머신러닝 파이프라인의 이해

이광희 PhD. / Co-founder / CAIS

About Me

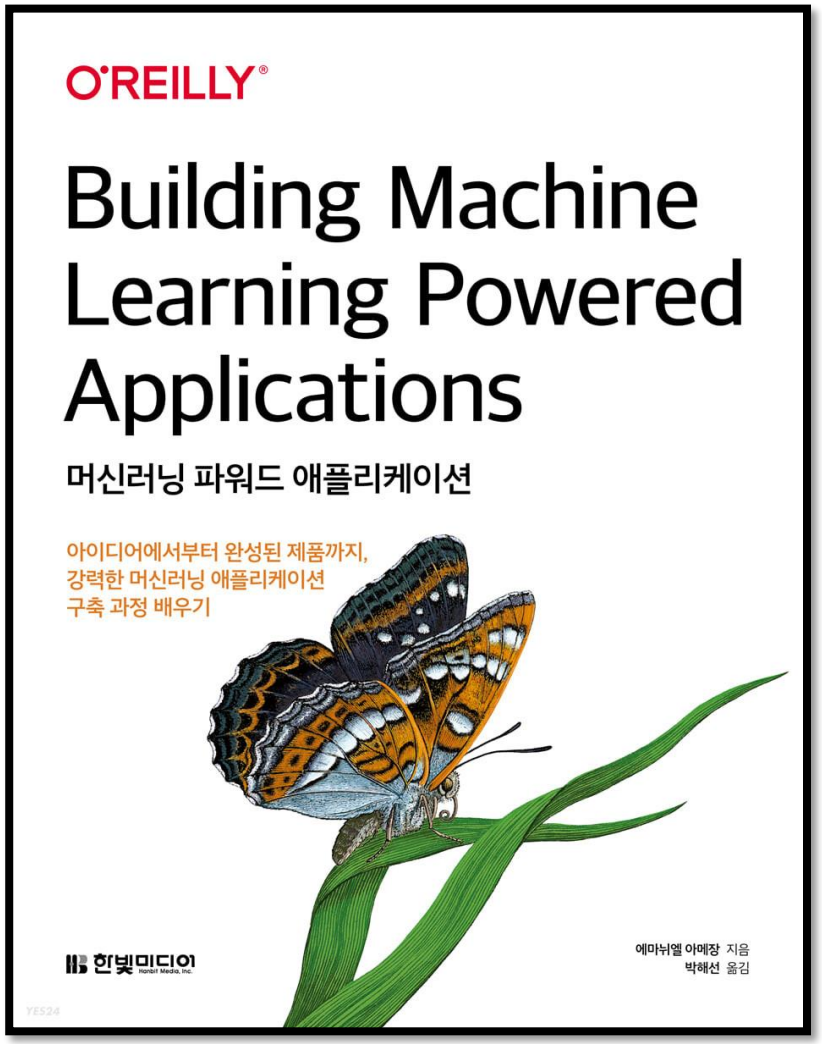
이 광 희 (Kwang Hee Lee), Ph D

Annotation-AI / Co-founder **/** Chief AI Scientist
VIVE STUDIOS / CTO
Trustworthy AI KR (페이스북 그룹) / 운영자
lkwanghee@gmail.com
+82-10-9990-9474

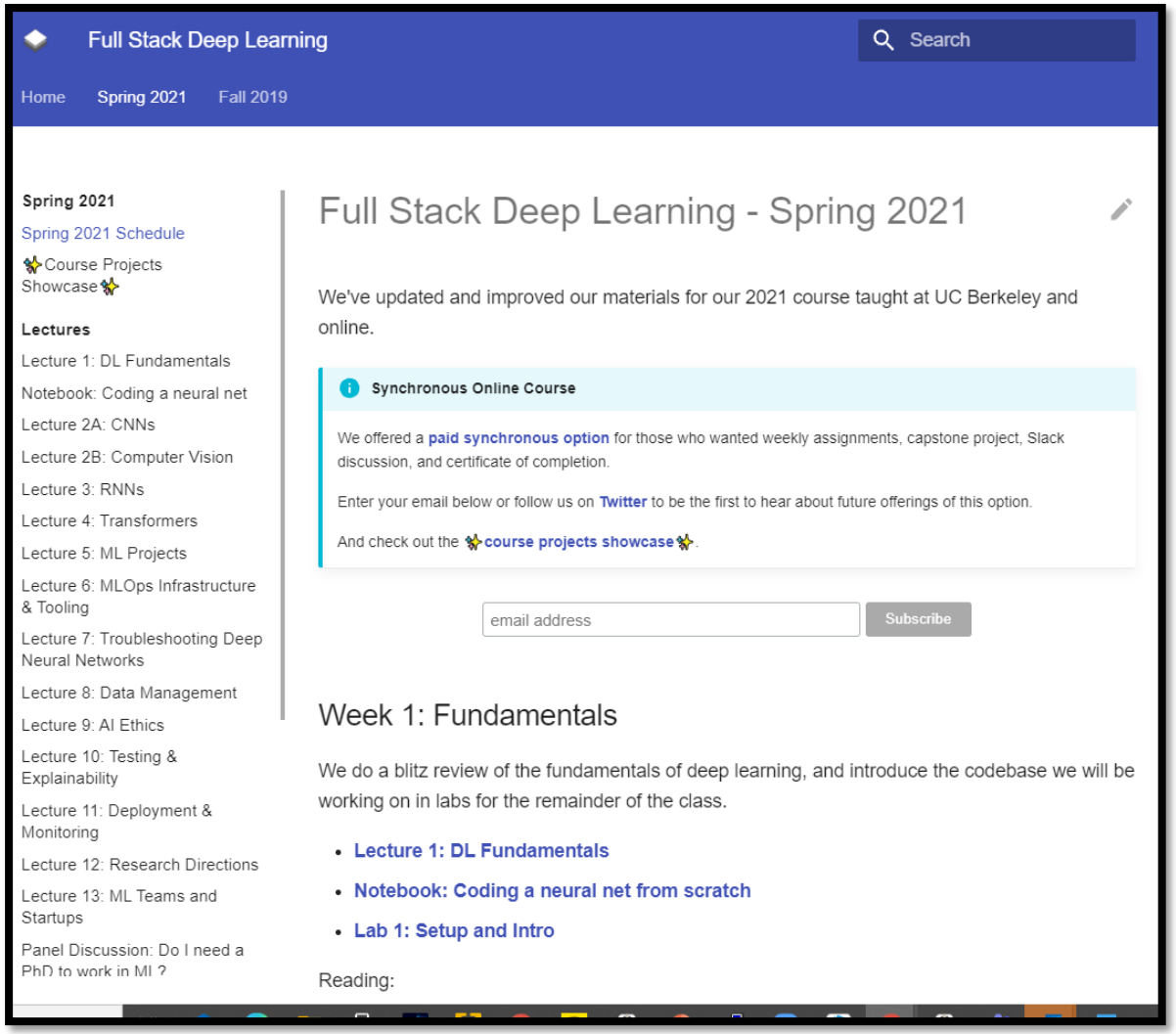
◆ Work Experience

- 前 **보잉 한국 기술연구소** / AI Tech Lead
- 前 **AIRI(인공지능연구원) /** Senior Researcher
- 前 **TOVIS** / Senior Researcher
- 前 **Samsung Medison /** Senior Research Engineer

참고도서 / 참고사이트



https://fullstackdeeplearning.com/spring2021/



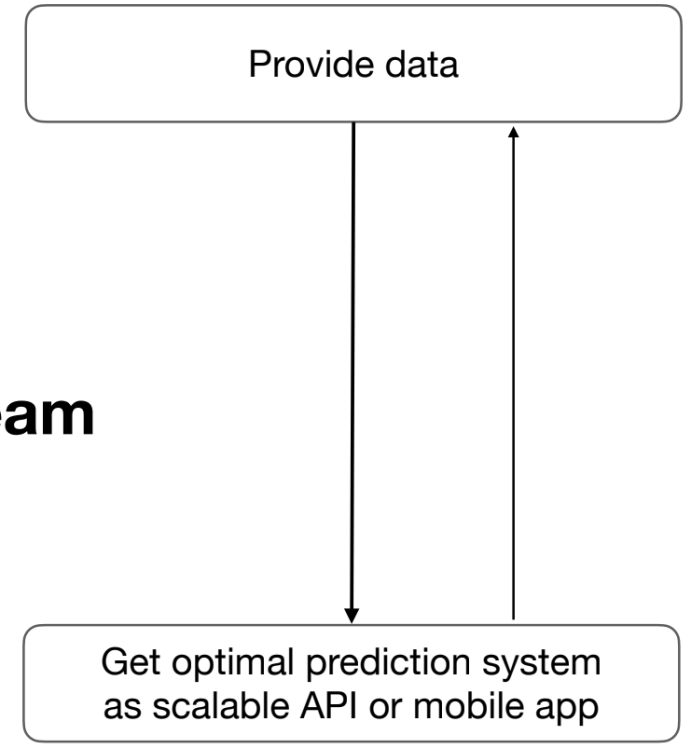
Machine Learning Pipeline의 개요

- **머신러닝 파이프라인의 개념**
  - 데이터 수집부터 전처리, 학습 모델 배포, 예측까지 전과정을 순차적으로 처리하도록 설계된 머신러닝 아키텍처

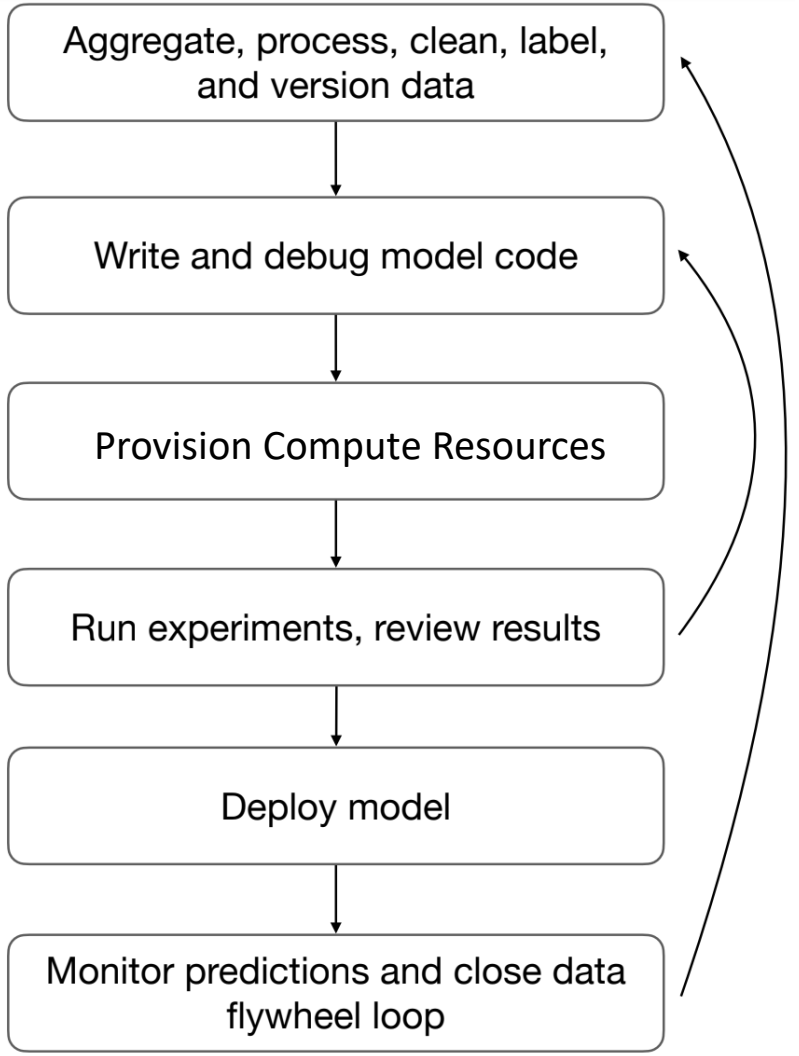
- **머신러닝 파이프라인의 필요성**
  - **머신러닝 자동화:** 불필요한 시간 소모를 줄이기 위해 머신러닝 모델 개발 전 과정의 지속적인 수행을 위한 파이프라인 기반 자동화
  - **예측 정확도 향상:** 내부 구조 이해를 통한 머신러닝 성능(예측의 정확성) 향상

Dream & Reality for ML Practitioners

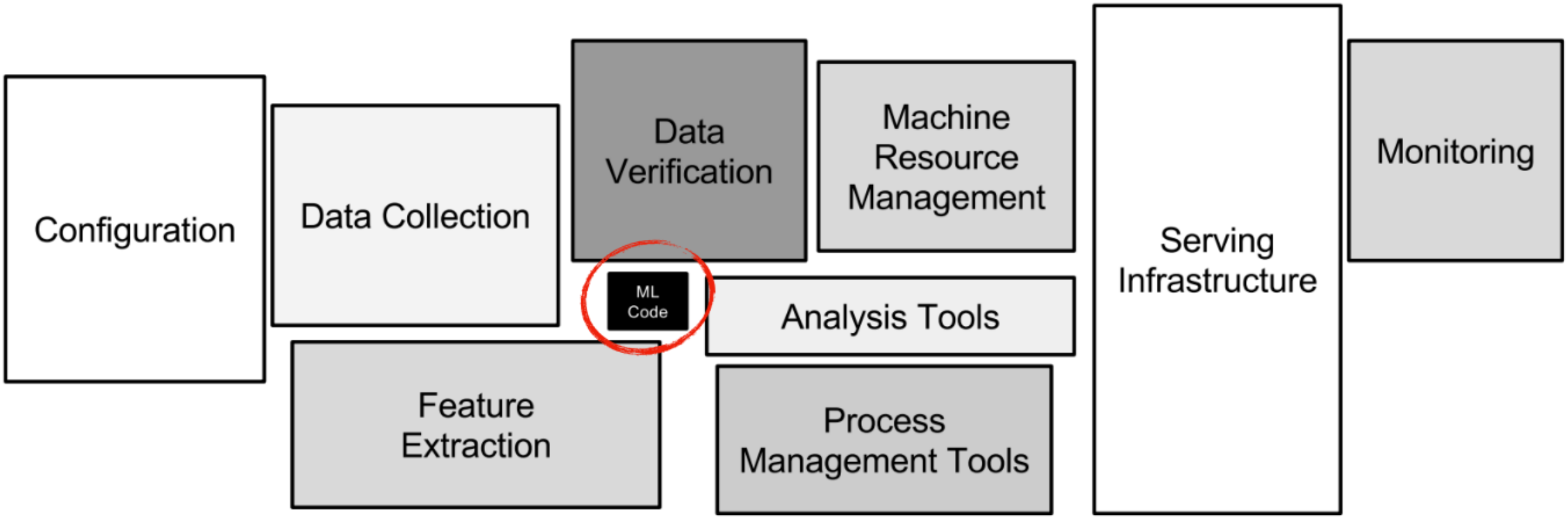
Dream



Reality



ML프로젝트에서 시스템의 요소



production level의 ML project에서 순수 ML 모델 개발은 5%도 채 안됨

https://chacha95.github.io/2020-10-11-Docker_Kubernetes9/

왜 AI/ML 프로젝트가 실패할까? (1/2)

1. 역량 부족에 의한 잘못된 개발 방향 설정

- 경험이 부족한 AI 인력에 의해 잘못된 방향으로 프로젝트 계획이 세워질 수 있음.
- 머신러닝을 적용하기에 적절한 분야에 적용 필요.
- 전통적인 엔지니어링이 더 좋은 솔루션을 만들 수 있음.

2. 데이터 전략의 부실

- AI 프로젝트를 진행하는 과정에서 가장 큰 문제로 부상하는 것이 부실한 데이터 전략임.
- 프로젝트를 본격적으로 운영하기 전에 신뢰할 수 있는 데이터 전략 수립이 중요함.
- 보유한 데이터를 파악하고, 필요한 데이터 양을 예측하며, 데이터를 선택, 수정하는 방법도 계획에 포함시켜야 함.
- 데이터 전략이 불충분하거나 데이터가 부족하면 결국 AI 프로젝트의 실패를 가져올 수 있음.

https://www.hankyung.com/it/article/202107060891i
https://riiidtechblog.medium.com/ai-%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8%EB%A5%BC-%EC%8B%A4%ED%8C%A8%ED%95%98%EA%B2%8C-%EB%A7%8C%EB%93%9C%EB%8A%94-5%EA%B0%80%EC%A7%80-%EC%9D%B4%EC%9C%A0-8e7813cf0a95

왜 AI/ML 프로젝트가 실패할까? (2/2)

3. 뛰어난 인공지능 엔지니어링 팀의 부재

- 투자를 통해 뛰어난 인공지능 엔지니어링 팀을 채용하는 것이 중요함.
- 아무리 매력적인 프로젝트라도, 뛰어난 인재가 없이는 프로젝트를 성공시키기 어려움.
- 사업적 성과를 위해서는 데이터 과학자, 머신러닝 연구자, 머신러닝 엔지니어, 소프트웨어 엔지니어들이 적절히 모인 팀을 구성해야 함.

4. 문제정의와 성공의 평가 기준의 불명확함

- AI를 이용하여 새로운 비즈니스나 시스템을 만들어 내는 것에 대해서 명확한 문제정의가 어려움.
- 프로젝트의 실행목적과 기대효과에 대한 이해 부족으로 인해 성공기준이 불명확해 짐.
- 모든 이해관계자와 함께 목표를 정의하고 구체적이고 측정 가능한 평가 지표를 설정 해야함.

5. 비즈니스에 대한 이해 부족

- 머신러닝 개발자가 비즈니스 문제를 명확히 이해해야 오류를 줄일 수 있음.

머신러닝 모델 개발에서 마주하는 난관

- **정제되지 않은 파이프라인으로 인한 부진한 진척도**
  - 머신러닝 모델 개발 작업은 반복적이고 비선형적인 프로세스를 따름.
  - 데이터를 기반으로 하고 있기 때문에 모니터링 결과를 바탕으로 주기적인 업데이트 필요
  - 많은 기업들이 이 단계를 원활히 수행할 수 있는 파이프라인을 마련하지 못해 모델 개발단계에서 배포 단계까지 진척이 늦어 짐.

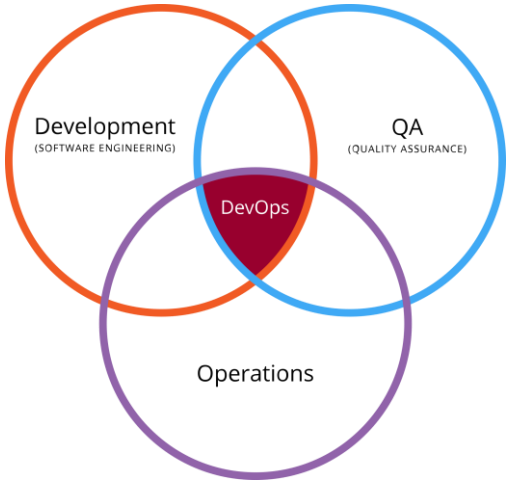
- **불필요한 리소스 활용 증가**
  - 데이터의 가변성으로 인해 특정 모델 배포 후 새로운 데이터 기반의 지속적인 모니터링, 학습, 업데이트 필요.
  - 체계화된 파이프라인이 없는 상태에서 운영팀으로부터 머신러닝 팀에 새로운 데이터 전달하는데 많은 리소스 및 시간이 소요됨.

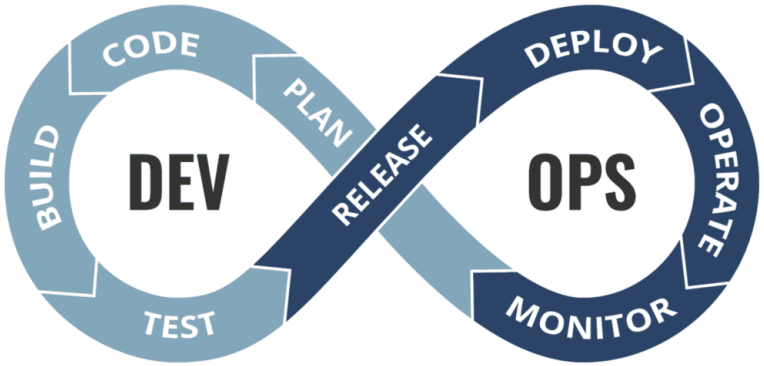
- **데이터 사이언티스트와 소프트웨어 개발자 간의 데이터 사일로 (Data Silo)**
  - 머신러닝 리서처 및 엔지니어와 소프트웨어 개발자간의 업무 공유와 협업이 어려워지는 문제가 발생.

*데이터 사일로 (Data Silo)
조직 내에 존재하는 일련의 정보가 조직 내 다른 부서와 공유되지 않아 특정 부서만의 데이터로 남게 되는 현상

DevOps란?

- 소프트웨어 개발(**Dev**elopment)과 운영(**Op**eration**s**)의 합성어
- 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화
- **목표:** 소프트웨어 개발조직과 운영조직간의 상호 의존적 대응을 통해 조직이 소프트웨어 제품과 서비스를 빠른 시간에 개발 및 배포하는 것
- **특징:** 반복적으로 코드를 통합하는 지속적 통합 (Continuous Integration, CI)과 이를 반복적으로 배포하는 지속적 배포 (Continuous Delivery, CD)

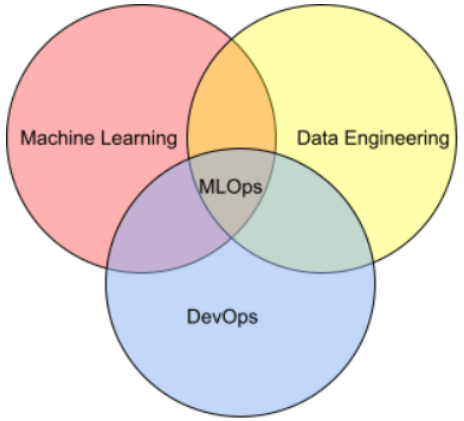


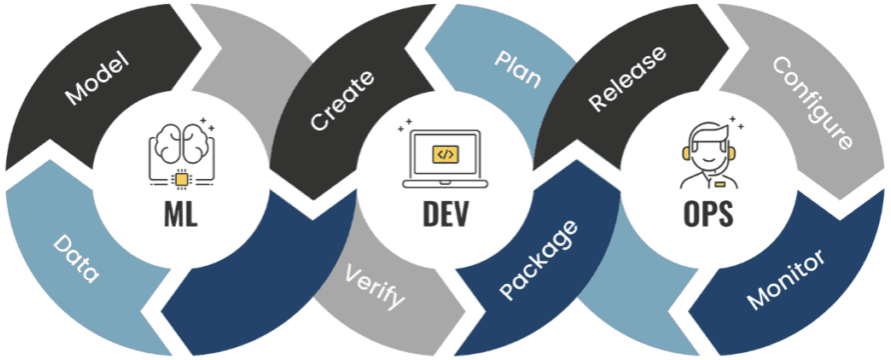


https://ko.wikipedia.org/wiki/%EB%8D%B0%EB%B8%8C%EC%98%B5%EC%8A%A4
https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference/

MLOps란?

