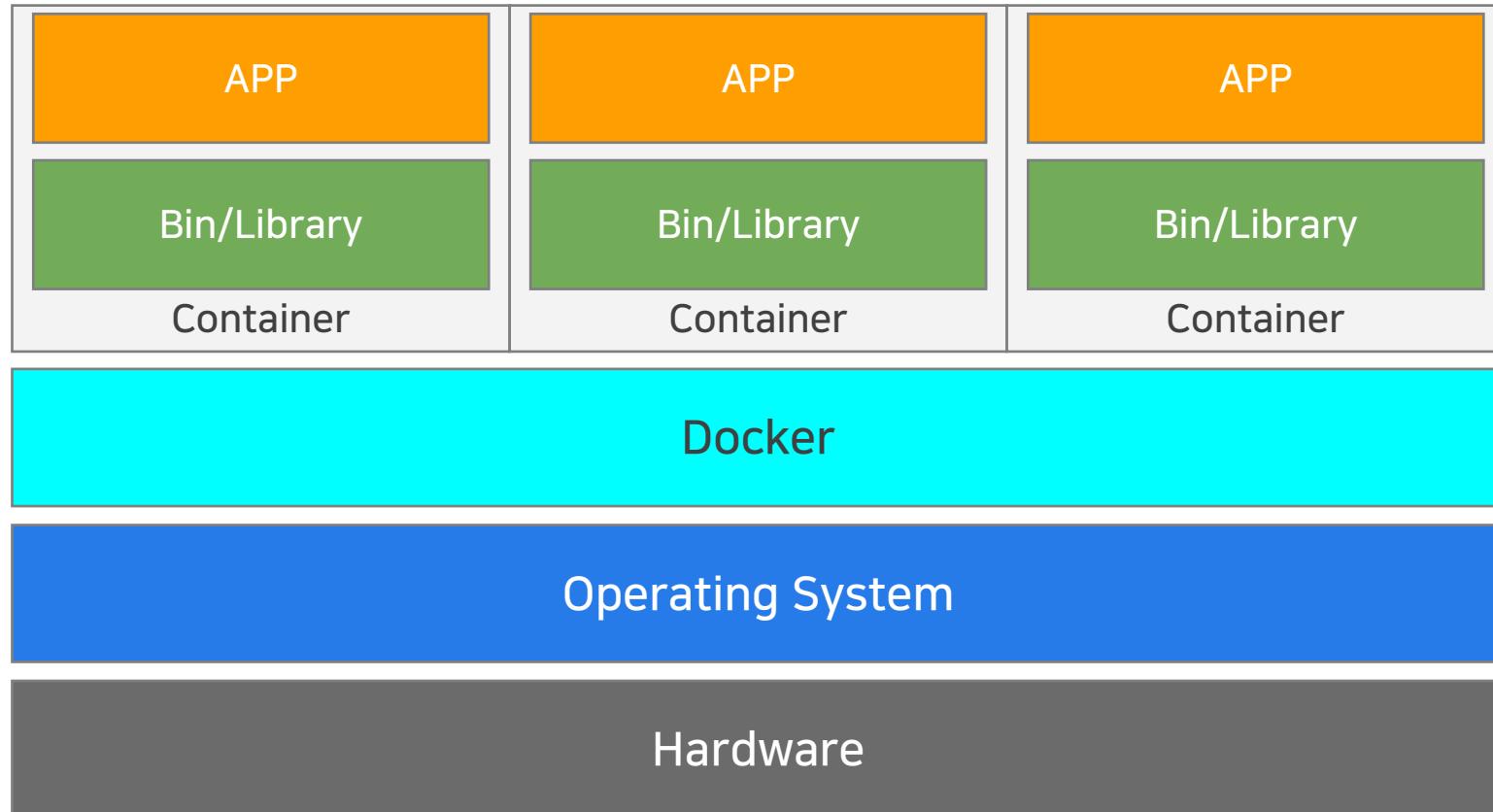
Scaling Infrastructure

Building Containers



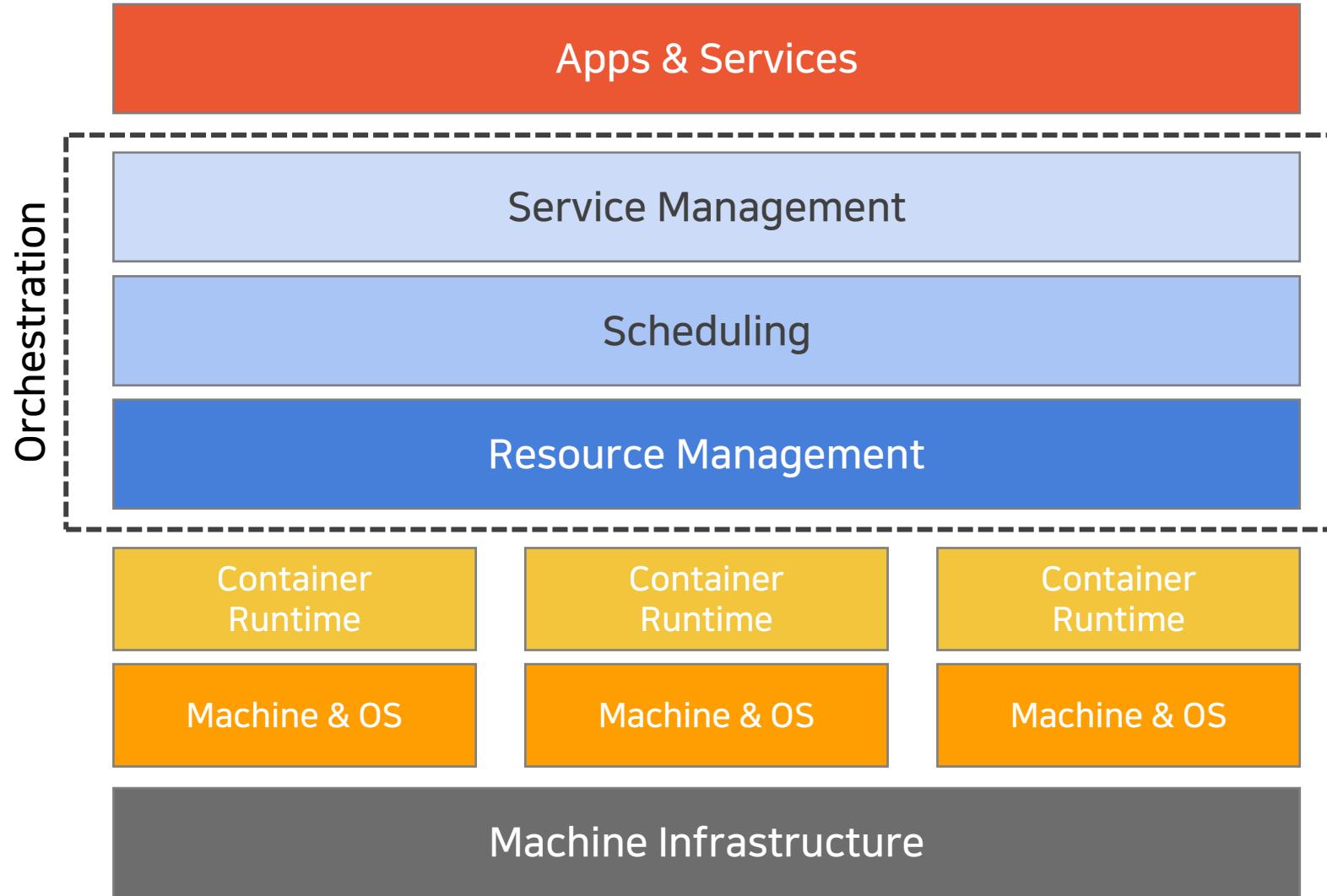
Scaling Infrastructure

Docker: Container Runtime



Scaling Infrastructure

Enter Container Orchestration



MLOps[4]

Deploying Machine Learning Models in Production

III

Model Management and Delivery

1. MLOps 방법론
2. Continuous delivery

MLOps 방법론

자동화 수준에 따라

성숙도	설명
MLOps level 0	<ul style="list-style-type: none"> ML 모델 구축 + 배포 프로세스가 완전히 수동 훈련된 모델을 예측 서비스로 배포하는 것에만 초점을 맞춤 성능 저하 및 drift 모니터링을 따로 진행하지 않음
MLOps level 1	<ul style="list-style-type: none"> 데이터 유효성 검사부터 모델 유효성 검사까지의 파이프라인 자동화 새로운 데이터를 사용하여 모델을 재훈련하는 프로세스 자동화를 위해 트리거, 메타 데이터 관리, 데이터 및 모델 검증 단계 도입
MLOps level 2	<ul style="list-style-type: none"> Feature engineering, model architecture, hyperparameter 을 신속하게 탐색할 수 있도록 자동화된 CI/CD 사용 Continuous Integration(CI) <ul style="list-style-type: none"> 새 코드가 소스 코드 레포지토리에 커밋/푸시 될때마다 트리거 주로 구성 요소에 대한 빌드, 패키징, 테스트 수행 테스트를 통과하면 테스트된 코드와 패키지를 파이프라인에 전달 Continuous Delivery(CD) <ul style="list-style-type: none"> 대상 환경에 새 코드와 훈련된 모델 배포 대상 환경과 코드 및 모델의 호환성을 보장 ML 배포 시 예측 서비스 성능 확인 → 새 모델이 성공적으로 제공되는지 확인