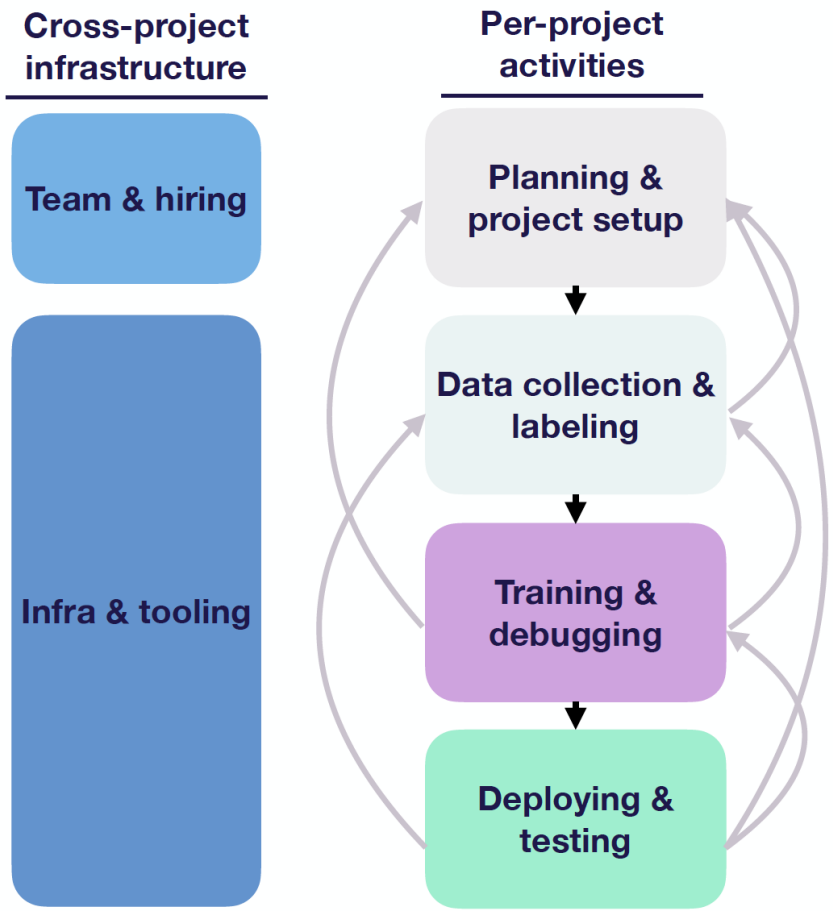
- 머신러닝(**M**achine **L**earning)과 운영(**Op**eration**s**)의 합성어
- 모델 개발 시스템과 운영 시스템을 통합하여 더욱 유연하고 효율적인 개발을 가능케 하는 도구이자 개발 환경
- **목표:** ML Research 팀이 개발한 최상의 모델을 배포 환경에서 동일하게 구현하여 유의미한 비즈니스 창출
- **특징:** 지속적 통합 (Continuous Integration, CI), 지속적 배포 (Continuous Delivery, CD) 단계를 일부 동일하게 이행, 추가적으로 데이터가 바뀔 때마다 모델을 반복 학습시키는 지속적 학습(Continuous Training, CT)





https://neptune.ai/blog/mlops-organizational-and-communication-problem-not-a-tech-problem
https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference/

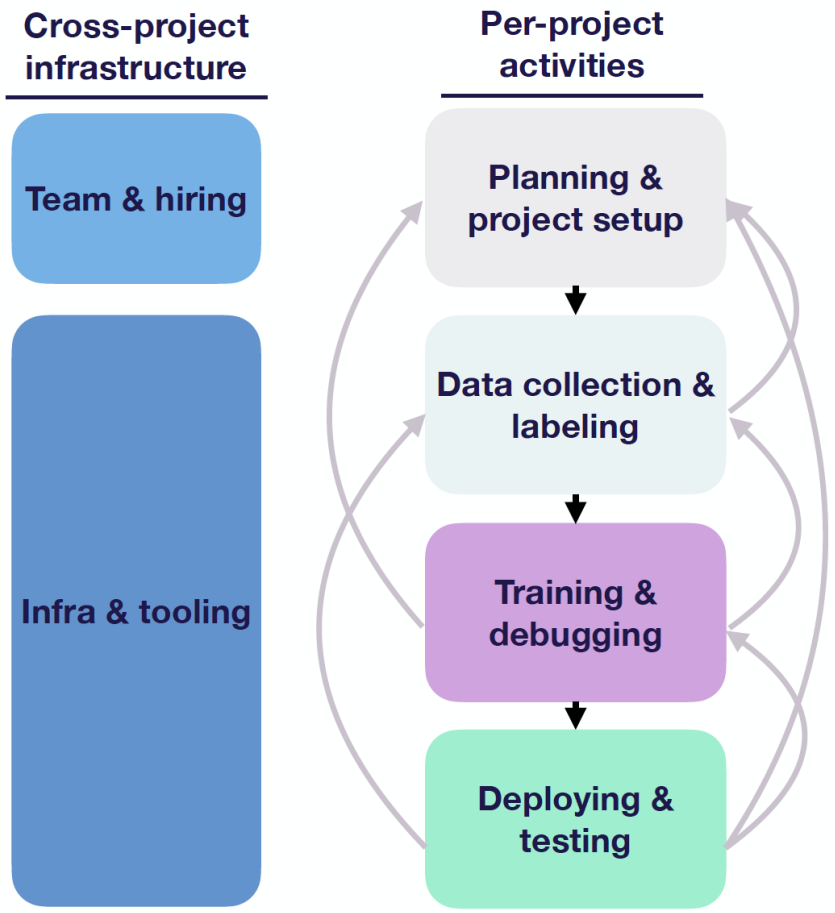
ML 프로젝트의 생애주기



1. **Planning & Project Setup**
   - 해결할 문제를 결정하고 목표와 요구사항을 판단한다.
   - 어떻게 적절한 리소스를 모을지 판단한다.
2. **Data Collection & Labeling**
   - 학습 데이터를 모으고 데이터를 모델 요구사항에 맞게 annotate한다. 데이터 취득이 어렵거나 테스크 수정이 필요 (1단계로)

https://fullstackdeeplearning.com/spring2021/lecture-5/

ML 프로젝트의 생애주기

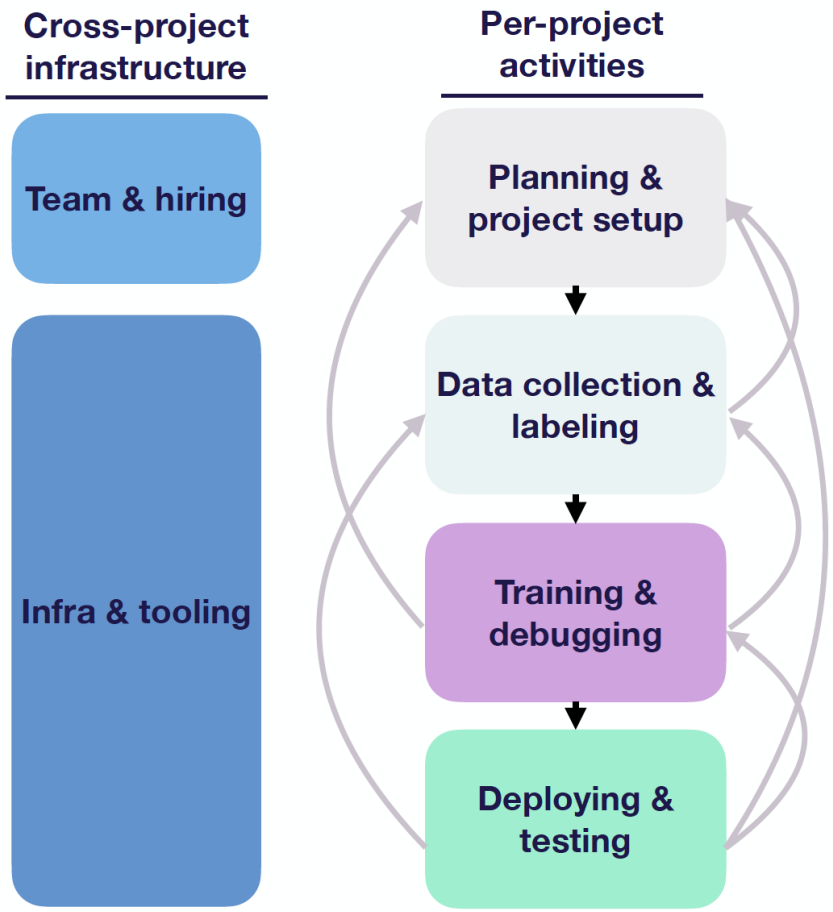


3. Training & Debugging

- 빠르게 Baseline Model을 만들고, SOTA 방법들을 찾고 Problem Domain에 맞춰 reproduce한다.
- 만든 것들을 debug하면서 특정 task 성능을 향상시킨다.
- 데이터 추가 또는 라벨링 품질 향상 필요 (2단계로)
- 작업의 난이도가 어려움을 인식, 또는 요구사항 조정 필요(1단계로)

https://fullstackdeeplearning.com/spring2021/lecture-5/

ML 프로젝트의 생애주기

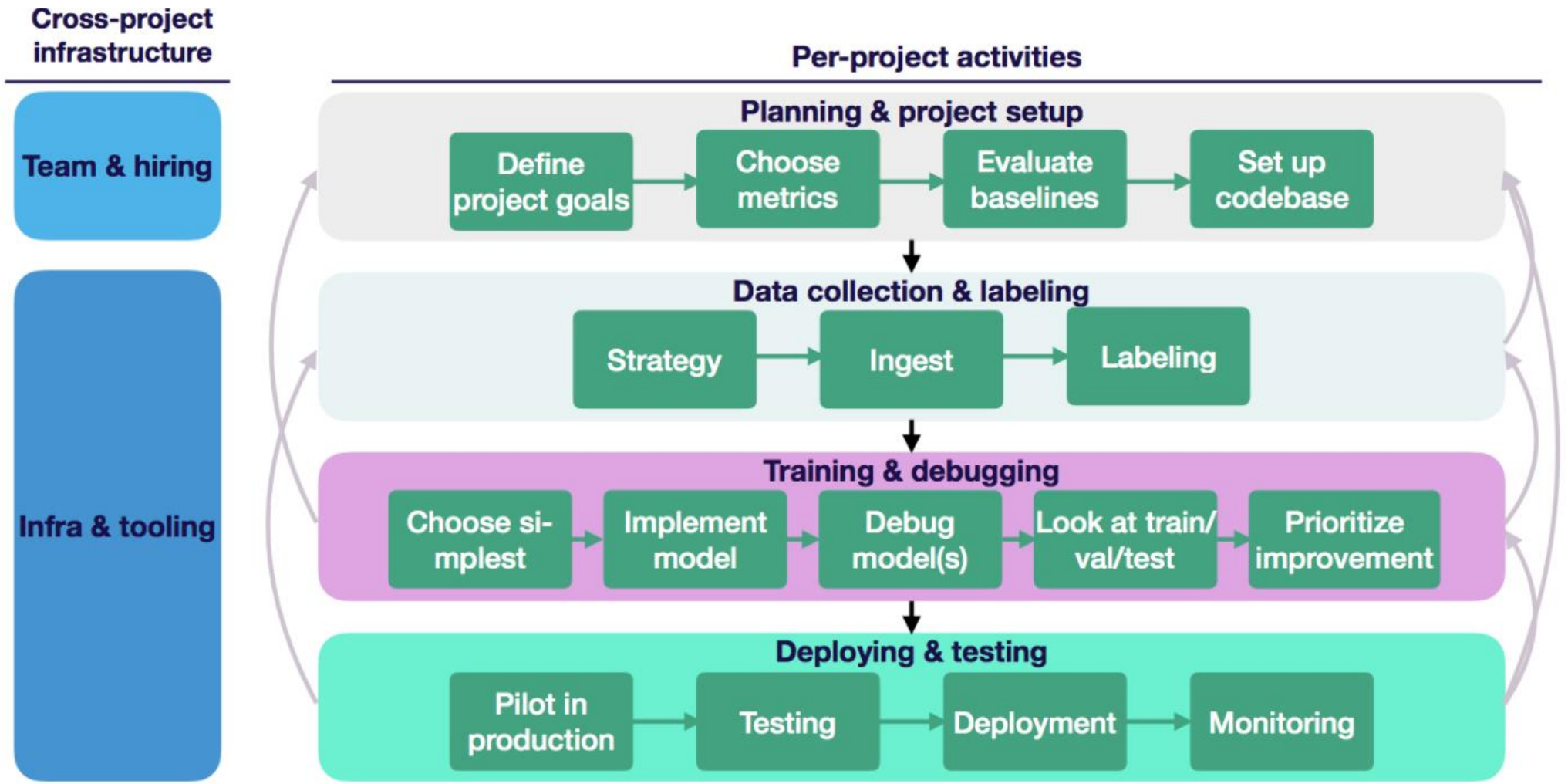


4. Deploying & Testing

- 제한된 환경(Lab)에서 모델을 pilot하고, 테스트를 수행하고, 모델을 프로덕션으로 배포한다.
- Lab환경에서 모델이 잘 동작하지 않음. (3단계로)
- Training data의 분포와 production 환경에서 취득한 data 분포의 차이를 인식하고, 추가로 데이터나 hard case 취득이 필요한 경우(2단계로)
- 모델 평가 metric이 user behavior와 맞지 않아서 재설정 필요하거나 실제 성능이 만족스럽지 않아 요구사항을 재설정 해야 할 경우 (1단계로)

https://fullstackdeeplearning.com/spring2021/lecture-5/

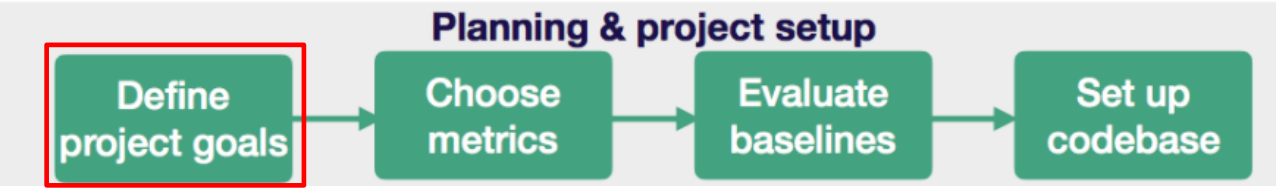
ML 프로젝트의 생애주기

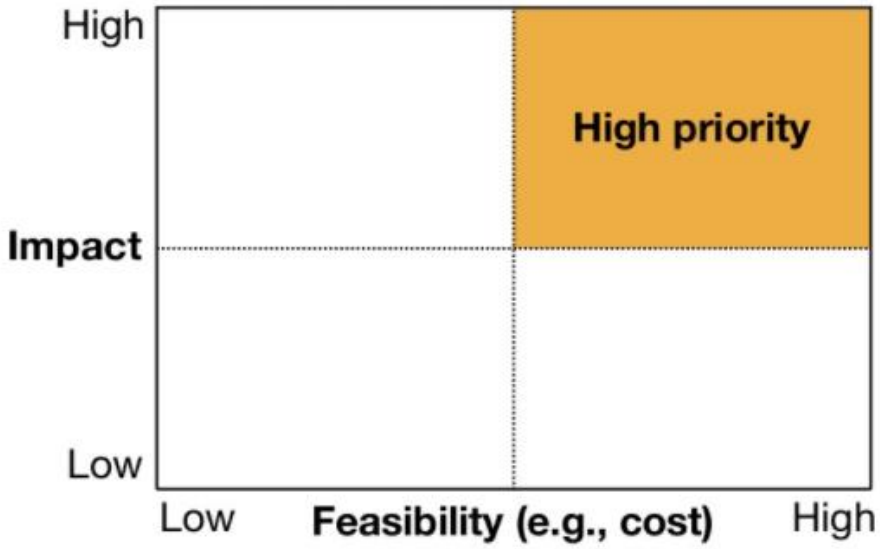


https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup

Planning & Project Setup-우선순위 정하기(Prioritizing)

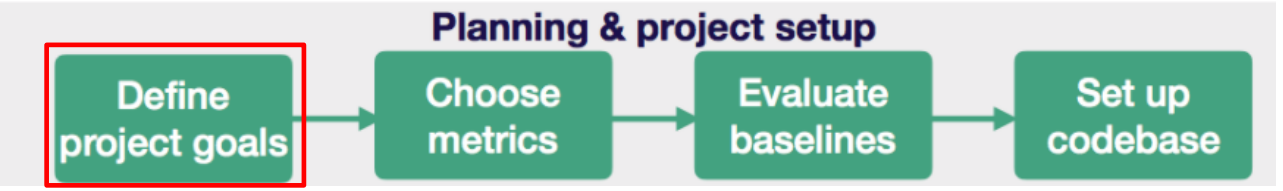




- **프로젝트 우선순위를 정하기 위한 키 포인트**
  - **High Impact:** 비즈니스 프로세스의 복잡한 부분에서 cheap prediction으로 큰 효과를 내는 가치 있는 프로젝트
  - **High Feasibility:** 데이터 가용성 (Data Availability), 정확도 요구사항(Accuracy Requirement), 문제 난이도(Problem Difficulty)를 바탕으로 높은 실행 가능성을 가진 프로젝트

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-우선순위 정하기(Prioritizing)

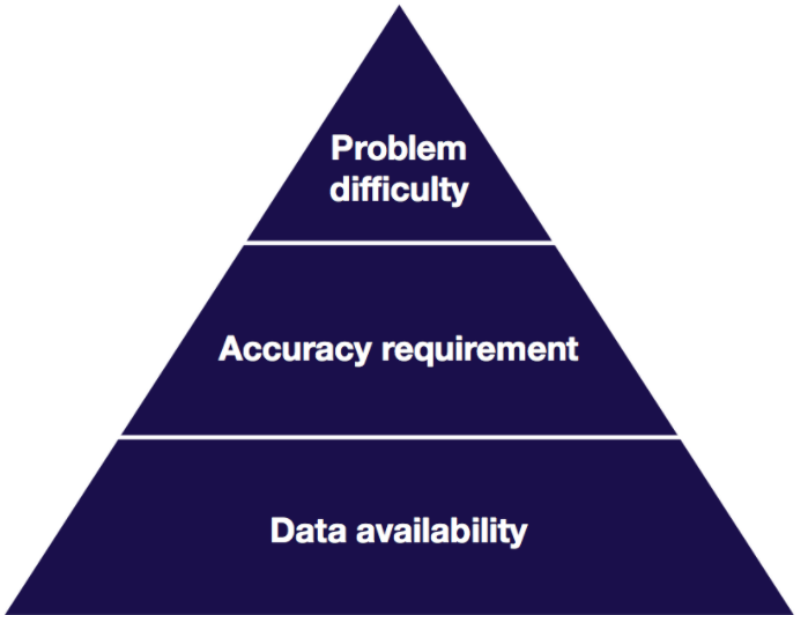


Assessing feasibility of ML projects

문제 난이도

정확도 요구사항

데이터 가용성



Main considerations

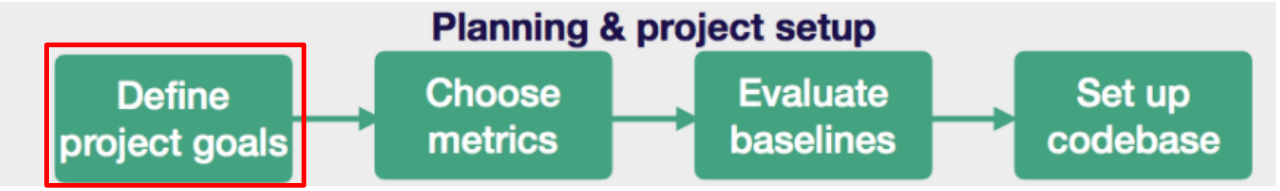
- Good published work on similar problems? (newer problems mean more risk & more technical effort)
- Compute needed for training?
- Compute available for deployment?

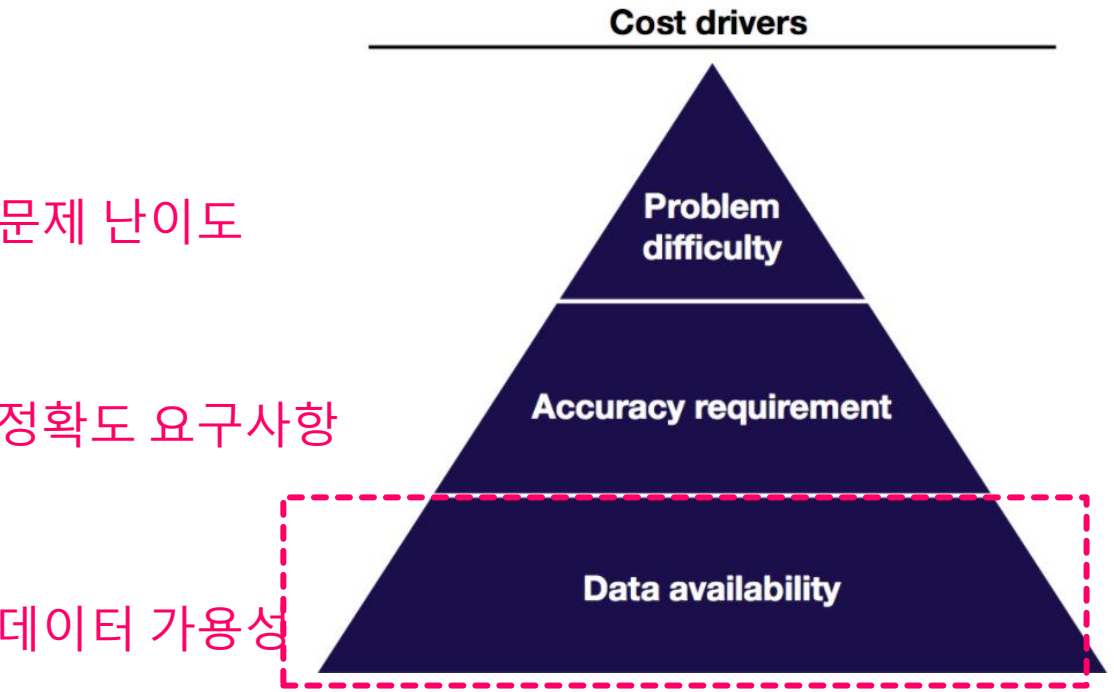
- How costly are wrong predictions?
- How frequently does the system need to be right to be useful?

- How hard is it to acquire data?
- How expensive is data labeling?
- How much data will be needed?