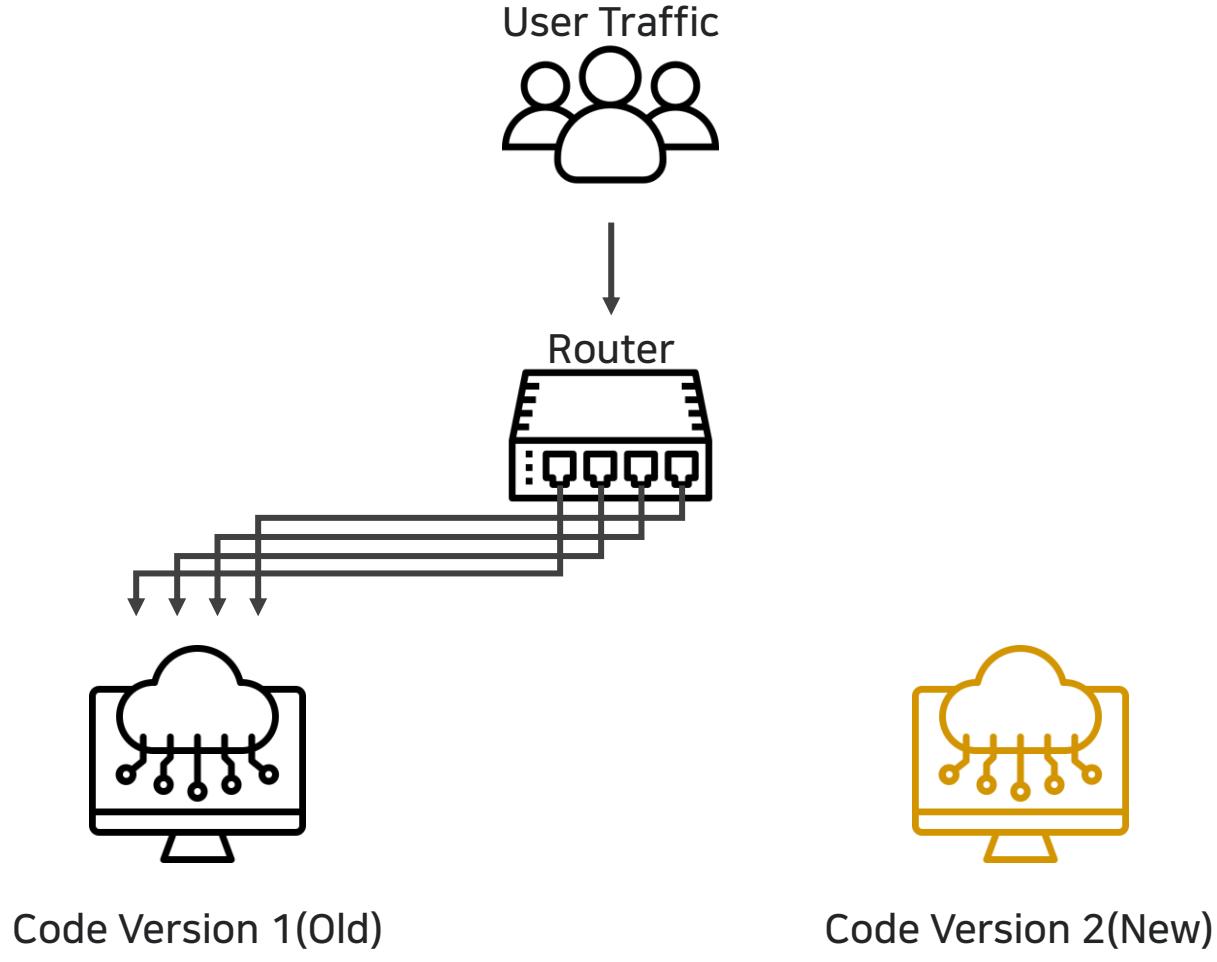
성숙도: 새 모델 교육 및 배포 속도, 느릴수록 0, 빠를수록 2

Continuous delivery

배포 스타일(1): Canary 배포

III

...