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-우선순위 정하기(Prioritizing)



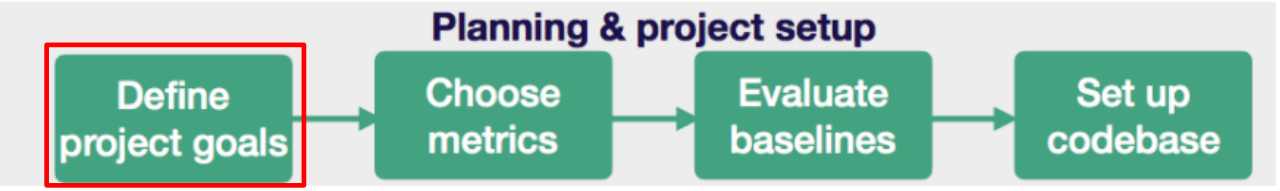


데이터 가용성 (Data Availability)

- 레이블을 가진 데이터가 있는 경우
  - 모델 훈련과 품질 평가가 수월함.
  - 요구사항에 맞고, 무료로 레이블된 데이터셋을 구하는 것은 어려움.
- 유사 레이블을 가진 데이터가 있는 경우
  - 요구사항에 맞는 레이블은 아니지만, 관련은 있는 레이블
  - 완벽한 레이블된 데이터보다 찾기 수월.
- 레이블이 없는 데이터가 있는 경우
  - 데이터는 있으나 레이블이 없는 경우
  - 레이블링을 하거나, 레이블 되지 않은 데이터로 학습하는 모델을 찾아야 함.
- 데이터를 새로 수집해야 하는 경우

https://fullstackdeeplearning.com/spring2021/lecture-5/

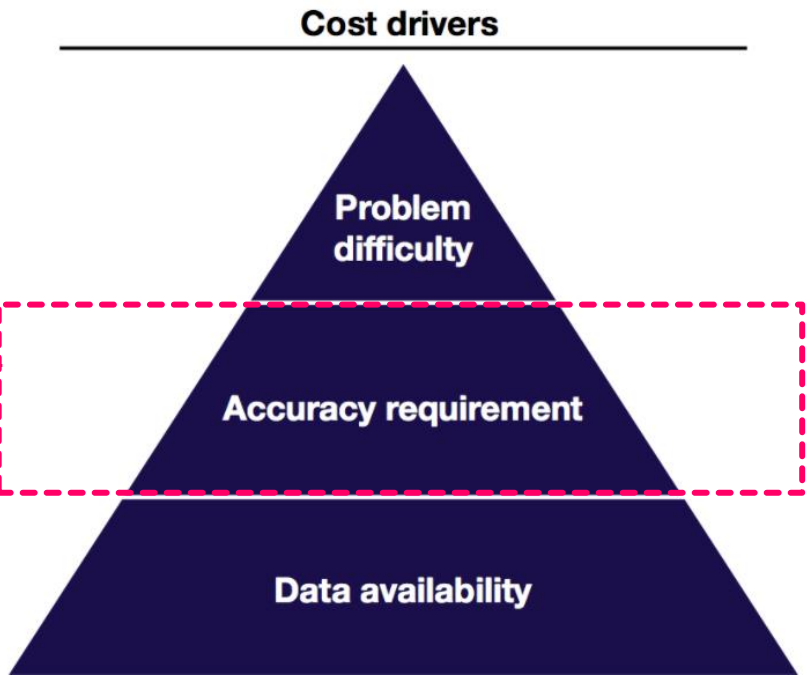
Planning & Project Setup-우선순위 정하기(Prioritizing)

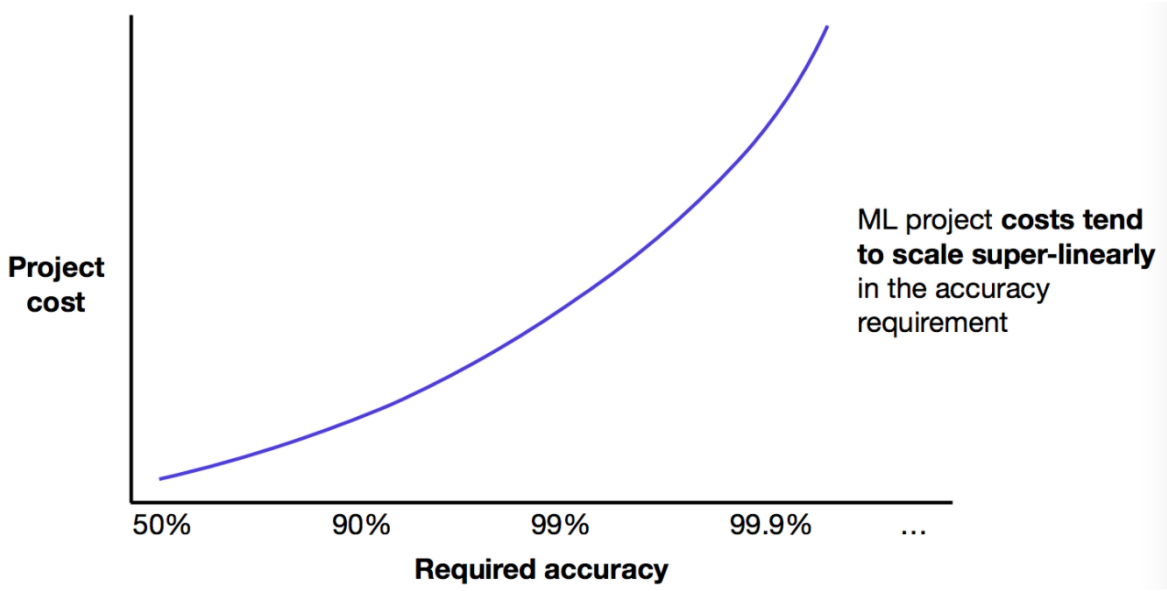


문제 난이도

정확도 요구사항

데이터 가용성

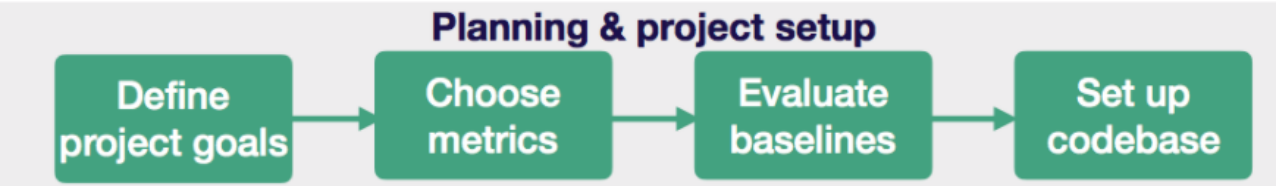




- ML프로젝트의 Cost는 데이터 가용성에 의해 결정되지만, 정확성 요구도 큰 역할을 함.
- 높은 성능까지 가려면 cost가 지수함수처럼 증가함

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-모델 & 데이터의 형태

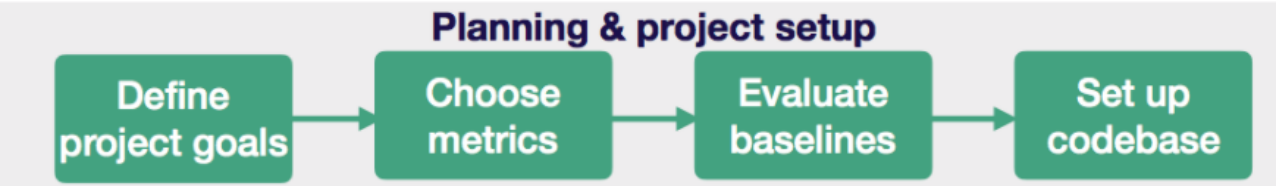


- 여러 가지 방식을 고려하고 가능성을 평가하기 위해 머신러닝 문제의 두 가지 핵심 요소인 모델과 데이터를 파악할 필요 있음.

- **모델의 형태**
  - 지도 학습 (Supervised Learning) / 비지도 학습 (Unsupervised Learning) / 반지도 학습 (Semi-supervised Learning) / 약지도 학습 (Weakly Supervised Learning) / 강화학습(Reinforcement Learning)
  - 분별모델 (Discriminative Model)/ 생성모델 (Generative Model)
  - 컴퓨터 비젼 / 자연어 처리 / 음성인식 / 추천시스템 / 시계열 예측 (Time Series Forecasting) / 이상치 탐지 (Anomaly Detection)

- **데이터의 형태**
  - **정형 데이터** – 테이블 형태의 데이터 (데이터 베이스, 스프레드 시트 등)
  - **반정형 데이터** – 시스템 로그, 센서 데이터, HTML
  - **비정형 데이터**- 이미지, 동영상, 음성 데이터, 이메일, 문서 등.

Planning & Project Setup-초기모델 설계



- **초기 모델 설계의 3가지 방법**
  1. **기준 모델(Baseline):** 도메인 지식을 활용해 머신러닝을 사용하지 않고 간단한 규칙을 설계해 본다.
  2. **단순한 모델 (Simple Model):** 문제를 여러 단계로 나누어 단계별로 머신러닝 모델 및 규칙기반 모델을 적용해 본다.
  3. **복잡한 모델 (Complex Model):** 문제를 하나의 end-to-end 방식으로 모델링 한다. 대응하는 데이터와 자원이 필수.

- **초기 모델 설계 Tip**
  - 기준 모델을 머신러닝 없이 설계해서 적용해보고, 이를 통해 머신러닝의 필요성을 평가한다.
  - 위의 평가 시 머신러닝이 필요하면 적절한 모델링 방법을 찾아본다.
  - 대부분의 경우 머신러닝 없이 시작하는 것이 머신러닝 제품을 만드는 가장 빠른 방법이다.

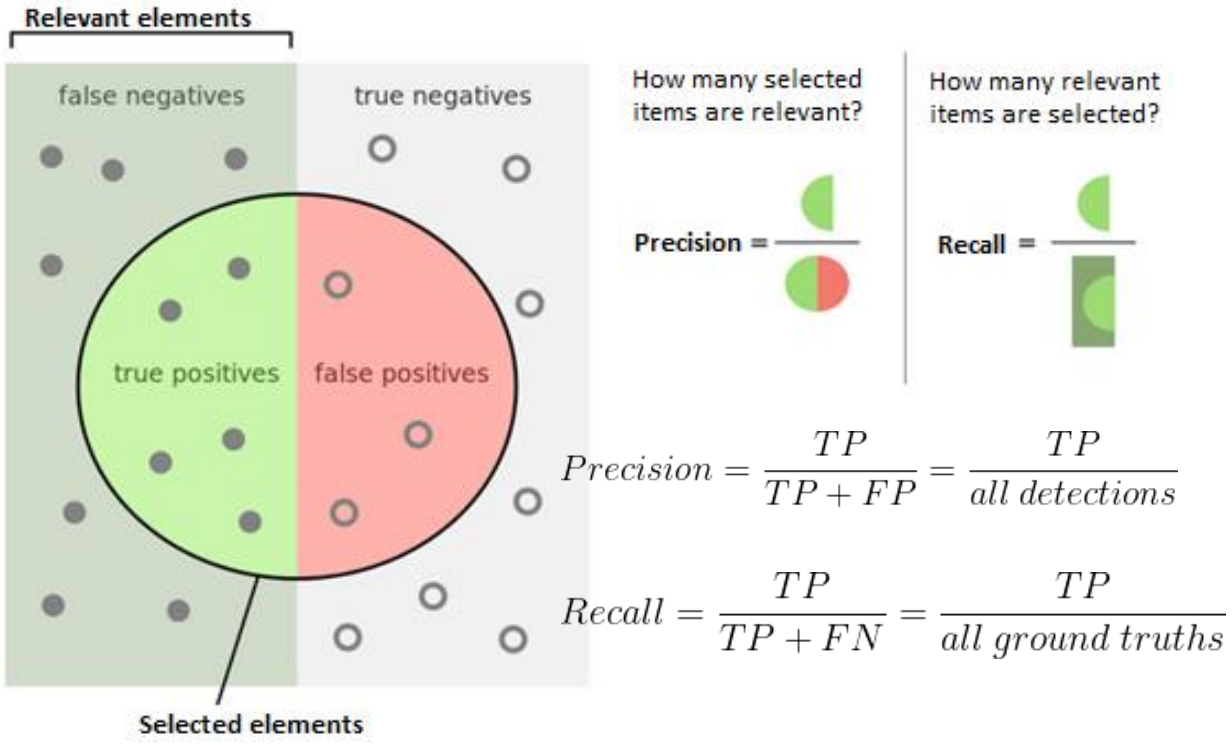
Planning & Project Setup-Choose Metrics

- **기계학습 프로젝트를 최적화 하기 위한 metric은 어떻게 선택하는가?**

  1. 현실세계는 복잡하기에 다양한 metric으로 바라볼 수 있다.
  2. 하지만 ML 시스템은 하나의 수치에 최적화 할 때 잘 동작한다.
  3. 결과적으로 다양한 metric를 조합해서 사용해야 한다.
  4. 이러한 조합은 바뀔 수 있다.

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-Choose Metrics



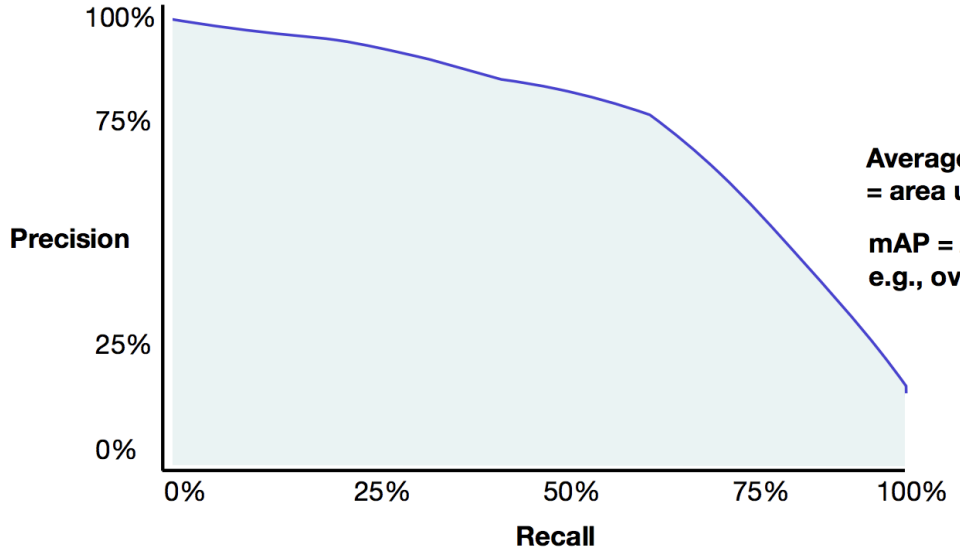
How many selected items are relevant?

Precision = $\frac{TP}{TP+FP} = \frac{TP}{all\ detections}$

How many relevant items are selected?

Recall = $\frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$

Domain-specific metrics: mAP



https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-Choose Metrics

왜 single metric을 선택할 필요가 있을까?

	Precision	Recall
Model 1	0.9	0.5
Model 2	0.8	0.7
Model 3	0.7	0.9

Which is best?

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-Choose Metrics

- Precision과 recall의 평균값?

	Precision	Recall	(p + r) / 2
Model 1	0.9	0.5	0.7
Model 2	0.8	0.7	0.75
Model 3	0.7	0.9	0.8

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup-Choose Metrics

- **Thresholding Metrics:** Recall이 0.6 이상인 것 중 precision이 높은 것?

	Precision	Recall	(p + r) / 2	p @ (r > 0.6)
Model 1	0.9	0.5	0.7	0.0
Model 2	0.8	0.7	0.75	0.8
Model 3	0.7	0.9	0.8	0.7

https://fullstackdeeplearning.com/spring2021/lecture-5/

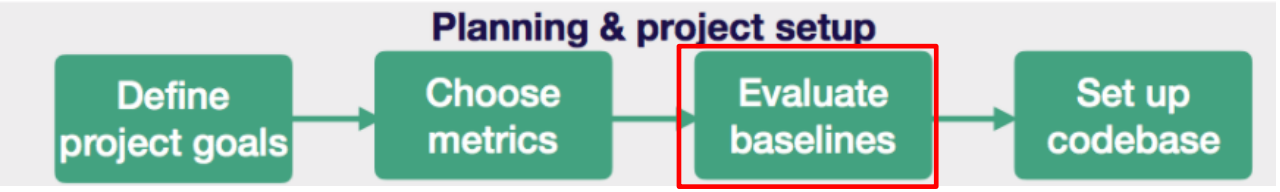
Planning & Project Setup-Choose Metrics

- Domain 특화된 공식(ex. mAP)를 활용?

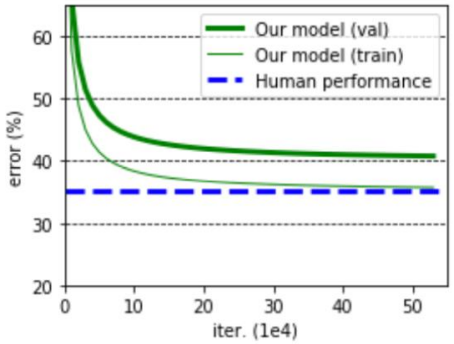
	Precision	Recall	(p + r) / 2	p @ (r > 0.6)	mAP
Model 1	0.9	0.5	0.7	0.0	0.7
Model 2	0.8	0.7	0.75	0.8	0.6
Model 3	0.7	0.9	0.8	0.7	0.6

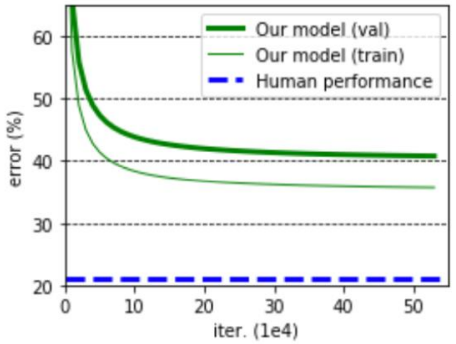
https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup- Baselines



Same model, different baseline —> different next steps



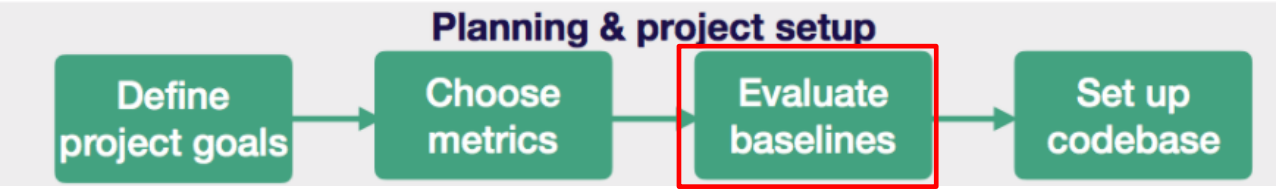


- **베이스라인**
  - 모델 성능의 최소 하한선을 제공 (최소한 이 이상은 해야함)
  - 하한선이 엄격할 수록 더 유용함

- **베이스라인이 중요한 이유**
  - 같은 모델이 다른 베이스라인을 갖으면 다음 행동이 달라짐.

https://fullstackdeeplearning.com/spring2021/lecture-5/

Planning & Project Setup- Baselines

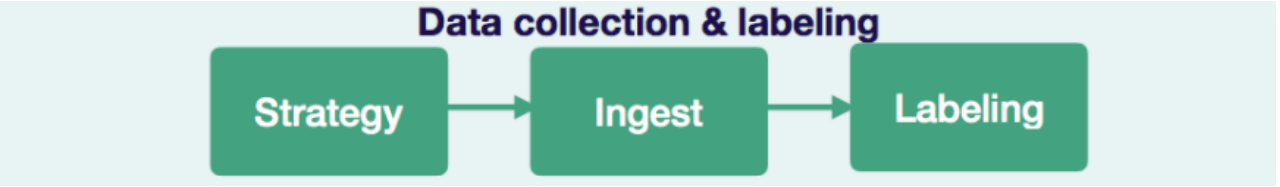


- **좋은 Baseline을 만드는 방법**
  - 일반인 보다 전문가들로 부터 모은 데이터가 baseline 질을 높여 주지만, 상대적으로 데이터 수집 의 난이도가 높아진다.

https://fullstackdeeplearning.com/spring2021/lecture-5/

Data Collection & Labeling

Data Collection & Labeling



- **데이터 수집과 레이블링 옵션**

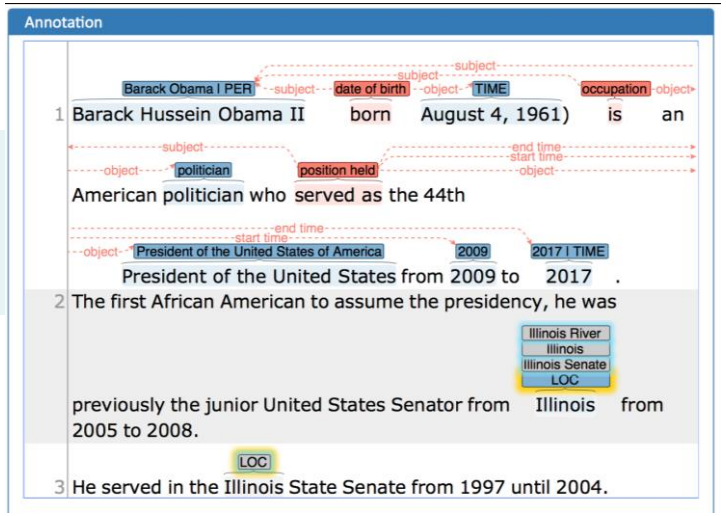
1. **자체 어노테이터 채용**
   - **빠르고 질 좋은** 어노테이션 가능
   - 비용이 많이 들고, 확장이 어려움.

2. **크라우드 소싱**
   - Amazon Mechanical Tuck과 유사 플랫폼 이용
   - 빠르고 싸지만, 어노테이션의 질을 보장하지 못함.

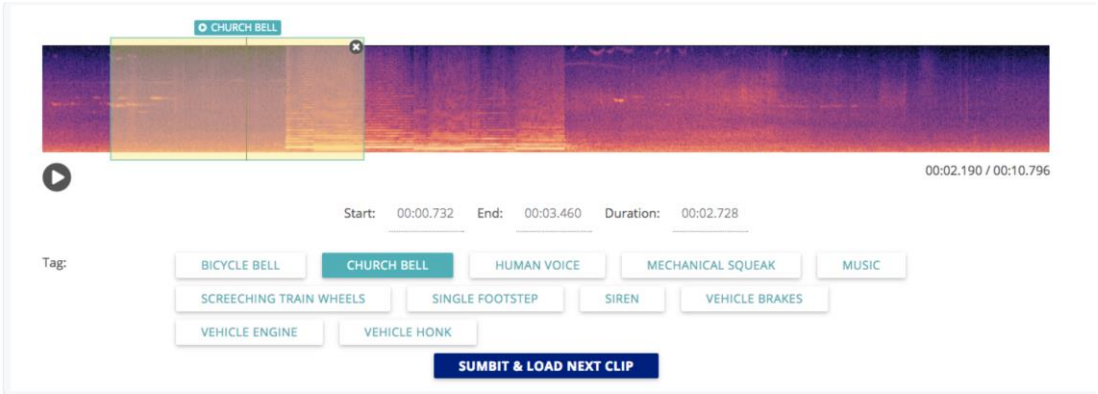
3. **데이터 구축 서비스 회사**
   - 비용과 시간을 고려할 때 매우 합리적 선택
   - 일부 검증 데이터에 직접 어노테이션을 수행 후 여러 업체들에 샘플을 요청하여 결정
   - **국내업체:** 크라우드웍스 (https://crowdworks.kr/), Annotation-AI (https://www.annotation-ai.com/) 테스트웍스 (https://www.testworks.co.kr/), AIMMO(https://aimmo.co.kr/)

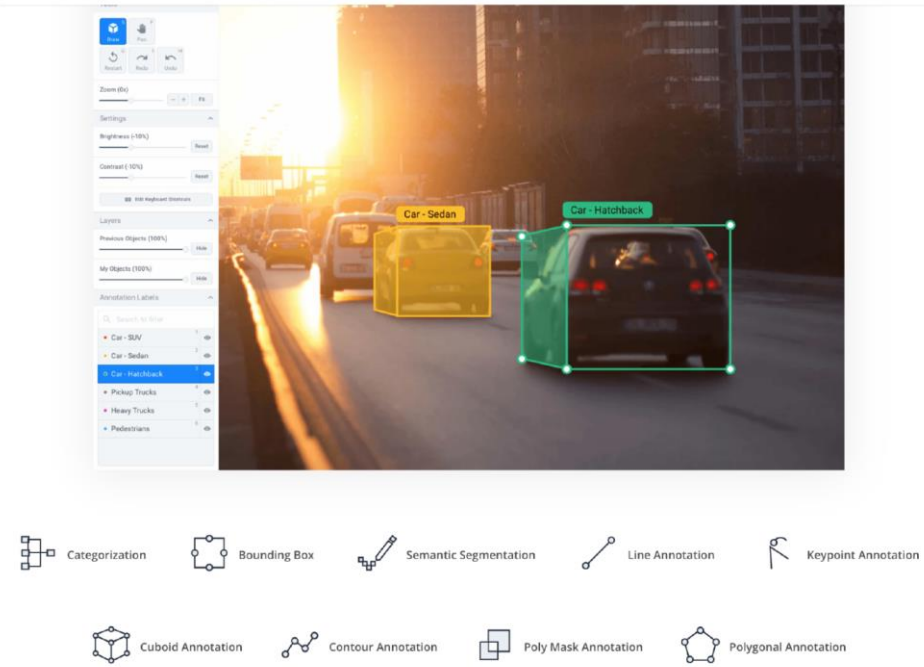
https://fullstackdeeplearning.com/spring2021/lecture-8/

Data Collection & Labeling



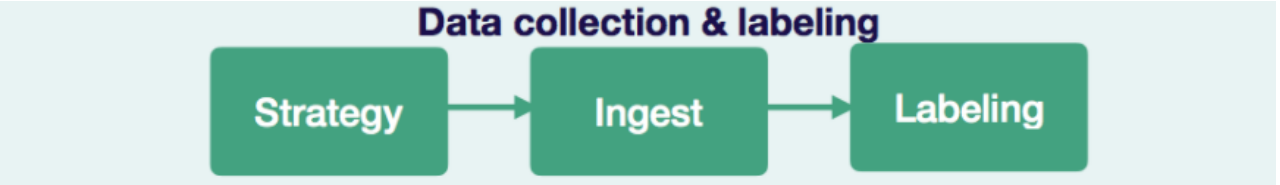
- **데이터 레이블링 소프트웨어**
  - https://github.com/jsbroks/awesome-dataset-tools
  - Images / Audio / Time Series / Text
  - 수동 / 반자동 / 모델기반 반자동 레이블링 기능
  - 관리자 / 검수자 / 어노테이터 소통 기능
  - **국내업체:** Annotation-AI (https://www.annotation-ai.com/) 테스트웍스 (https://www.testworks.co.kr/), AIMMO(https://aimmo.co.kr/)



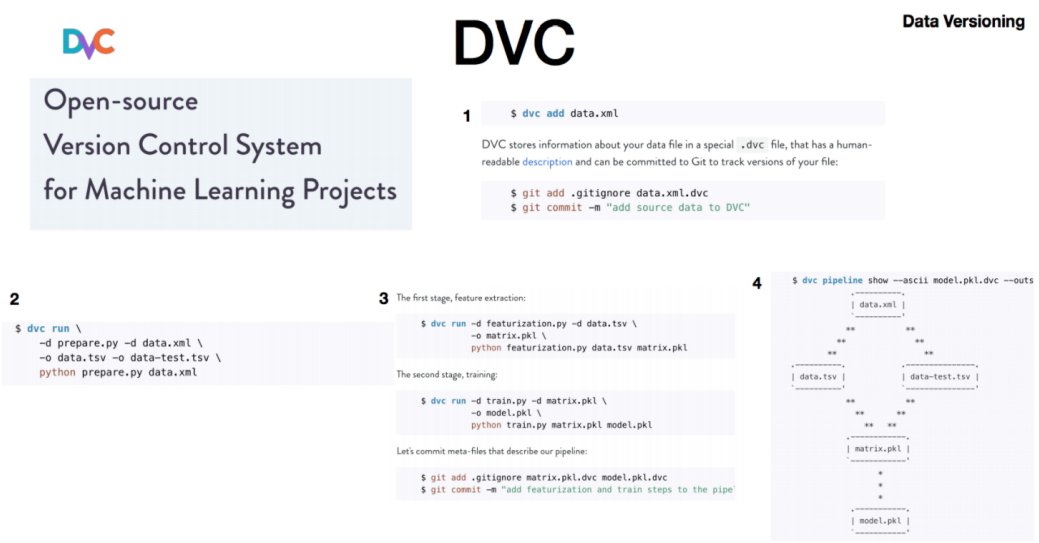


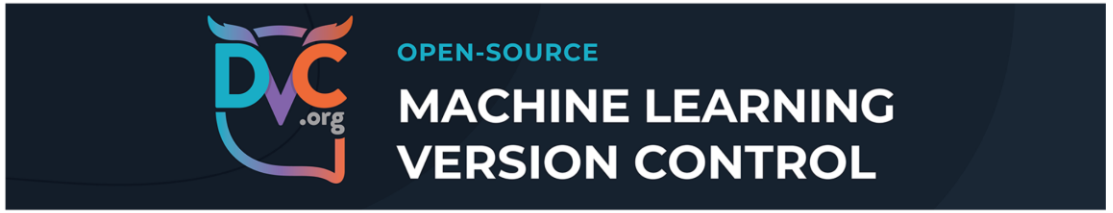
https://fullstackdeeplearning.com/spring2021/lecture-8/

Data Collection & Labeling



- **Data Versioning**
  - 머신러닝 모델은 소스코드와 데이터로 구성
  - 만약, data의 버전이 관리되지 않으면, 머신러닝 모델이 완벽하게 관리된다고 할 수 없음.





Website • Docs • Blog • Twitter • Chat (Community & Support) • Tutorial • Mailing List

Tests passing | maintainability B | codecov 91% | patreon donate | DOI 10.5281/zenodo.3677553

pip v2.7.3 | deb|pkg|rpm|exe v2.7.3 | brew v2.7.2 | conda invalid | choco v2.7.1 | snap install

Data Version Control or **DVC** is an **open-source** tool for data science and machine learning projects. Key features:

1. Simple **command line** Git-like experience. Does not require installing and maintaining any databases. Does not depend on any proprietary online services.
2. Management and versioning of **datasets** and **machine learning models**. Data can be saved in S3, Google cloud, Azure, Alibaba cloud, SSH server, HDFS, or even local HDD RAID.
3. Makes projects **reproducible** and **shareable**; helping to answer questions about how a model was built.
4. Helps manage experiments with Git tags/branches and **metrics** tracking.

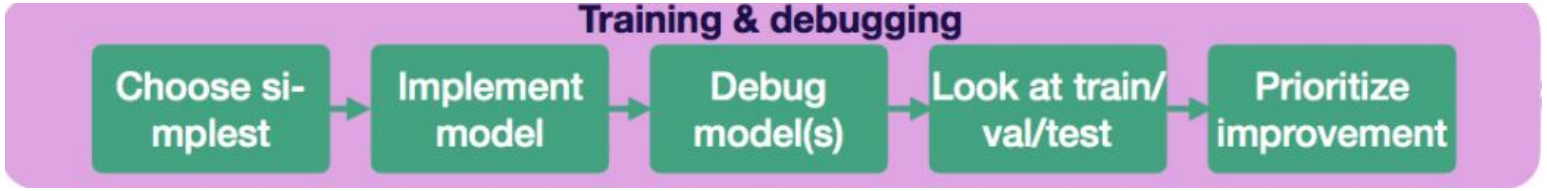
DVC aims to replace spreadsheet and document sharing tools (such as Excel or Google Docs) frequently used as both knowledge repositories and team ledgers. DVC also replaces both ad-hoc scripts to track, move, and deploy different model versions and ad-hoc data file suffixes and prefixes.

Contents

https://github.com/iterative/dvc

Training & Debuging

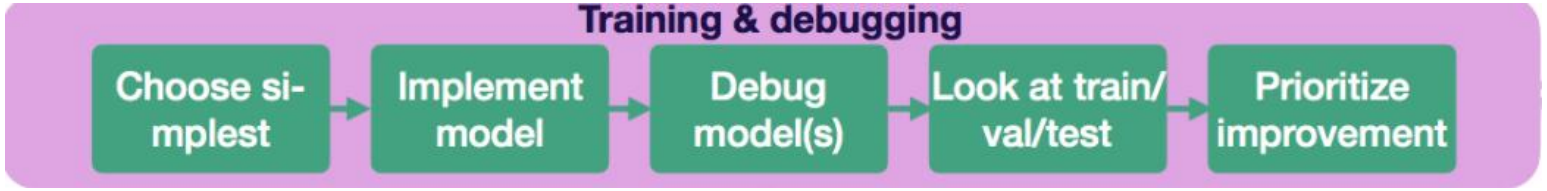
Training & Debugging



- **딥러닝 문제해결의 어려움**

  - 실무진들이 보통 느끼기에
  - 80~90%의 시간을 디버깅과 튜닝에 사용하고,
  - 오직 10~20%의 시간을 수학공식을 유도하고 모델을 구현하는데 사용한다.

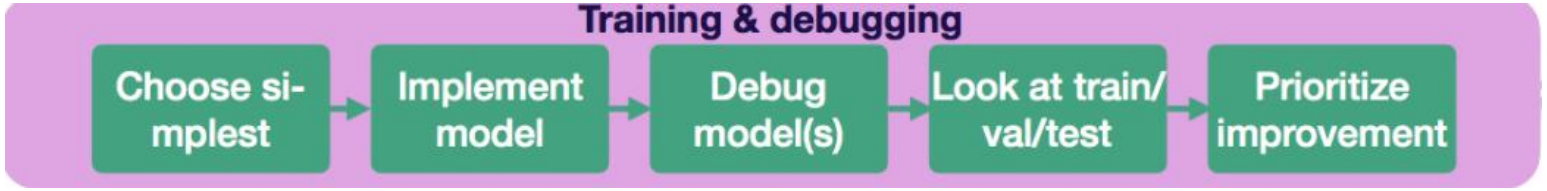
Training & Debugging



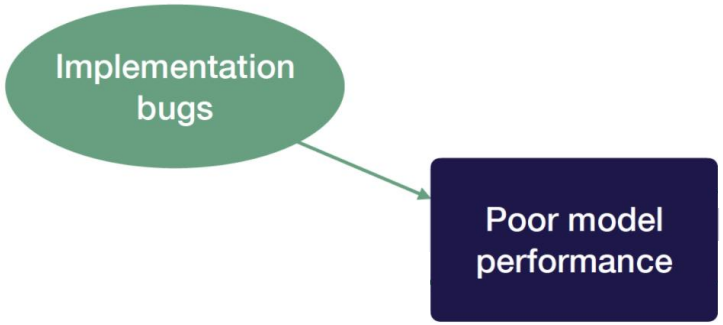
- 모델 성능을 떨어뜨리는 요인들:

Poor model performance

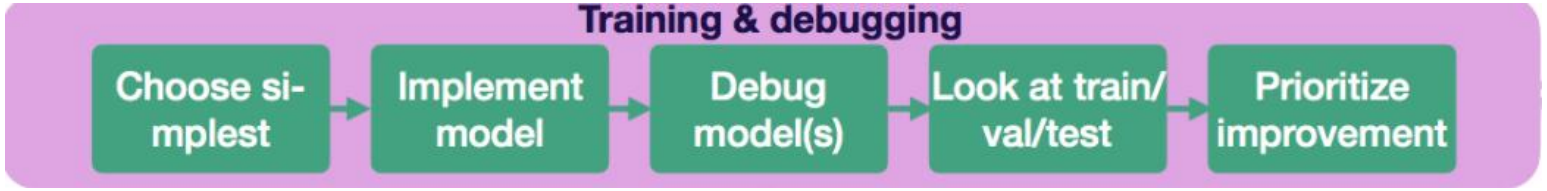
Training & Debugging



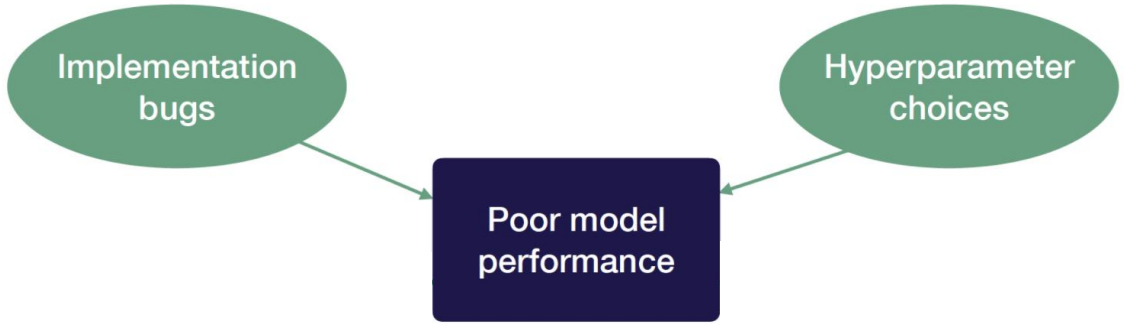
- 모델 성능을 떨어뜨리는 요인들:



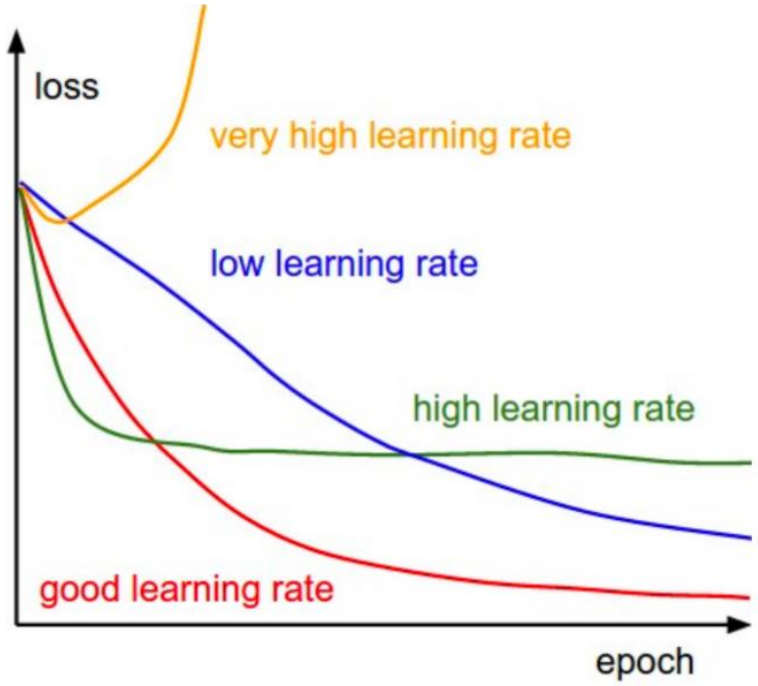
Training & Debugging



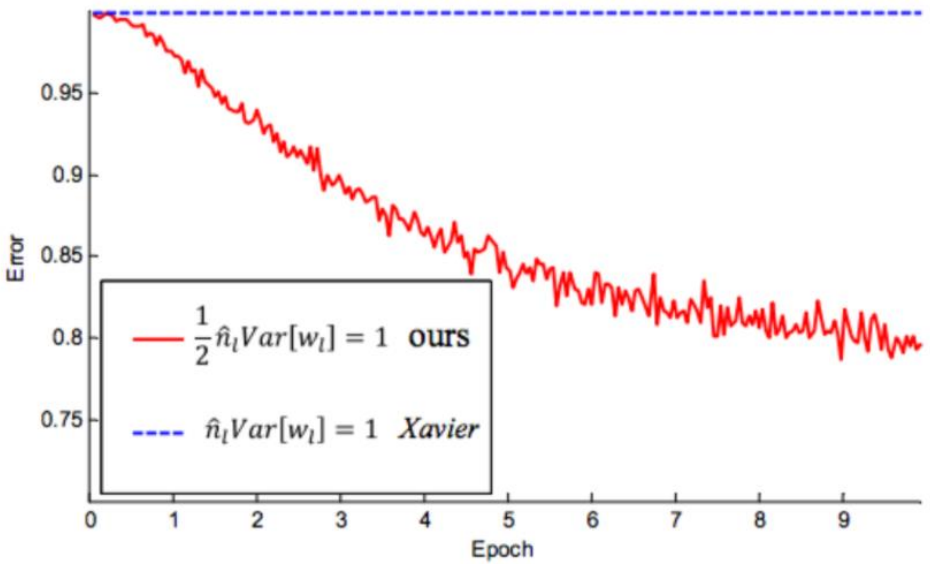
- 모델 성능을 떨어뜨리는 요인들:



Hyperparameter 선택의 영향

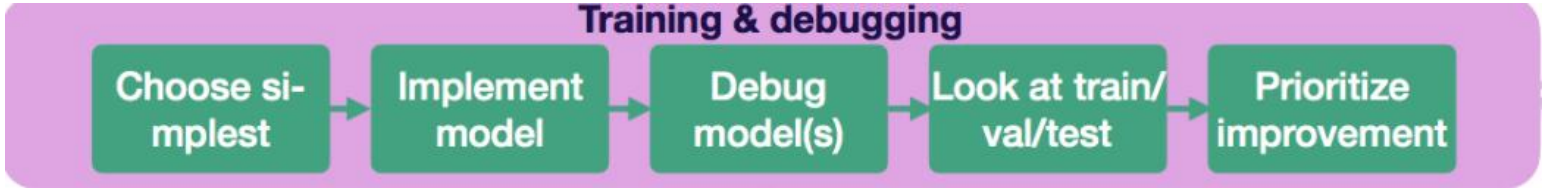


Andrej Karpathy, CS231n course notes

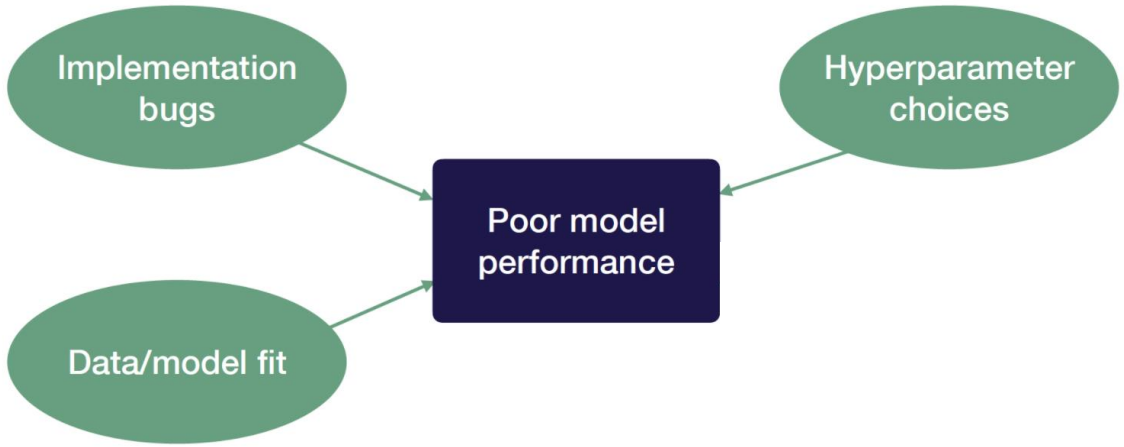


He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

Training & Debugging

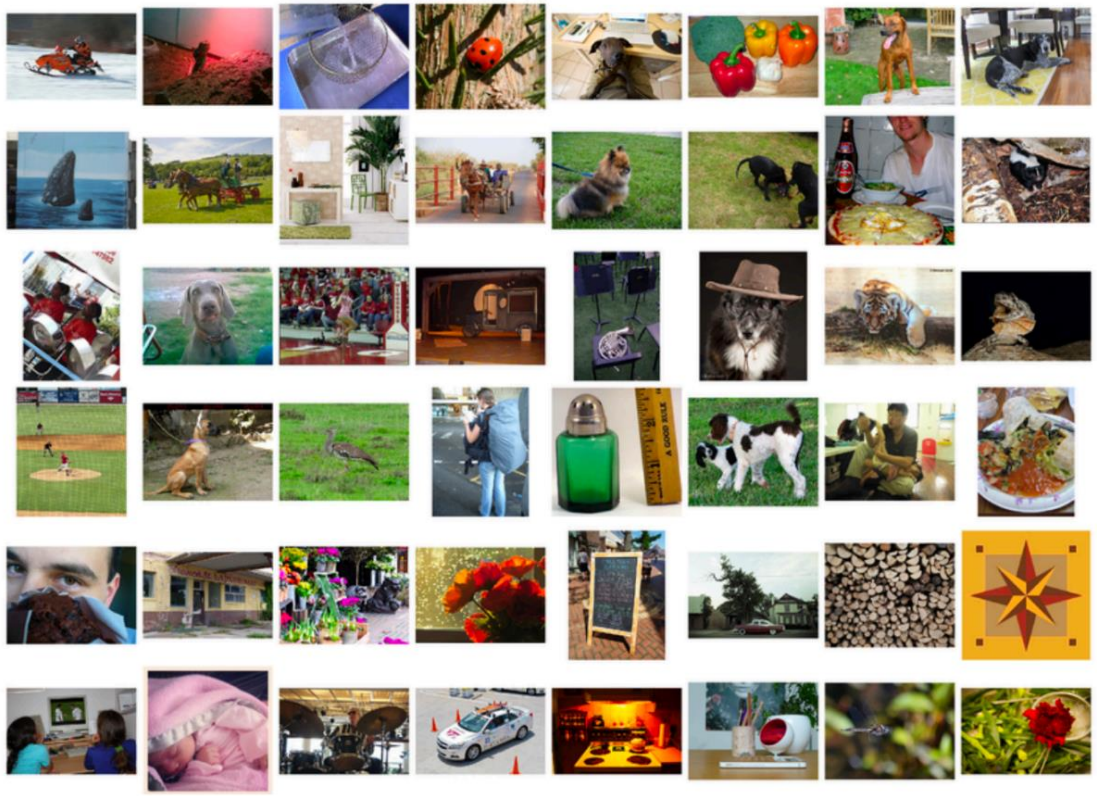


- 모델 성능을 떨어뜨리는 요인들:



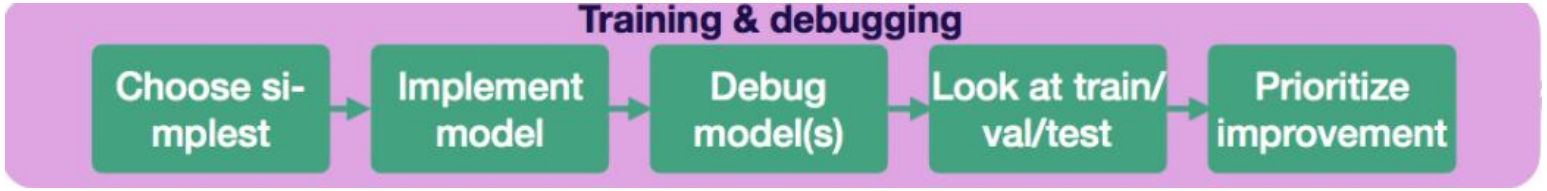
Data / Model Fit

Data from the paper: ImageNet

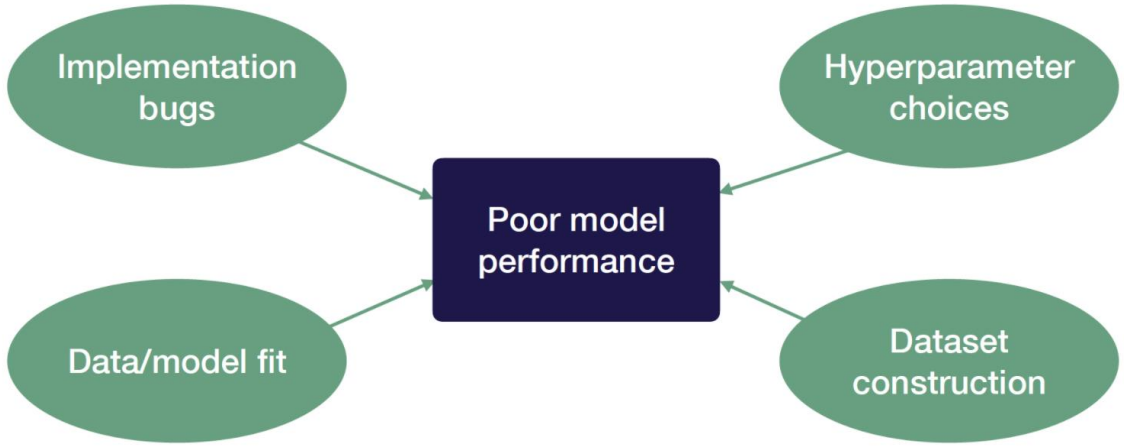


Yours: self-driving car images

Training & Debugging



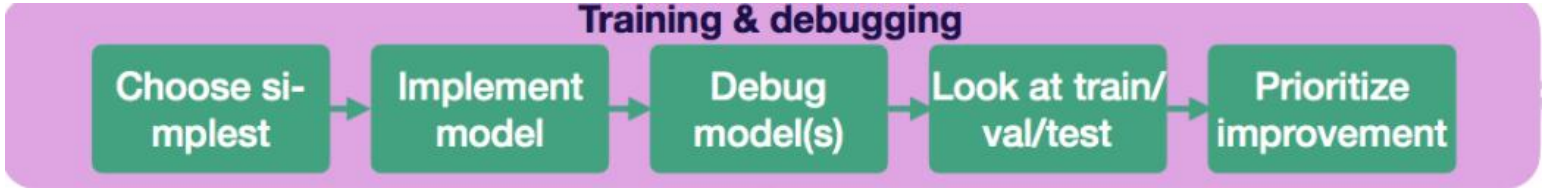
- 모델 성능을 떨어뜨리는 요인들:



일반적으로 dataset 구성 시 생기는 이슈들

- Not enough Dataset
- Noisy labels
- Imbalanced classes
- Train / test from different distributions

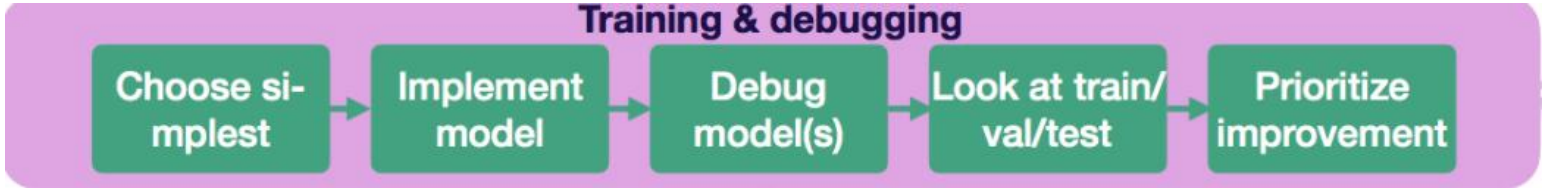
Training & Debugging



- **왜 딥러닝에서 문제해결이 어려운가?**
  - 버그가 있는지 알기 어려움.
  - 같은 성능 저하에 대해서 가능한 원인이 많음.
  - 하이퍼파라미터와 데이터 구성의 작은 변화에도 학습결과가 예민하게 변화함.