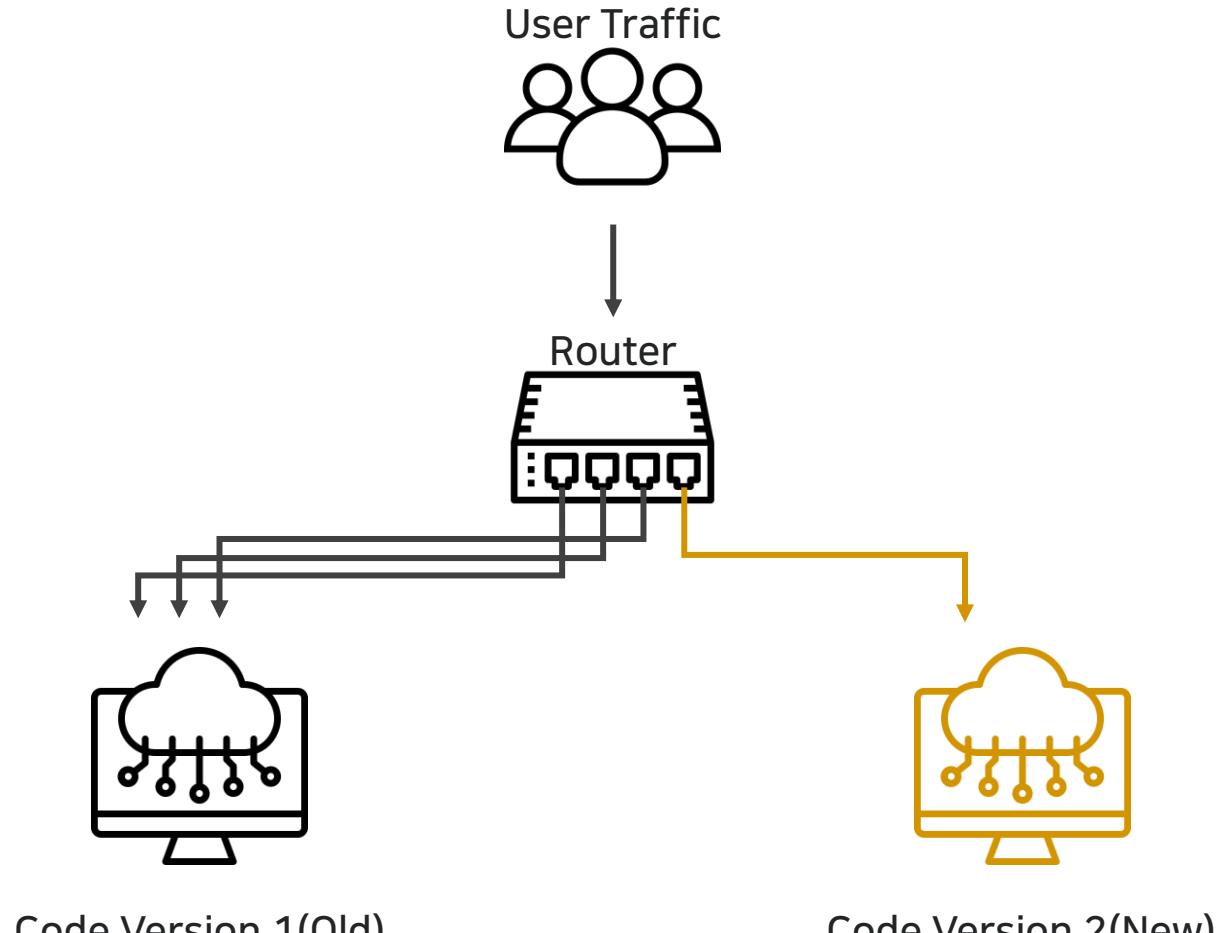


Code version 1을 사용하여 예측

Continuous delivery

배포 스타일(1): Canary 배포

III