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging



- 딥러닝 문제 해결에 필요한 마인드셋

Pessimism

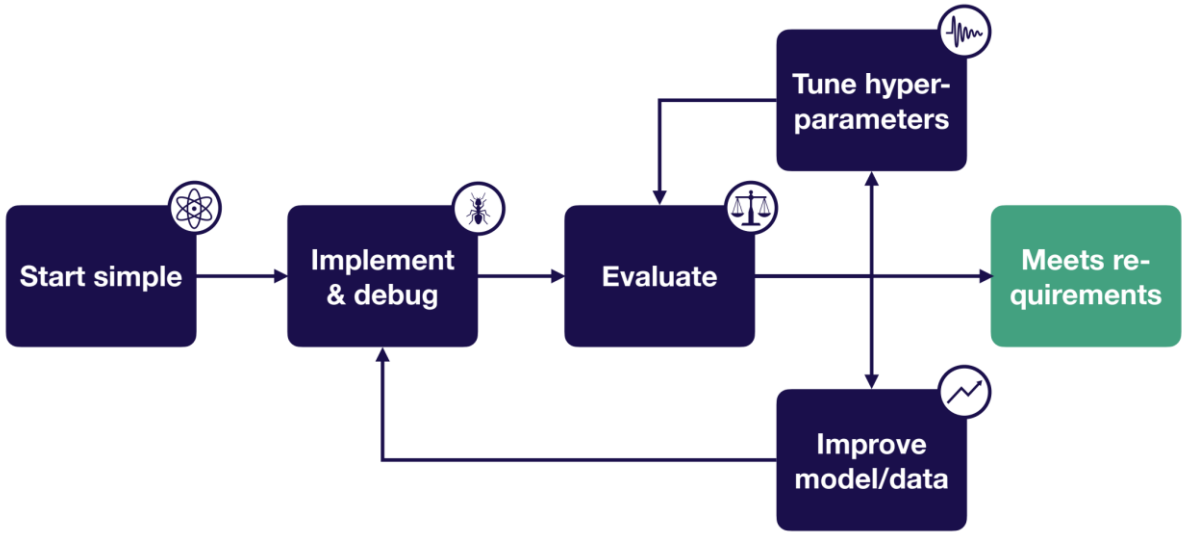
비관적으로 보자

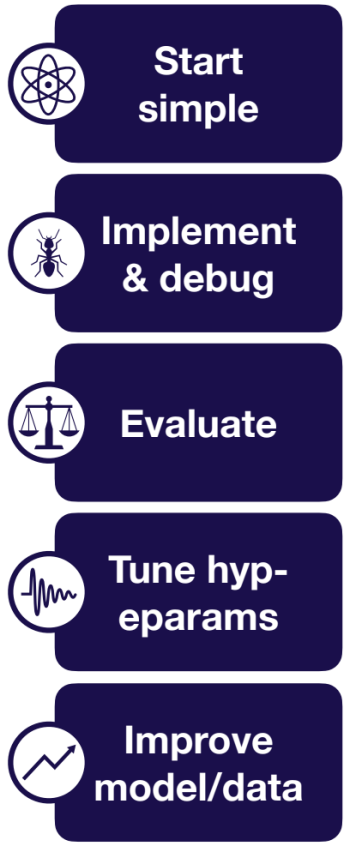
- 문제의 원인을 명확히 알기 어렵기 때문에, 먼저 간단하게 시작하고 (start simple), 점차 복잡도를 올리자

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging

- 딥러닝 문제해결을 위한 전략





- Choose the simplest model & data possible (e.g., LeNet on a subset of your data)
- Once model runs, overfit a single batch & reproduce a known result
- Apply the bias-variance decomposition to decide what to do next
- Use coarse-to-fine random searches
- Make your model bigger if you underfit; add data or regularize if you overfit

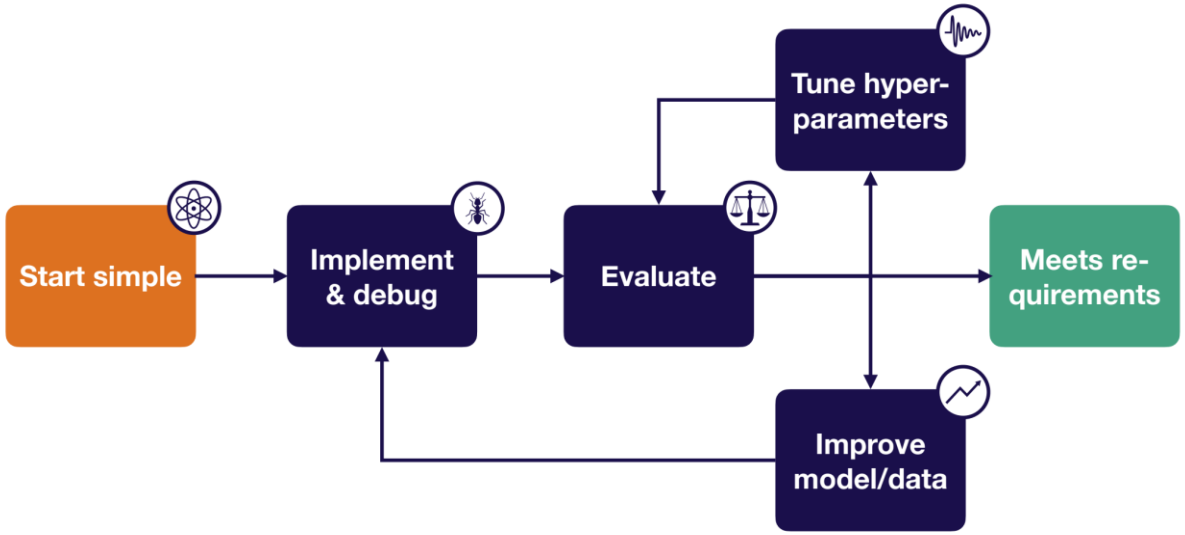
- 가정

1. 테스트 셋이 존재한다.
2. 향상시킬 간단한 metric이 있다.
3. human-level 성능이나 출간된 결과물이나,이전의 베이스라인 등 목표 성능이 있다고 가정한다.

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging

- 딥러닝 문제해결을 위한 전략



Starting simple

Steps

a. Choose a simple architecture — 간단한 모델 선택

b. Use sensible defaults — 검증된 초기 세팅

c. Normalize inputs — 입력 정규화

d. Simplify the problem — 문제의 단순화

https://fullstackdeeplearning.com/spring2021/lecture-7/

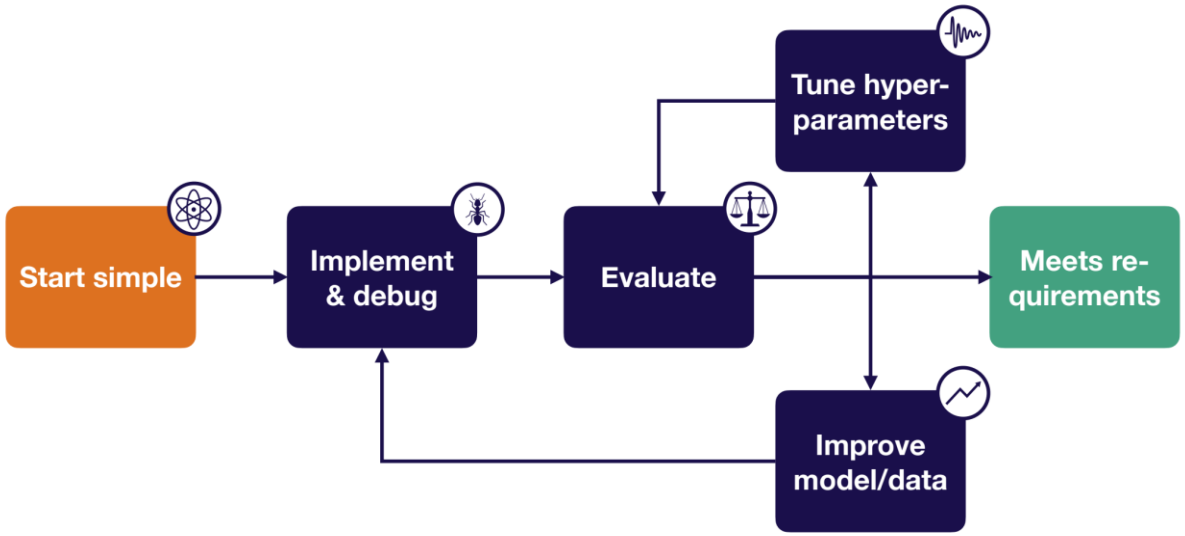
간단한 모델 아키텍처 선택

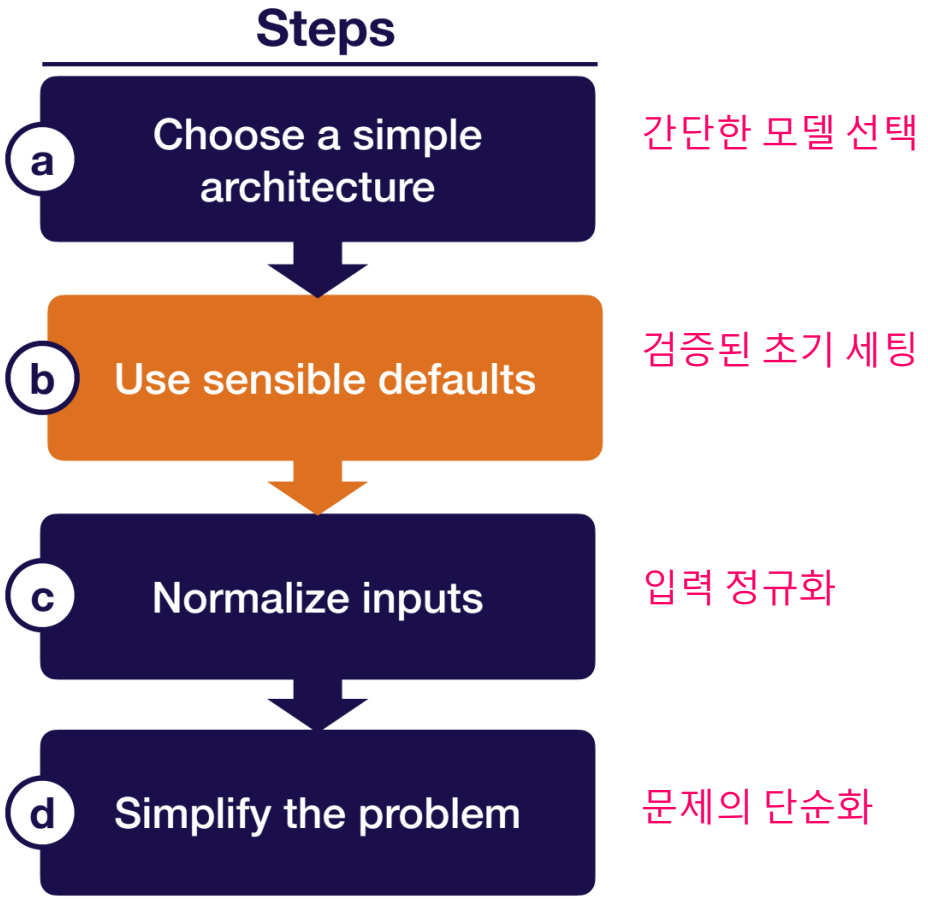
	Start here	Consider using this later
Images	LeNet-like architecture	ResNet
Sequences	LSTM with one hidden layer (or temporal convs)	Attention model or WaveNet-like model
Other	Fully connected neural net with one hidden layer	Problem-dependent

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging – Starting simple

- 딥러닝 문제해결을 위한 전략





https://fullstackdeeplearning.com/spring2021/lecture-7/

일반적으로 잘 동작하는 default 세팅 사용

- 일반적으로 추천되는 것을 먼저 사용
- **옵티마이저(Optimizer):** Adam optimizer, learning rate 3e-4
- **활성화 함수(Activation Function):** relu (FC 과 Conv 모델), tanh (LSTM모델)
- **초기화 (Initialization):** He normal (relu), Glorot normal (tanh)
- **Regulaization** 및 **Data normalization (batch norm)** 은 최종 성능을 끌어올릴 때 쓰자.
  (처음부터 쓰면 버그 유발 가능성)

- **He et al. normal (used for ReLU)**

$$\mathcal{N}\left(0, \sqrt{\frac{2}{n}}\right)$$

- **Glorot normal (used for tanh)**

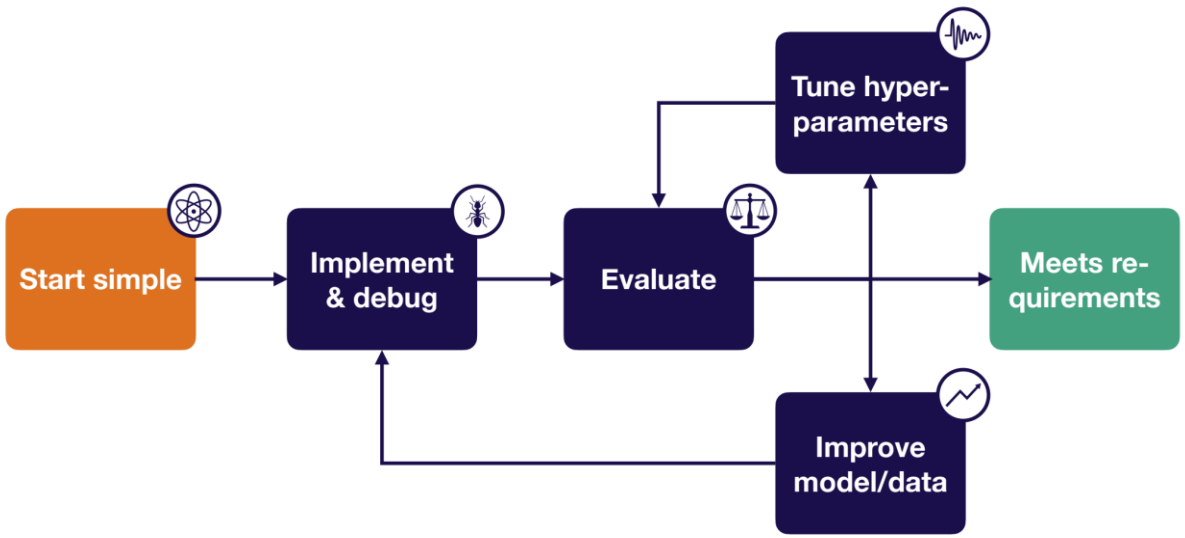
$$\mathcal{N}\left(0, \sqrt{\frac{2}{n+m}}\right)$$

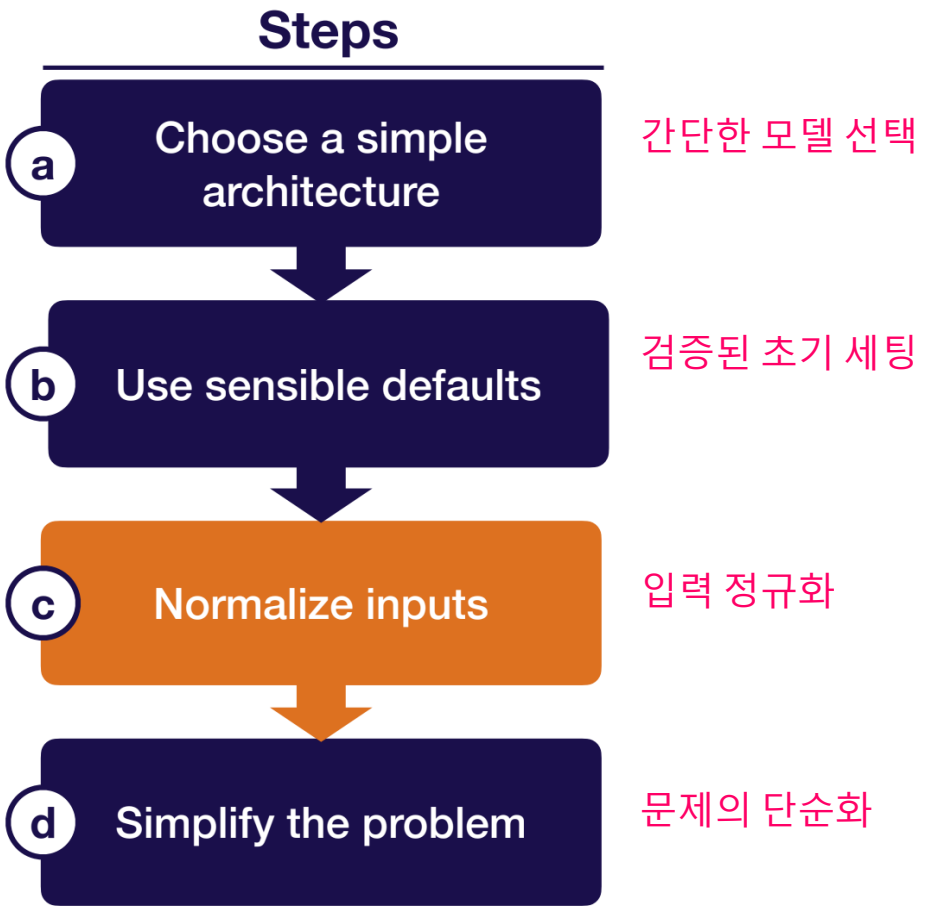
(n is the number of inputs, m is the number of outputs)

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging - Starting simple

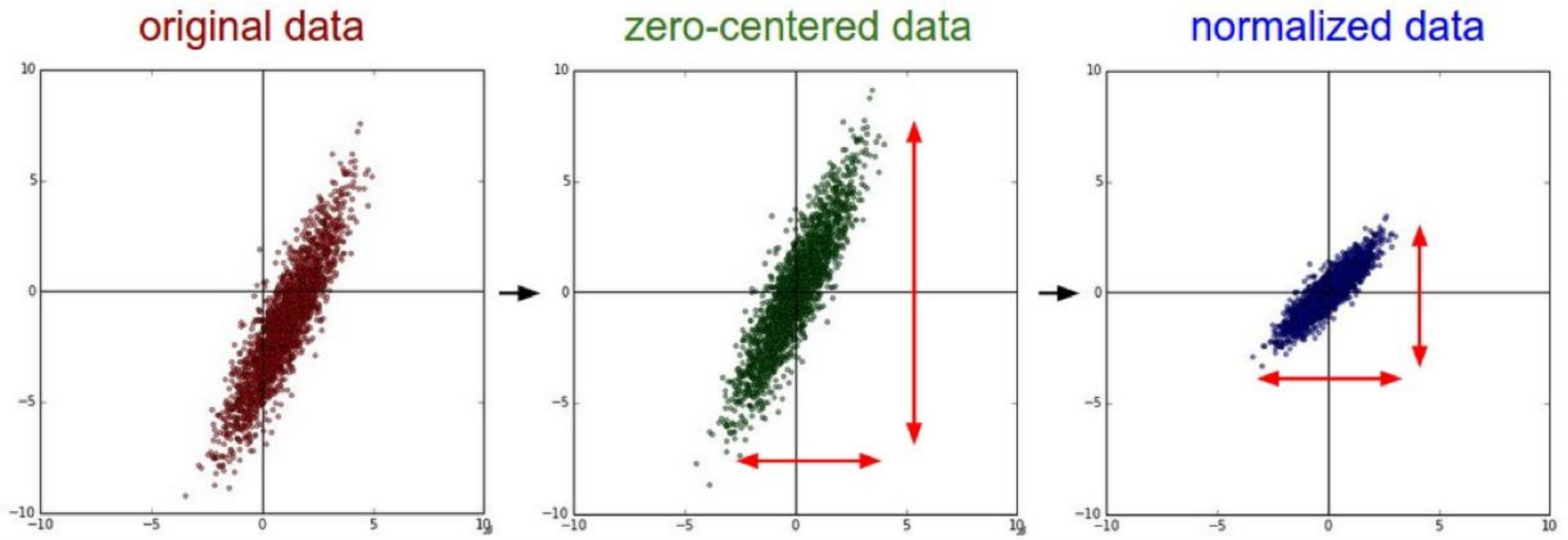
- 딥러닝 문제해결을 위한 전략





https://fullstackdeeplearning.com/spring2021/lecture-7/

Normalize Input



$$z = \frac{x - \mu}{\sigma}$$

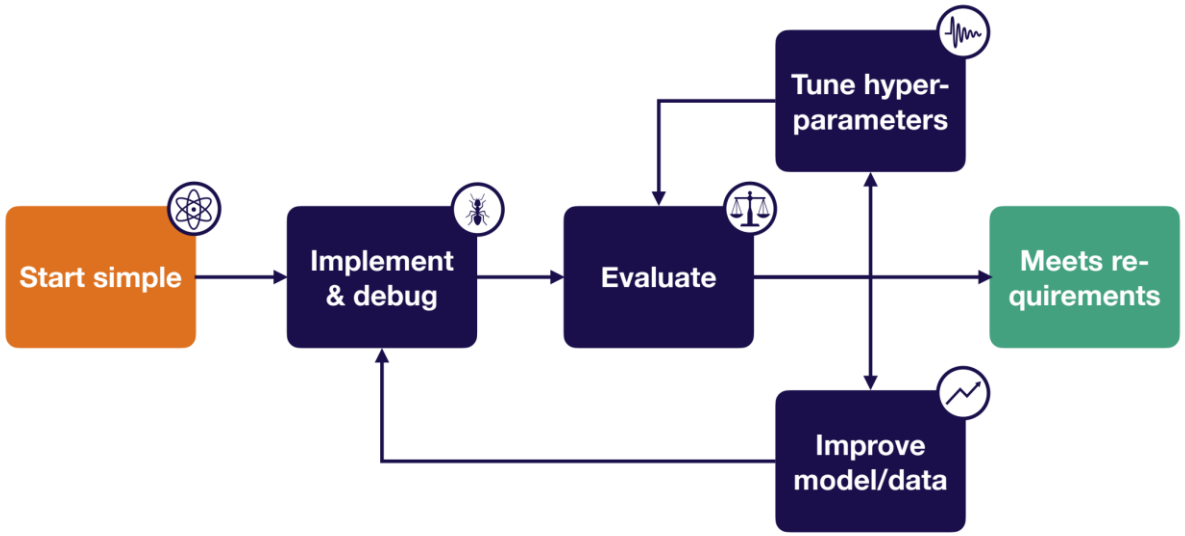
μ = Mean

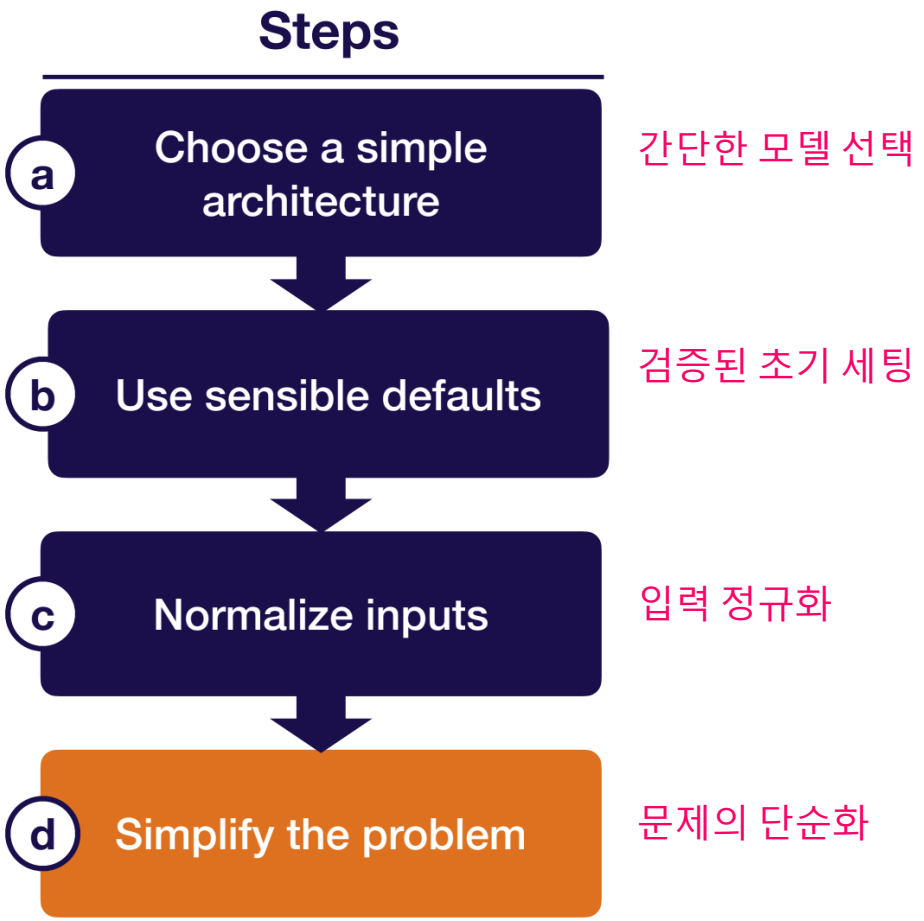
σ = Standard Deviation

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \in [0\ ,\ 1]$$

Training & Debugging - Starting simple

- 딥러닝 문제해결을 위한 전략





https://fullstackdeeplearning.com/spring2021/lecture-7/

문제를 단순화 하기

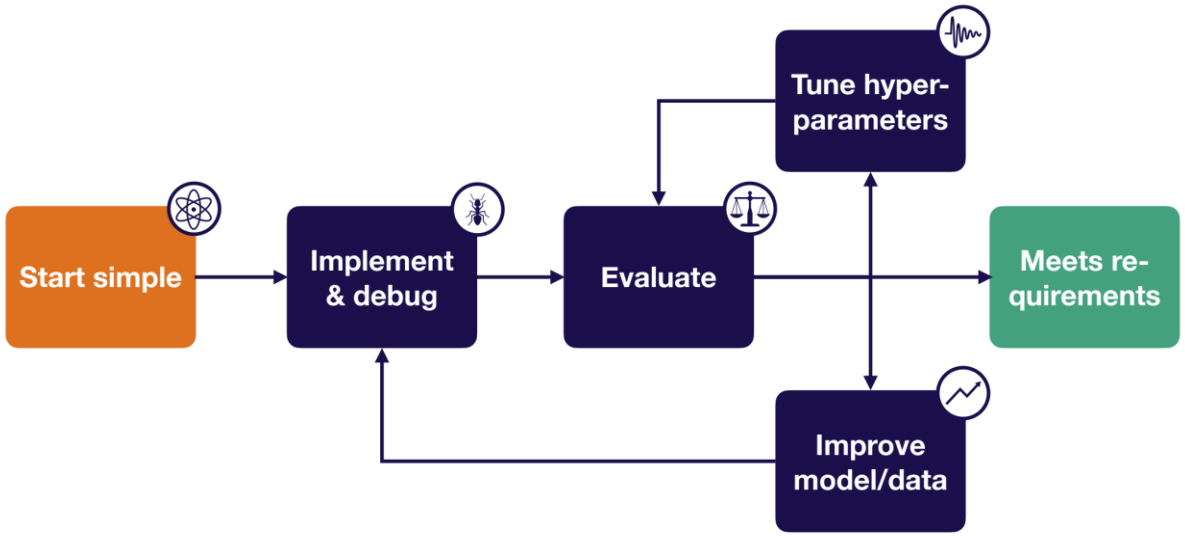
- 적은 학습데이터로 시작하자 (~10,000).
- 클래스의 개수 나 이미지 사이즈 등을 줄여서 시도해보자.
- 더 간단한 synthetic 학습데이터를 만들어서 시작해보자.

- 모델이 문제인지 task자체가 어려운지 모르니 우선 task를 최대한 간단하게 세팅하고 시작하자.

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Starting simple

- 딥러닝 문제해결을 위한 전략

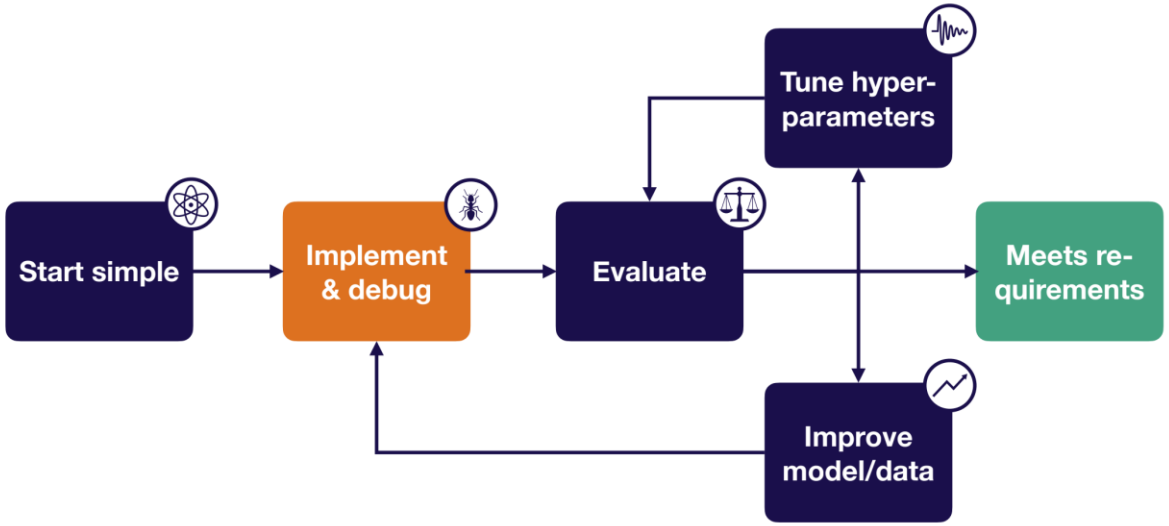


Steps	Summary
a Choose a simple architecture	• LeNet, LSTM, or fully connected
b Use sensible defaults	• Adam optimizer & no regularization
c Normalize inputs	• Subtract mean and divide by std, or just divide by 255 (ims)
d Simplify the problem	• Start with a simpler version of your problem (e.g., smaller dataset)

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging

- 딥러닝 문제해결을 위한 전략



- 5가지 가장 일반적인 딥러닝 버그들

1. Tensor shape의 불일치

2. 입력데이터에 대한 부정확한 전처리
   - 전처리를 하지 않거나 여러 번 전처리함.

3. Loss 함수에 부정확한 입력 형식
   - Loss함수에 잘못된 형태의 입력을 넣어줌.

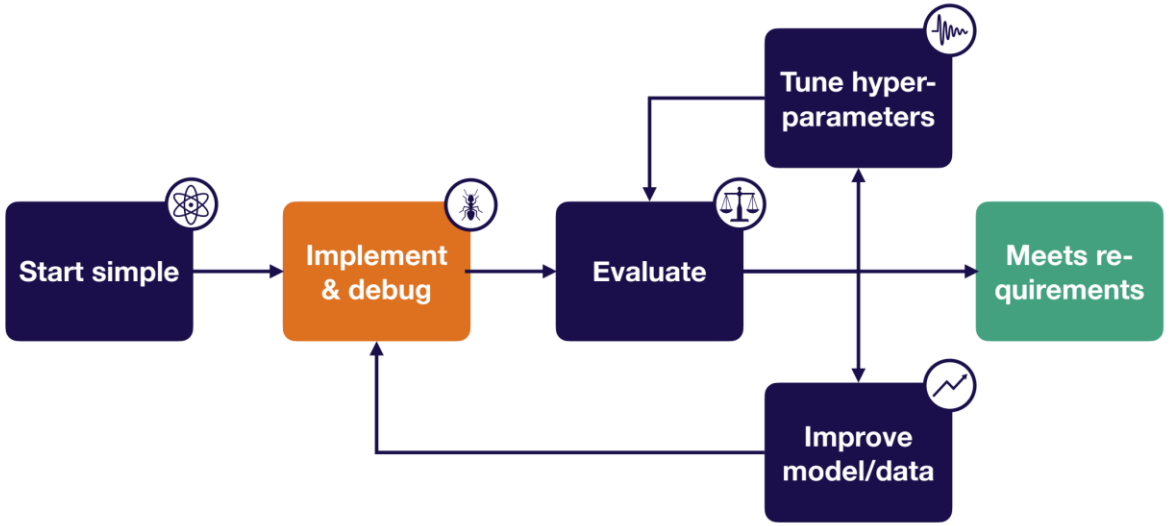
4. 학습 / 평가 모드의 잘못된 설정
   - 학습모드로 전환없이 평가모드에서 dropout이나 batch norm적용

5. 수치적 불안정성 (Numerical Instability)
   - Inf/NaN (exp, log, div opertation)

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging

- 딥러닝 문제해결을 위한 전략



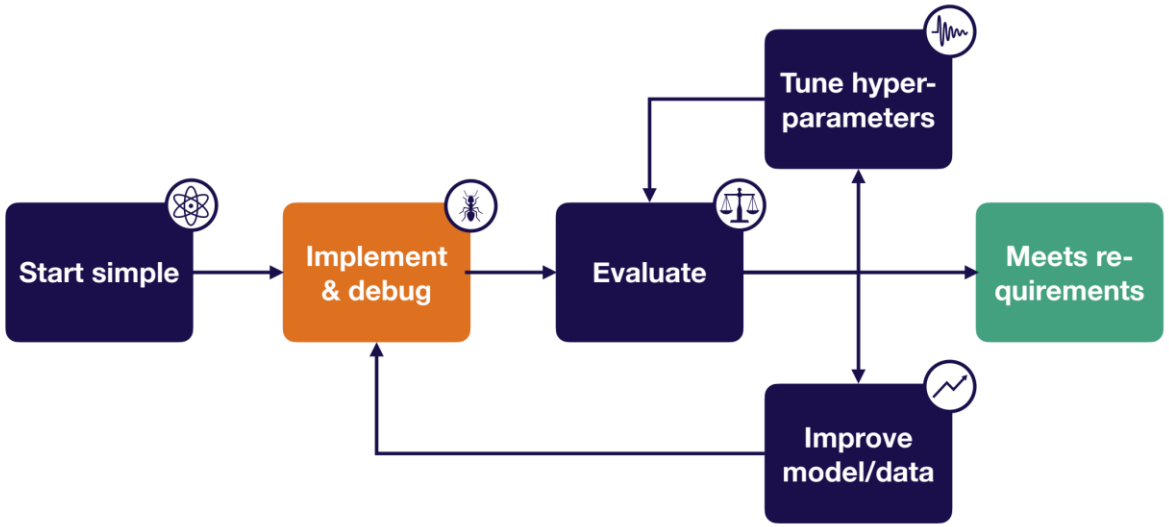
- 모델 구현을 위한 일반적인 조언

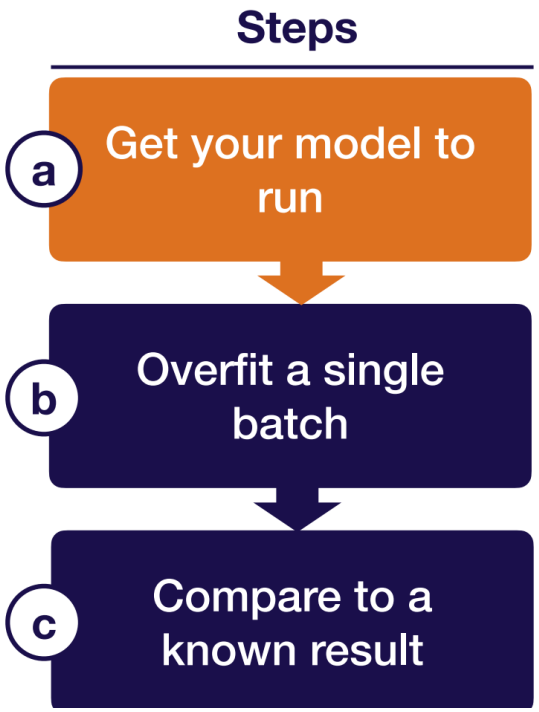
1. 가벼운 구현
   - V1은 가능하면 간소하게 (200줄 미만)
2. 이미 구현된 컴포넌트 사용
   - 텐서플로우 보단 케라스
   - Tf.nn.relu(tf/mamul(W,x)) 대신 tf.layers.dense()
3. 복잡한 데이터 파이프라인은 나중에 적용
   - 하나의 gpu memory에 올릴 수 있는 데이터셋을 갖고 시작