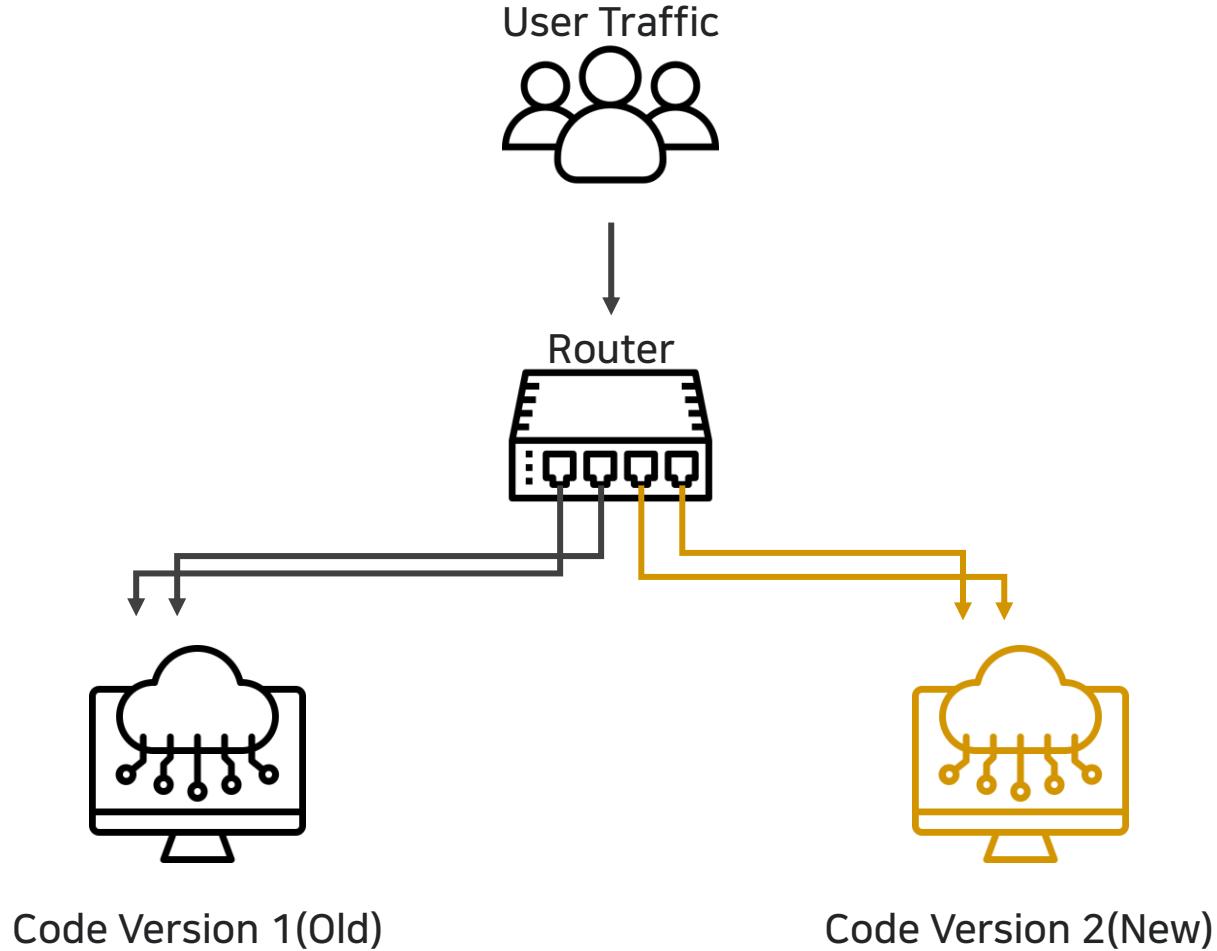
Code version 1과 함께 **작은 부분**만 Code version 2를 사용하여 예측

Continuous delivery

배포 스타일(1): Canary 배포

III

...