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Implement & debug

- 딥러닝 문제해결을 위한 전략





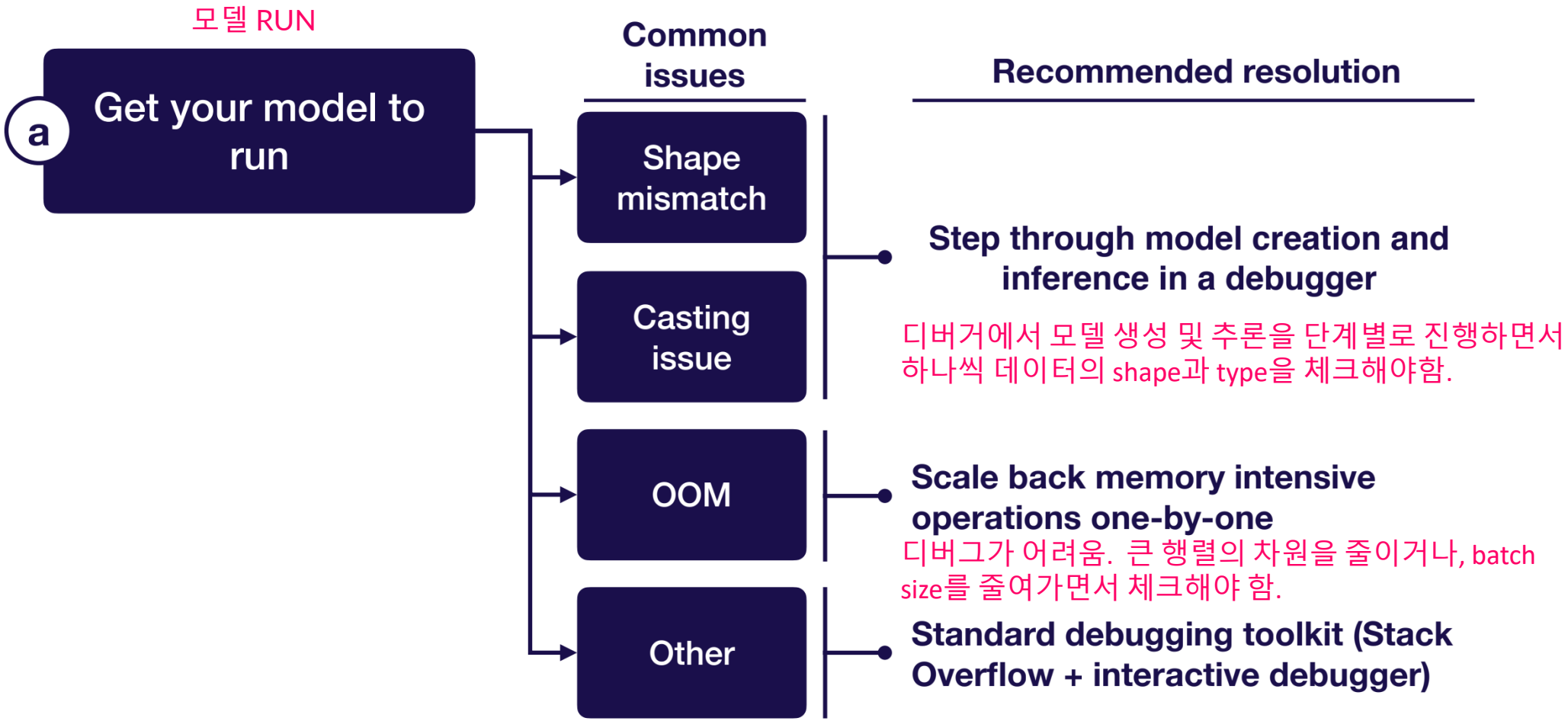
모델 RUN

single batch에 오버피팅

알고 있는 결과와 비교

https://fullstackdeeplearning.com/spring2021/lecture-7/

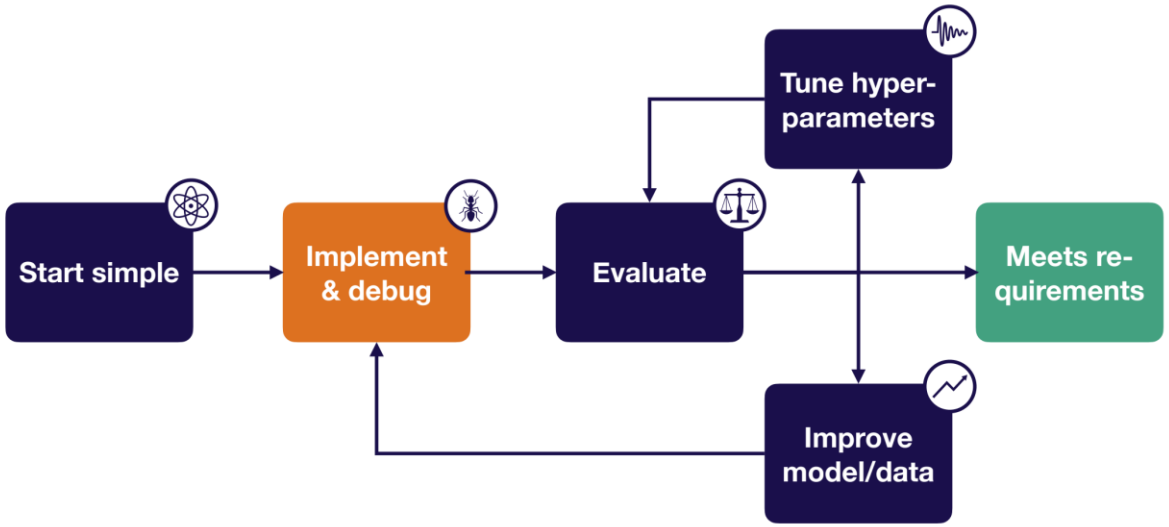
Training & Debugging- Implement & debug

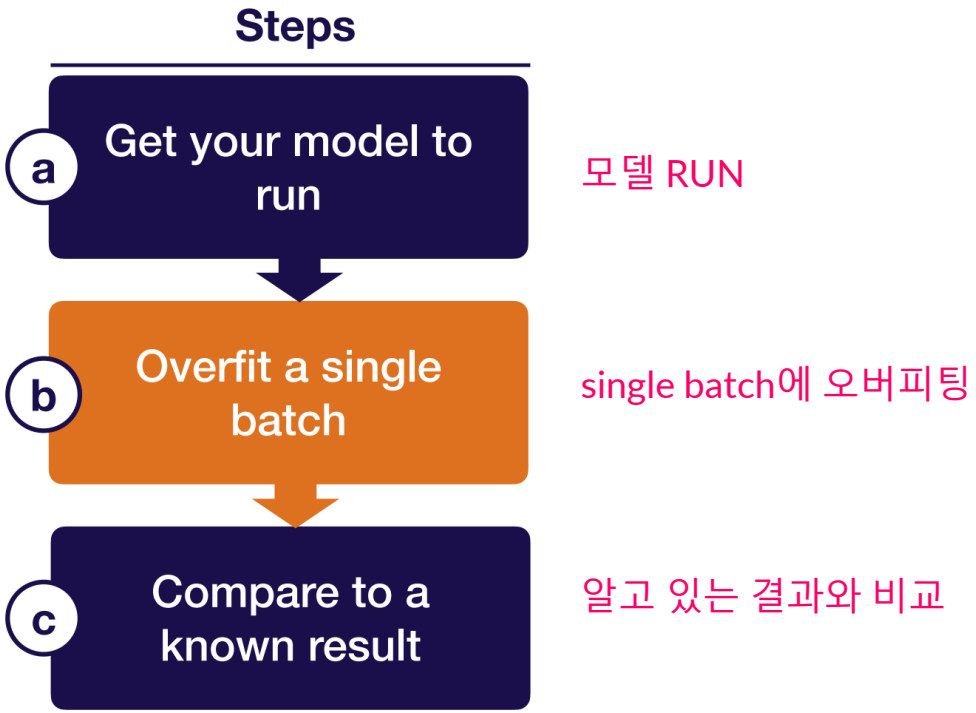


https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Implement & debug

- 딥러닝 문제해결을 위한 전략





모델 RUN

single batch에 오버피팅

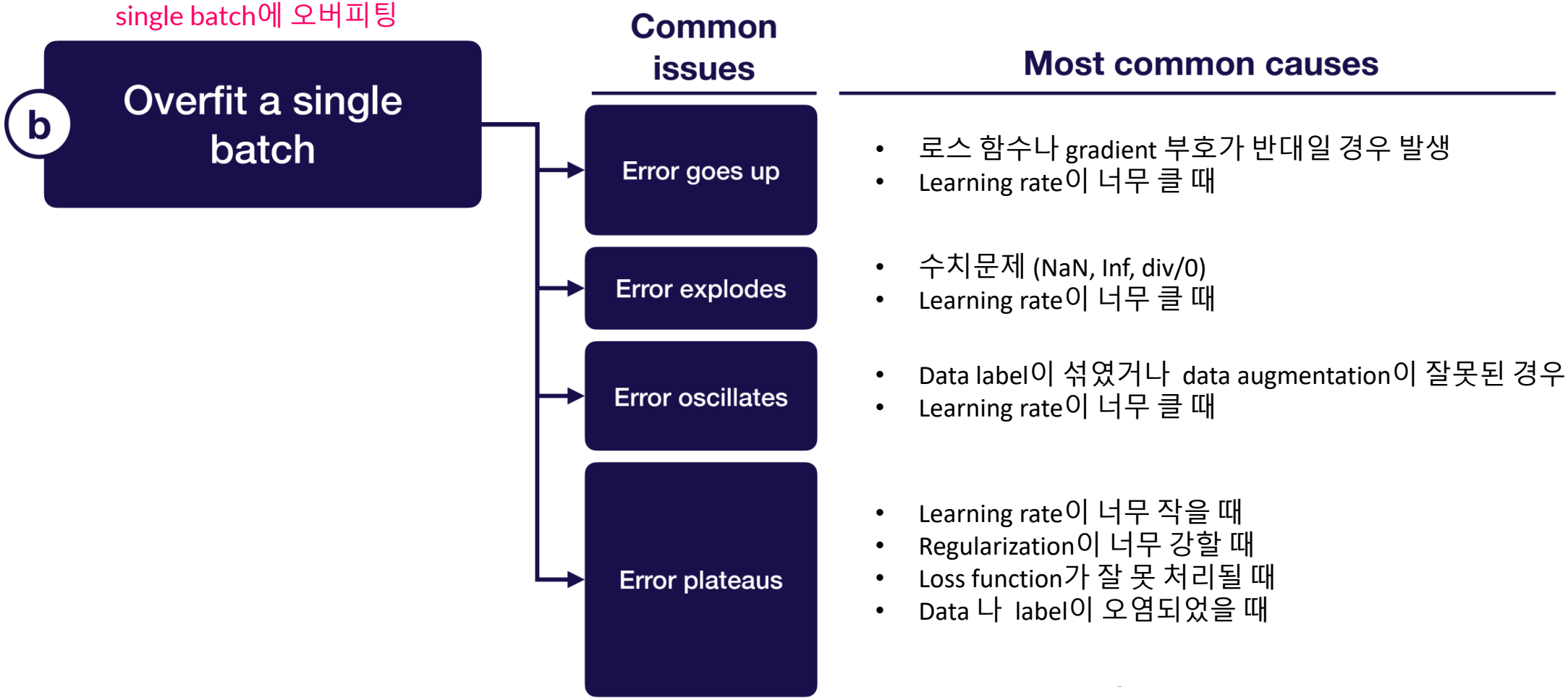
알고 있는 결과와 비교

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Implement & debug

single batch에 오버피팅

b **Overfit a single batch**



- 로스 함수나 gradient 부호가 반대일 경우 발생
- Learning rate이 너무 클 때

- 수치문제 (NaN, Inf, div/0)
- Learning rate이 너무 클 때

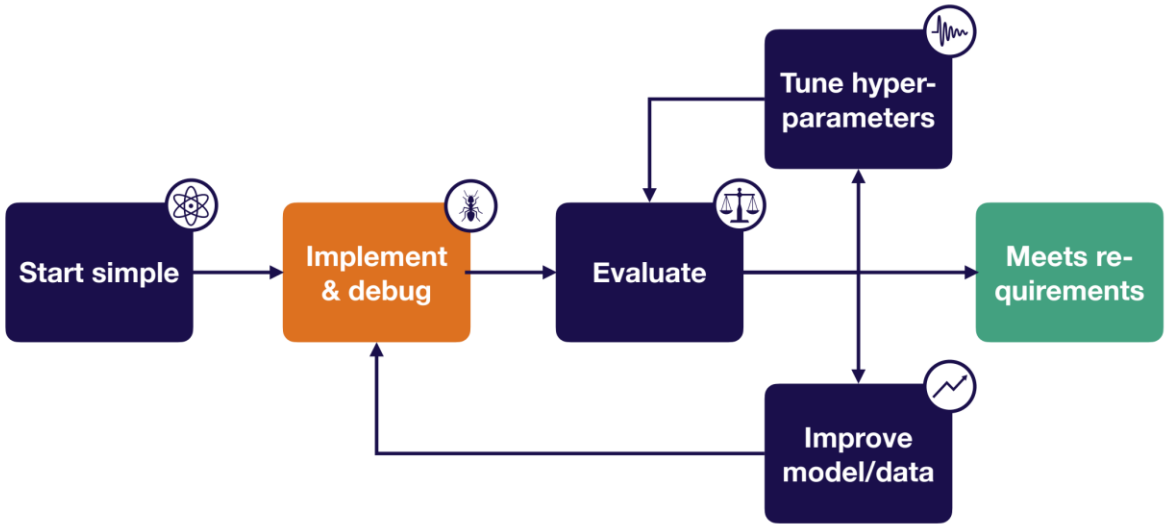
- Data label이 섞였거나 data augmentation이 잘못된 경우
- Learning rate이 너무 클 때

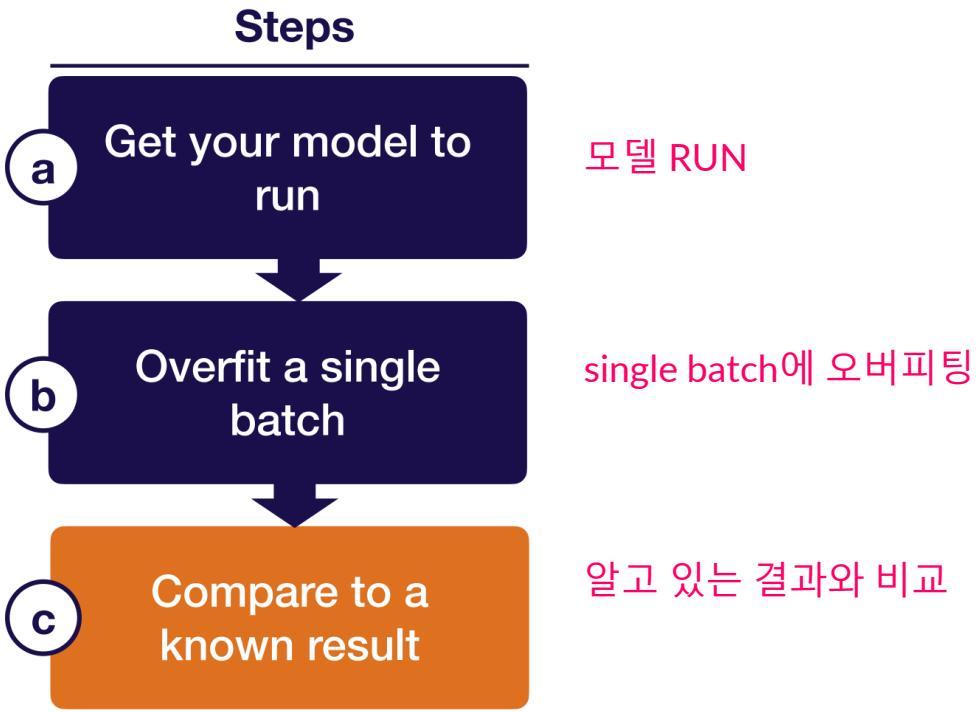
- Learning rate이 너무 작을 때
- Regularization이 너무 강할 때
- Loss function가 잘 못 처리될 때
- Data 나 label이 오염되었을 때

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Implement & debug

- 딥러닝 문제해결을 위한 전략





모델 RUN

single batch에 오버피팅

알고 있는 결과와 비교

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Implement & debug

- **Compare to a known results** (알려진 결과들과 비교)

More useful

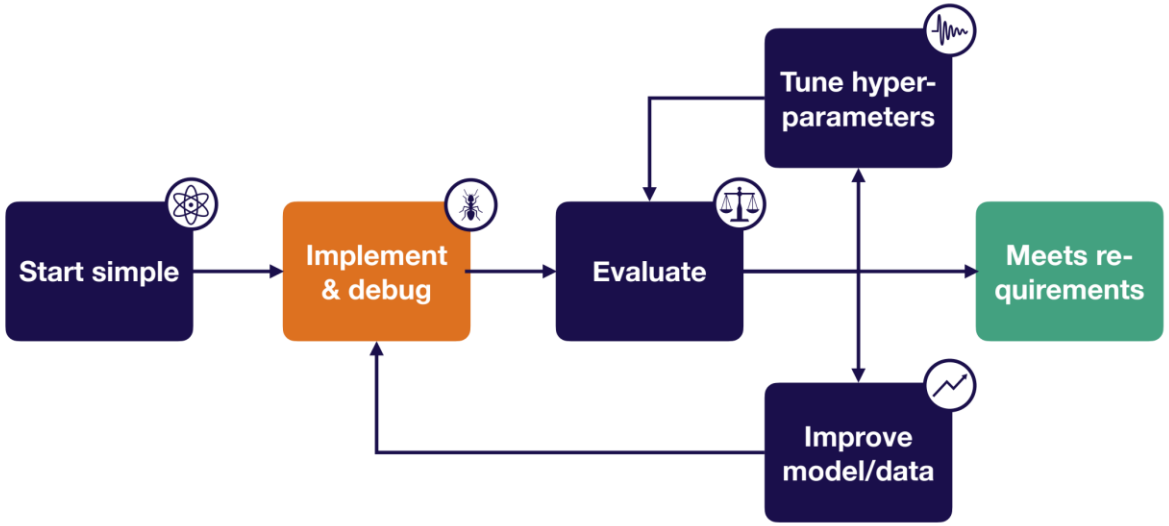
Less useful

- 공식모델 구현체에서 유사한 데이터셋의 성능과 비교
- MNIST같은 벤치마크 데이터셋에서 공식 모델 구현체의 성능과 비교
- 비공식 모델 구현과 비교
- 코드가 없을 경우 논문에 나온 결과들과 비교
- MNIST같은 벤치마크 데이터셋에 대한 내 모델의 성능과 비교
- 유사한 데이터셋의 유사한 모델의 성능과 비교
- 정말 간단한 베이스 라인(평균 출력이나 선형 회귀 등)과 비교

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Implement & debug

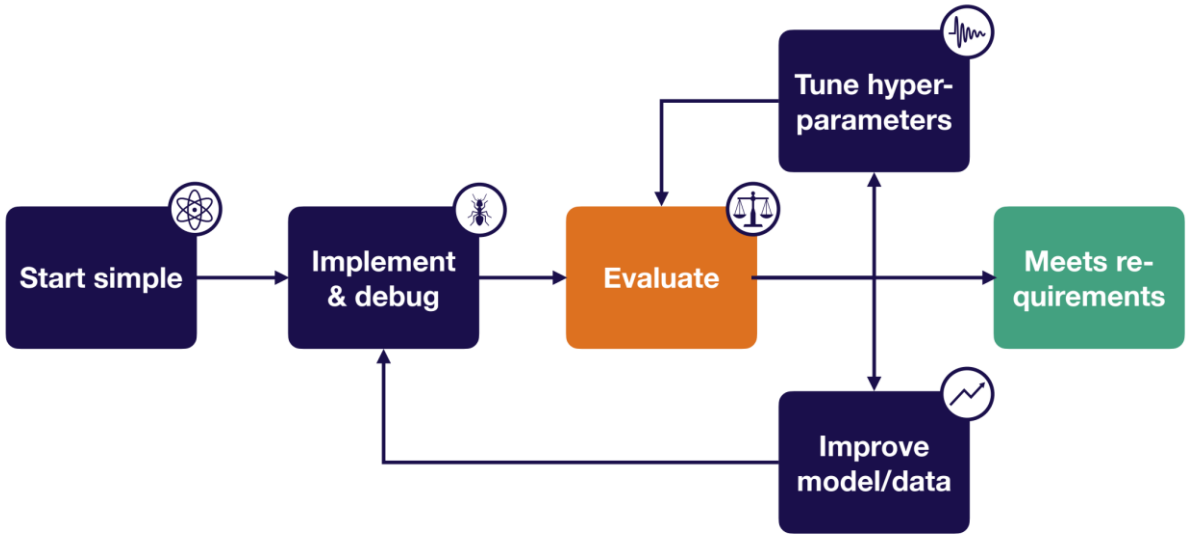
- 딥러닝 문제해결을 위한 전략



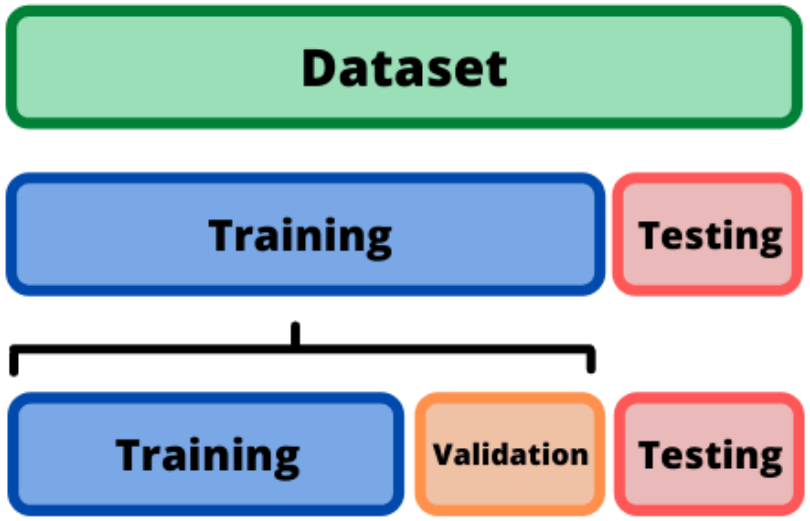
Steps	Summary
a Get your model to run	• Step through in debugger & watch out for shape, casting, and OOM errors
b Overfit a single batch	• Look for corrupted data, over-regularization, broadcasting errors
c Compare to a known result	• Keep iterating until model performs up to expectations

Training & Debugging- Evaluate

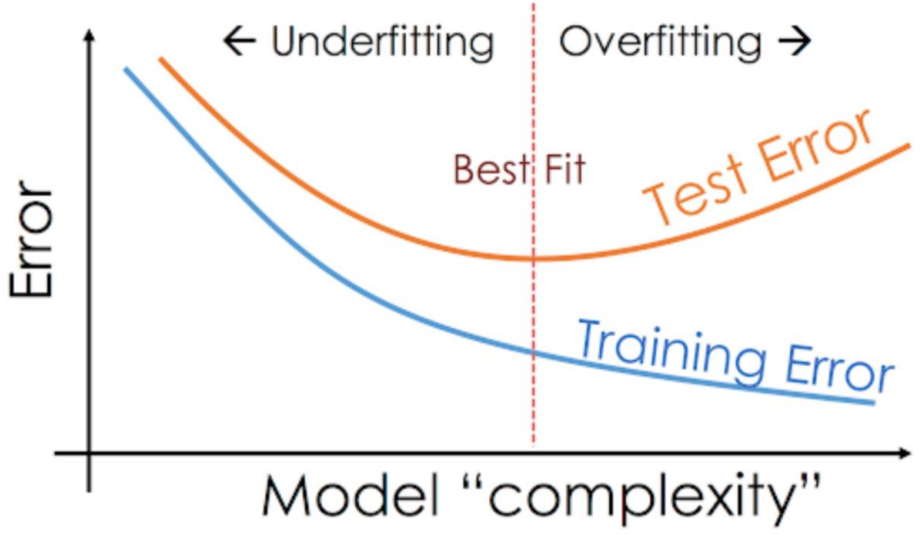
- 딥러닝 문제해결을 위한 전략



https://fullstackdeeplearning.com/spring2021/lecture-7/



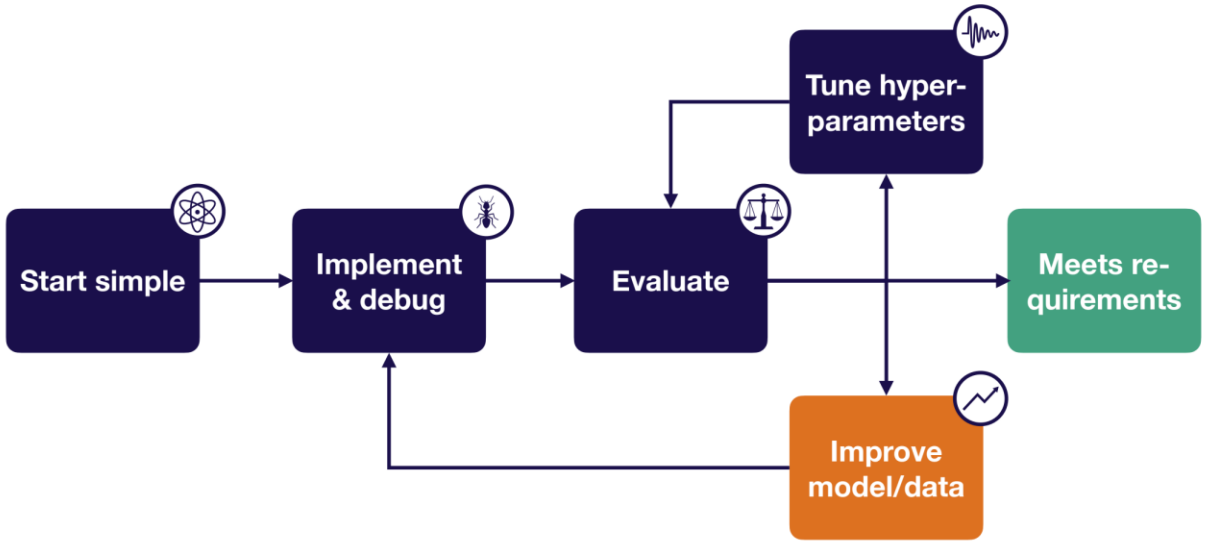
https://dhavalpatel2101992.wordpress.com/2021/05/21/kaggle-titanic-dataset-cleaning-split-data-into-train-validation-and-test-set/

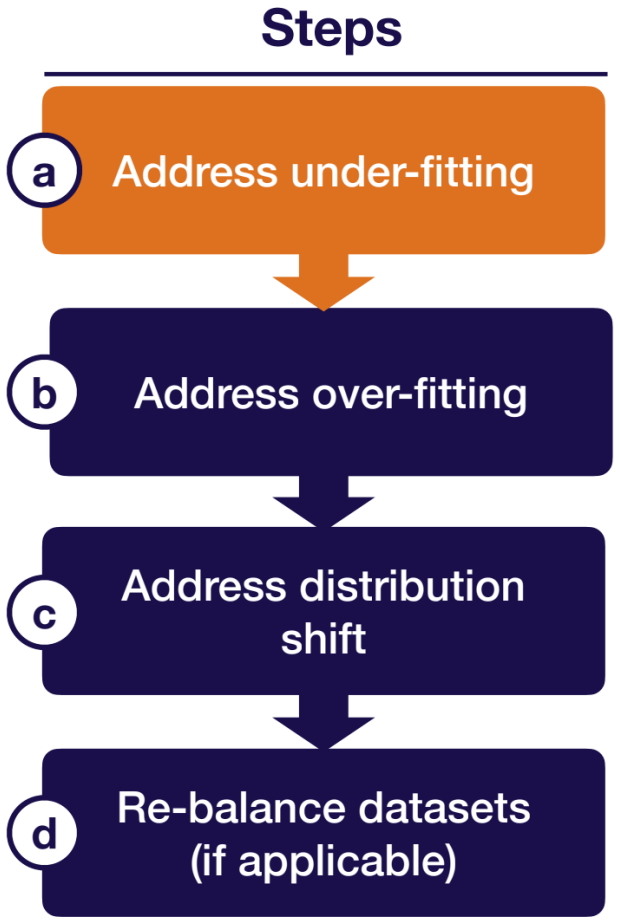


https://www.worshipwithyourlife.top/products.aspx?cname=overfit+training+data&cid=7

Training & Debugging- Improve model/data

- 딥러닝 문제해결을 위한 전략





https://fullstackdeeplearning.com/spring2021/lecture-7/

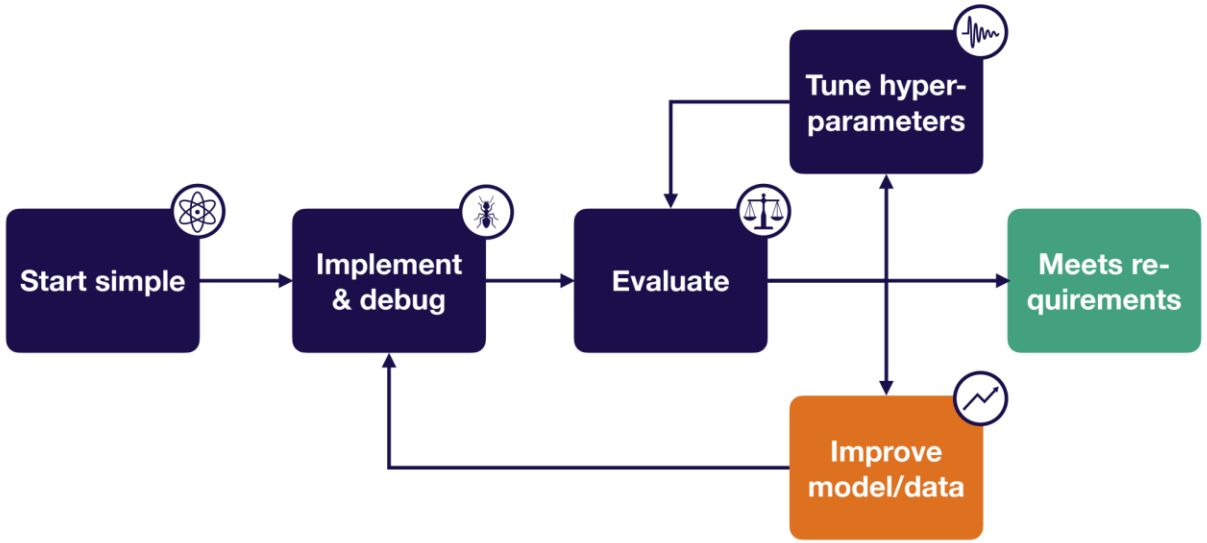
Training & Debugging- Improve model/data

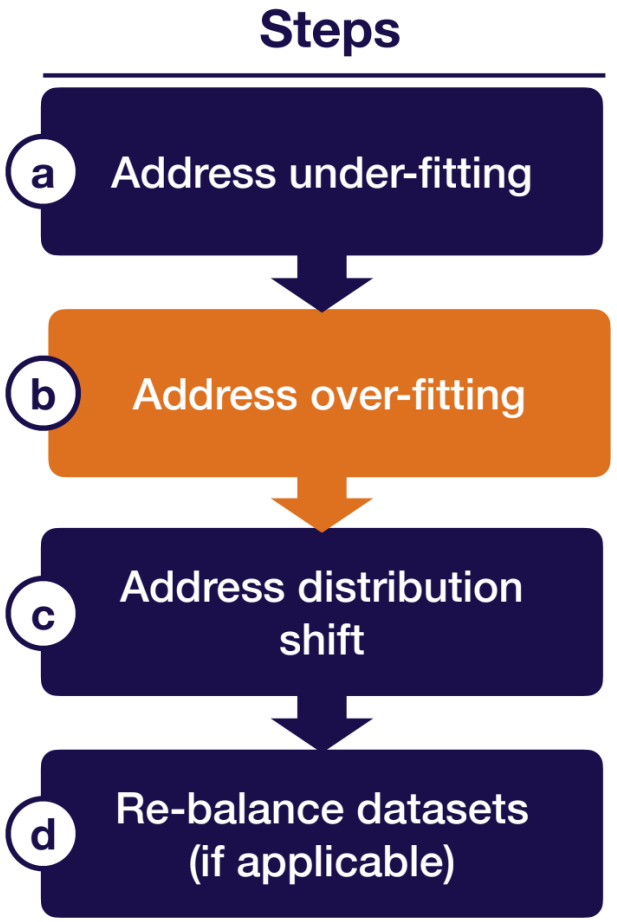
- **Under-fitting을 처리하는 방법 (Bias를 줄이기)**

1. 모델을 더 크게 만들어 본다. (레이어 추가 / 레이어 유닛 추가)
2. Regularization을 줄여 본다.
3. 에러 분석을 해본다.
4. SOTA 수준의 다른 모델을 써본다.
5. 하이퍼파라미터 튜닝한다.
6. Feature를 더 추가한다.