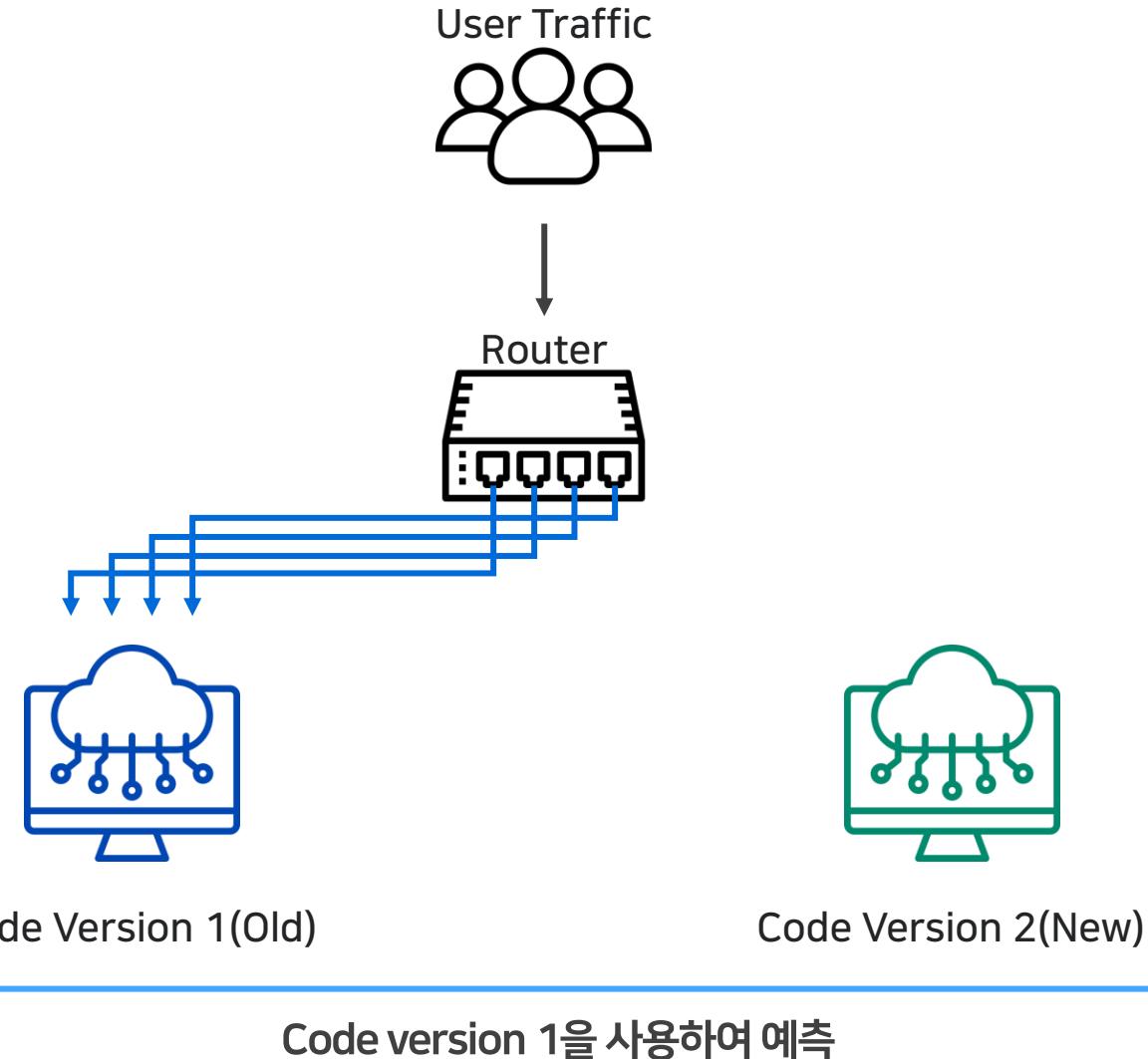


점차 확신이 생기면 Code version 2에 트래픽 비율을 높여 판단 진행

Continuous delivery

III

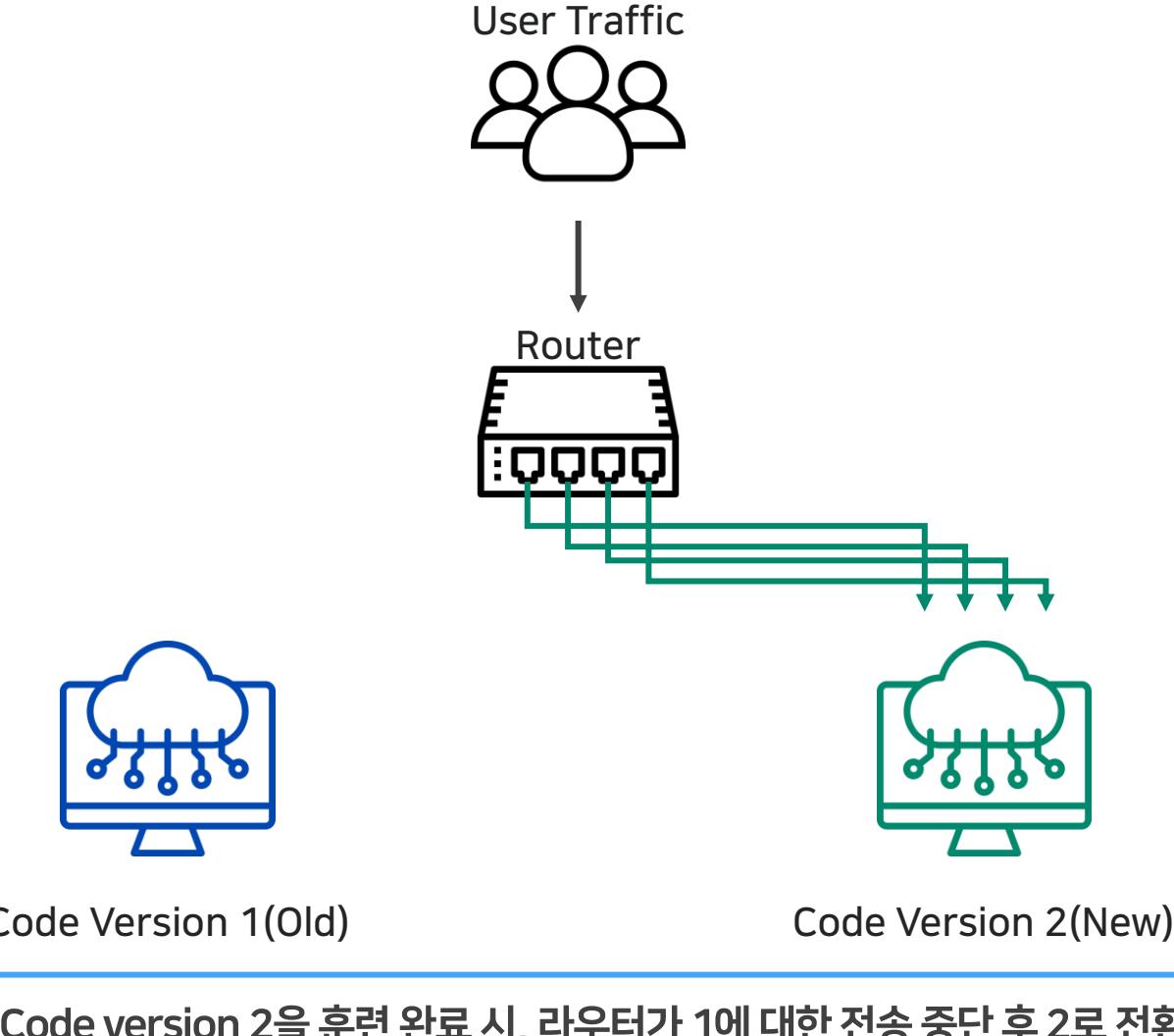
배포 스타일(2): Blue/Green 배포



Continuous delivery

III

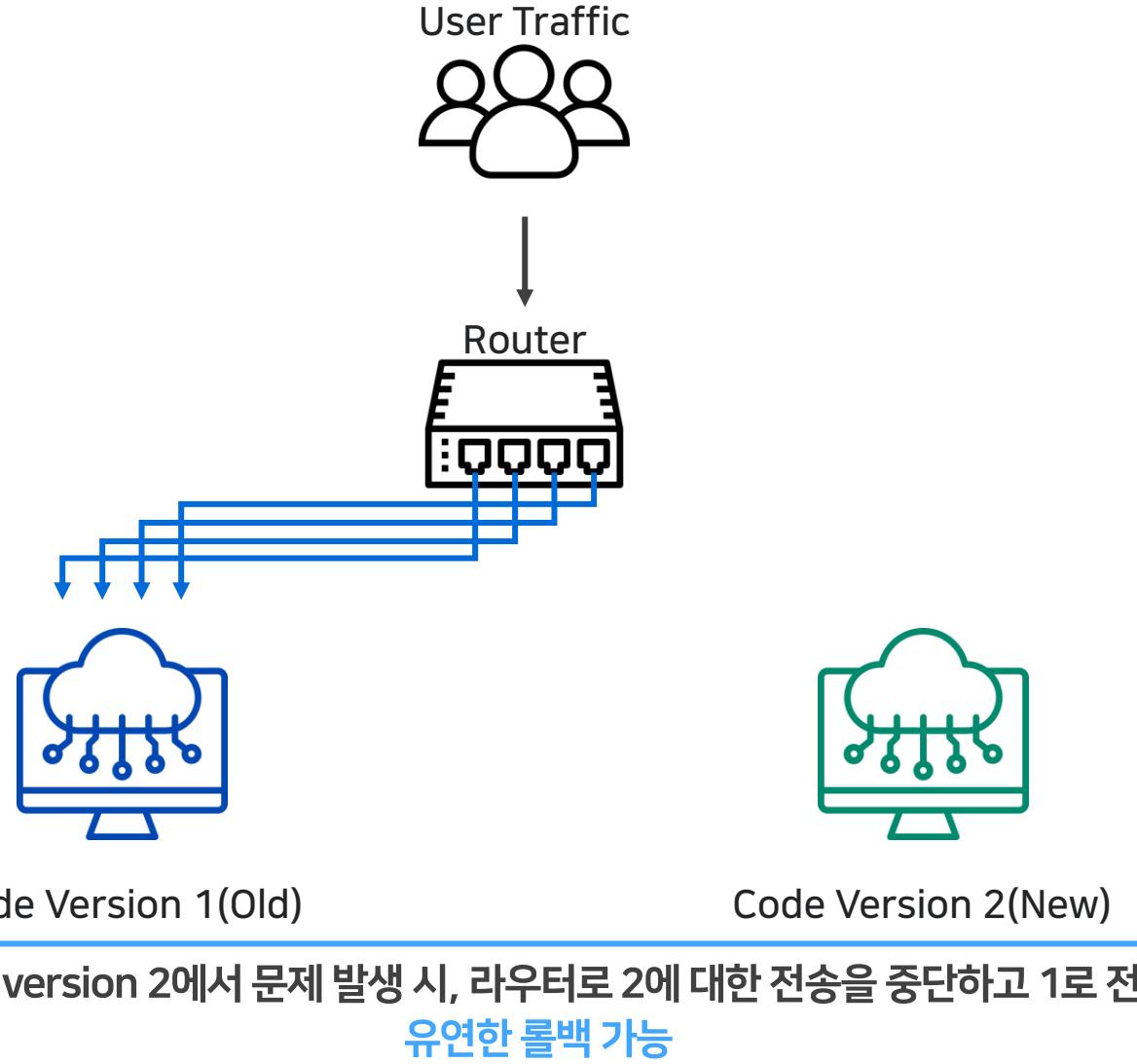
배포 스타일(2): Blue/Green 배포



Continuous delivery

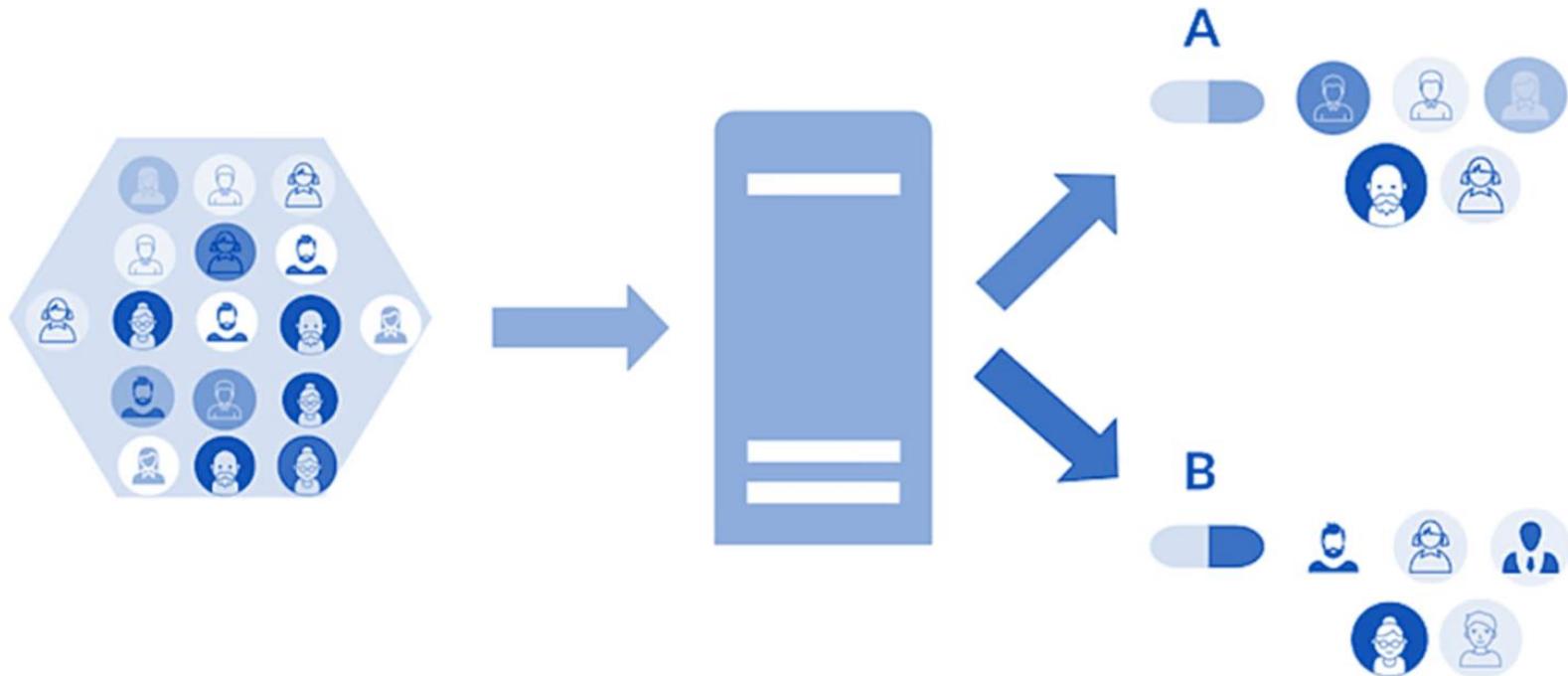
III

배포 스타일(2): Blue/Green 배포



Continuous delivery

배포 스타일(3): A/B Test



사용자를 그룹화하여 무작위로 선택한 모델에 대해 테스트 진행
각 모델에 대한 결과를 수집하여 최상의 결과 제공 모델 선택



감사합니다

Thank you