Training & Debugging- Improve model/data

- 딥러닝 문제해결을 위한 전략



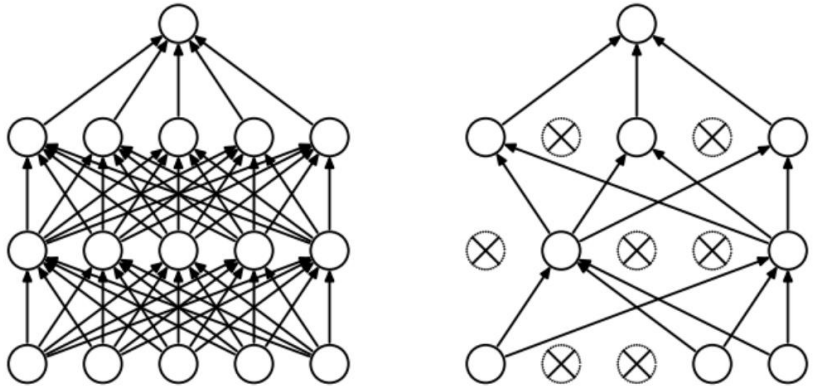


https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Improve model/data

- **Over-fitting을 처리하는 방법 (Variance를 줄이기)**

1. 학습데이터를 더 추가한다.
2. Data normalization을 추가해 본다 (batch norm, layer norm)
3. Regularization을 추가해 본다. (dropout, L1, L2 weight decay)
4. Data augmentation을 시도한다.
5. 에러 분석을 해본다.
6. 다른 모델을 사용해 본다.
7. 하이퍼파라미터 튜닝을 시도해 본다.



(a) Standard Neural Net (b) After applying dropout.

L1 Regularization

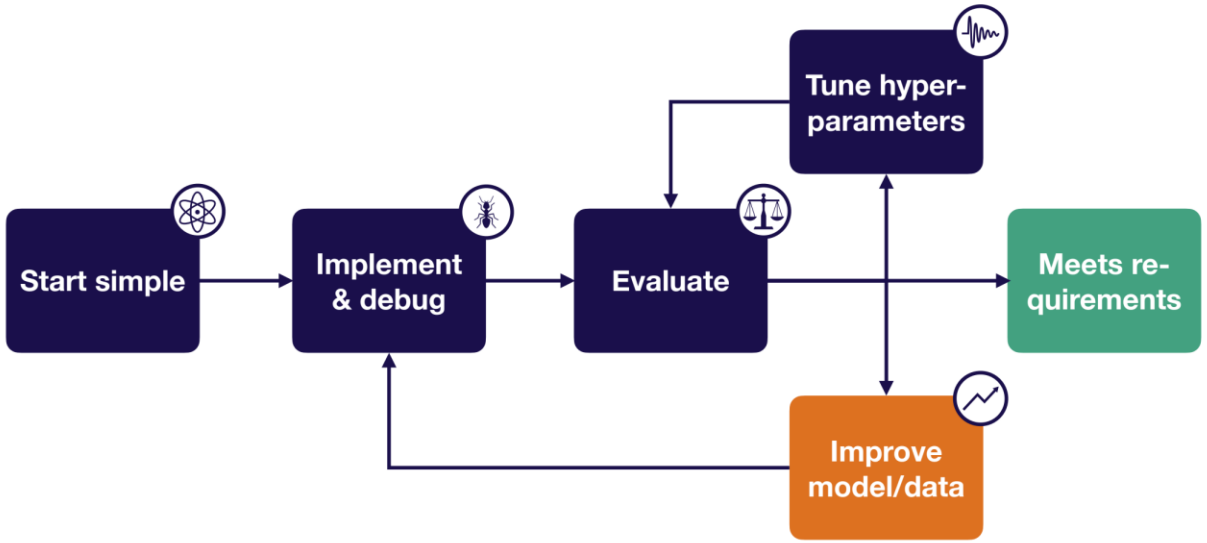
$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

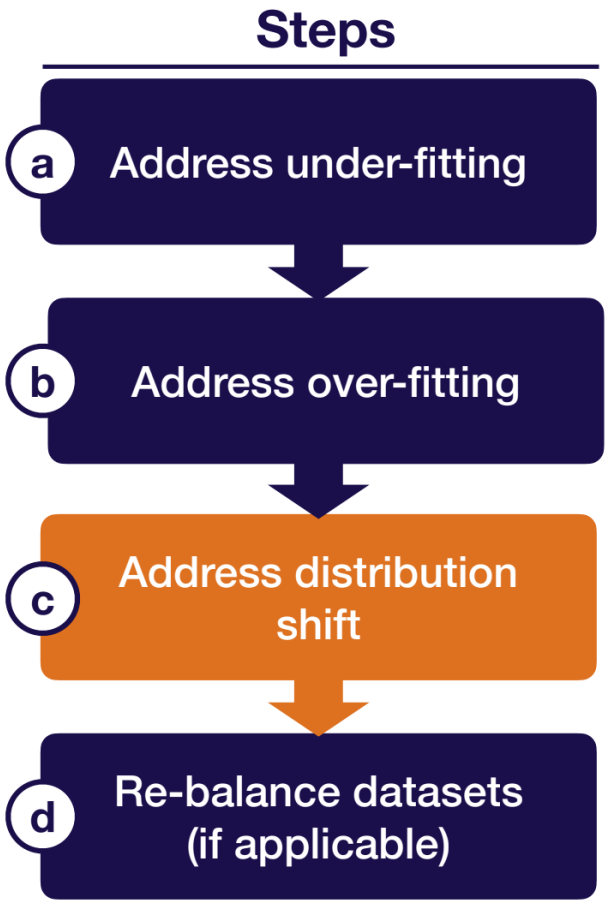
L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Training & Debugging- Improve model/data

- 딥러닝 문제해결을 위한 전략



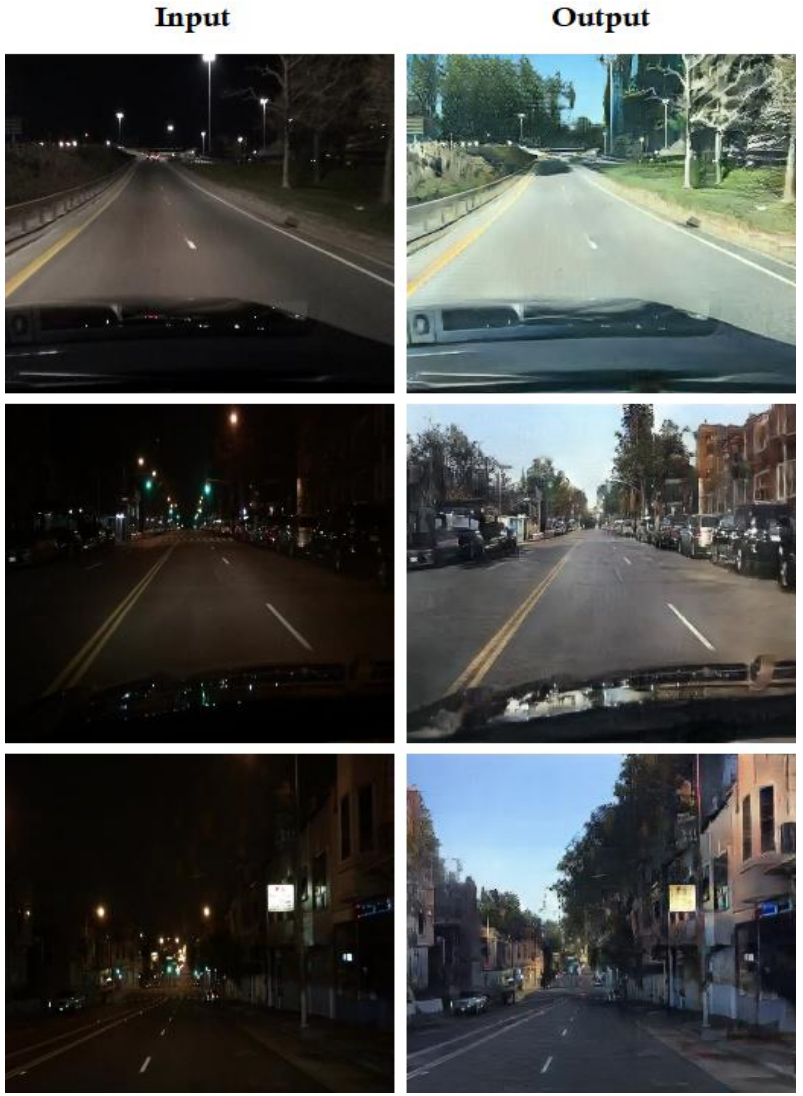


https://fullstackdeeplearning.com/spring2021/lecture-7/

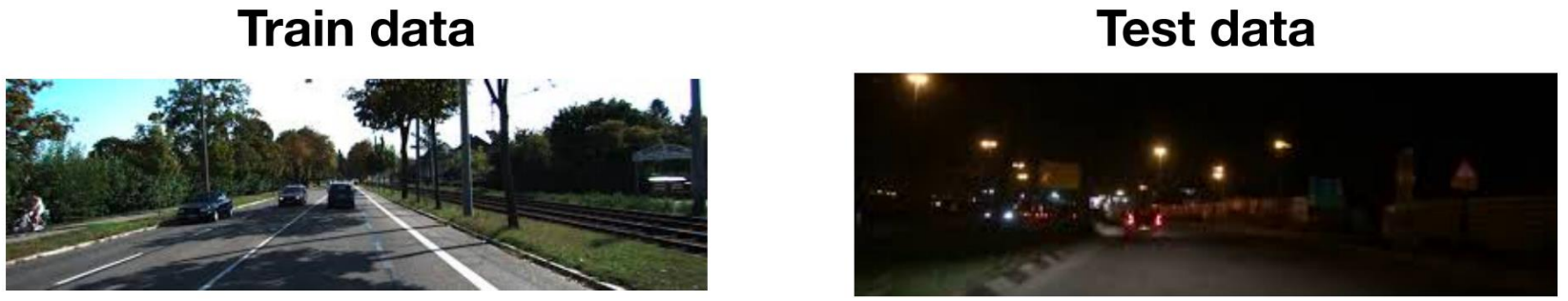
Training & Debugging- Improve model/data

- **학습데이터와 테스트데이터 분포 차이로 인한 에러 해결 방법**

1. Test-set 에러를 분석해 보고 학습데이터 추가 수집
2. Test-set 에러를 분석해 보고 학습데이터에 추가로 합성데이터를 넣어 줌.
3. Domain Adaptation기술을 적용



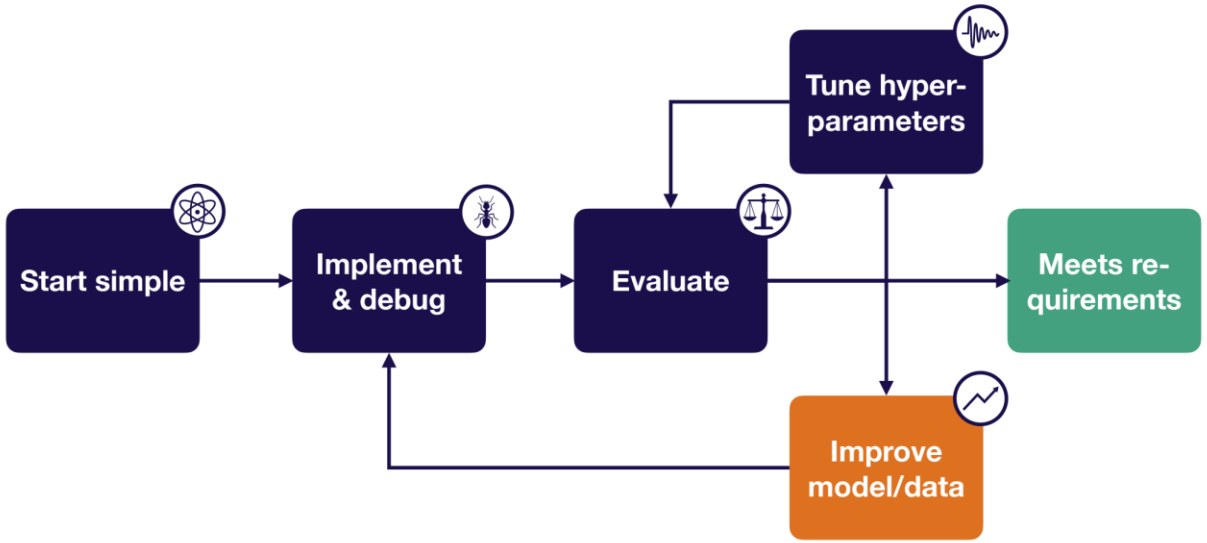
Domain Adaptation

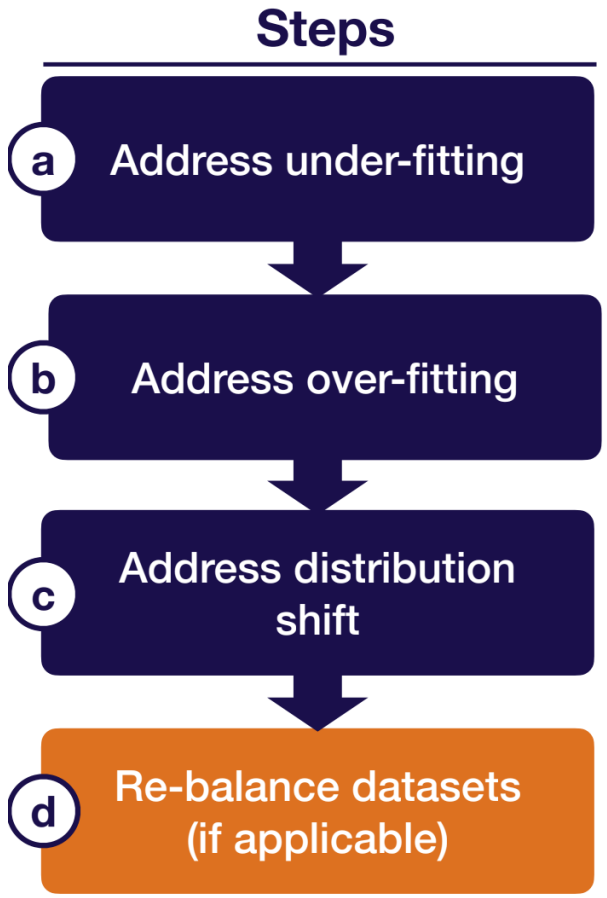


Distribution Shift

Training & Debugging- Improve model/data

- 딥러닝 문제해결을 위한 전략





https://fullstackdeeplearning.com/spring2021/lecture-7/

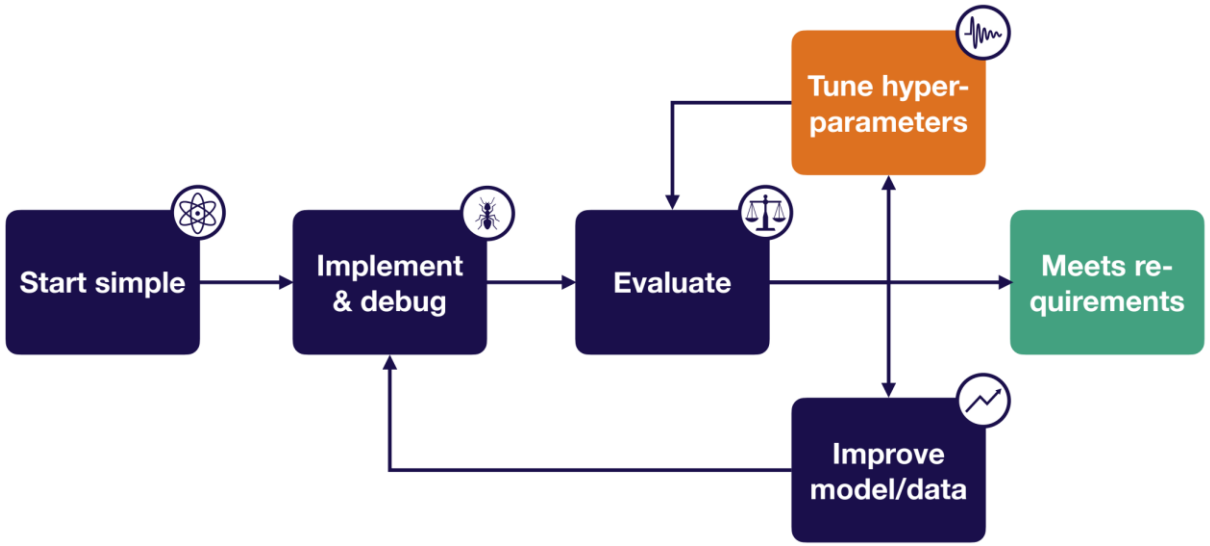
Training & Debugging- Improve model/data

- **Rebalance Datasets**
  - Test error보다 validation error가 상당히 작으면 validation set에 overfitting 된 것
  - 이것은 validation set이 상대적으로 작을 경우에 발생
  - Validation set을 다시 수집 (data 수를 증가 시킴)

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Tune Hyperparameters

- 딥러닝 문제해결을 위한 전략



Model & optimizer choices?

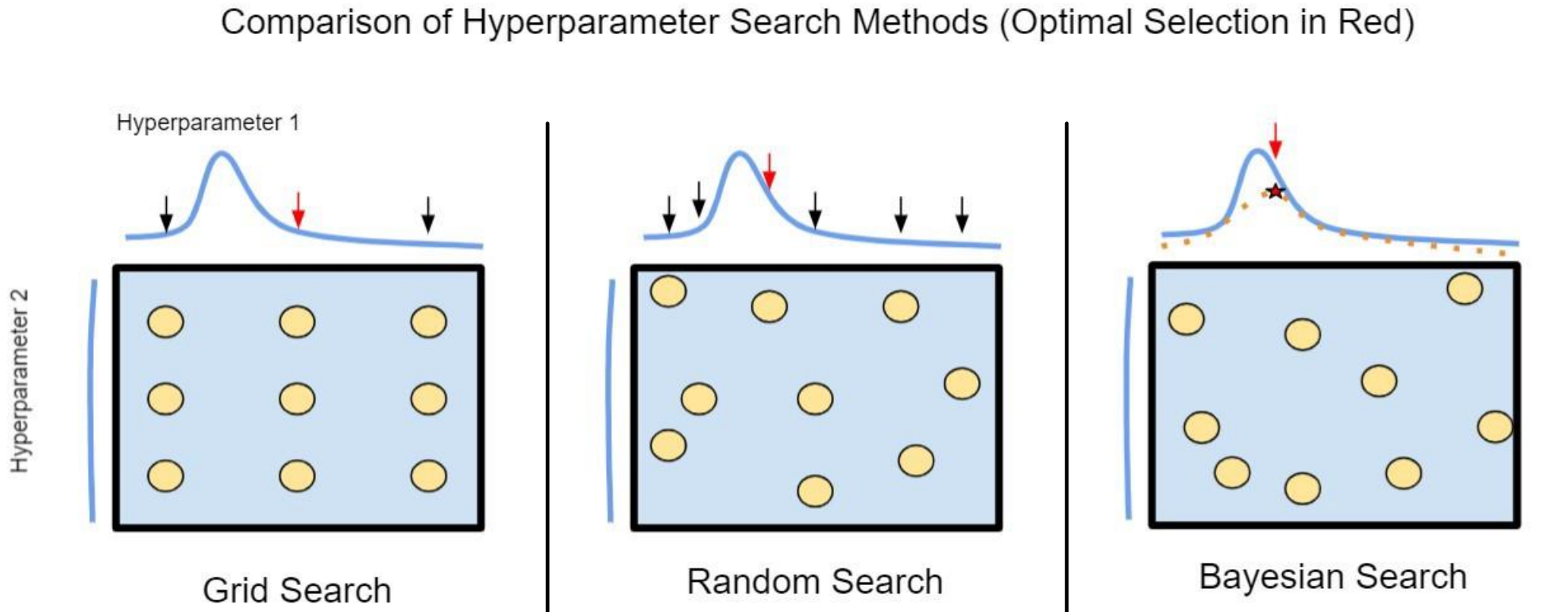
Network: ResNet
- How many layers? Layer 갯수
- Weight initialization? Weight 초기화
- Kernel size? 커널 크기 (3x3, 5x5,…, etc)
- Etc

Optimizer: Adam
- Batch size? Batch 크기
- Learning rate? 학습률
- beta1, beta2, epsilon?

Regularization
- ….

https://fullstackdeeplearning.com/spring2021/lecture-7/

Training & Debugging- Tune Hyperparameters



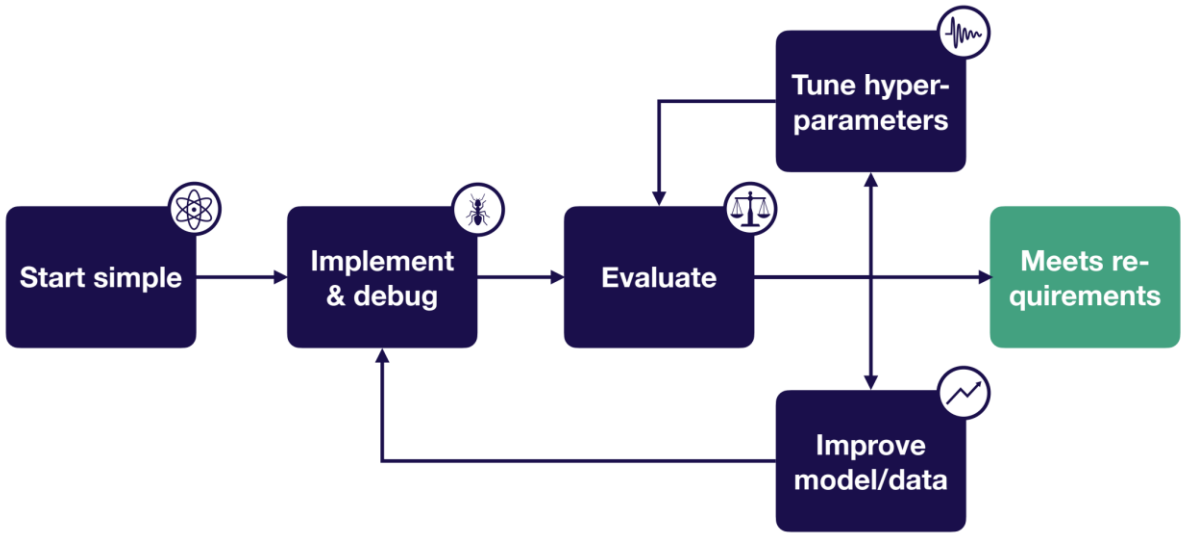
- 구현하기 쉽다.
- 많은 계산 필요, 비효율적
- 사전 지식이 있다면, 좋은 결과를 얻을 수 있음.

- 구현하기 쉽다.
- Grid Search보다 좋은 결과
- 해석이 어려움.
- 어느 정도의 사전지식 필요

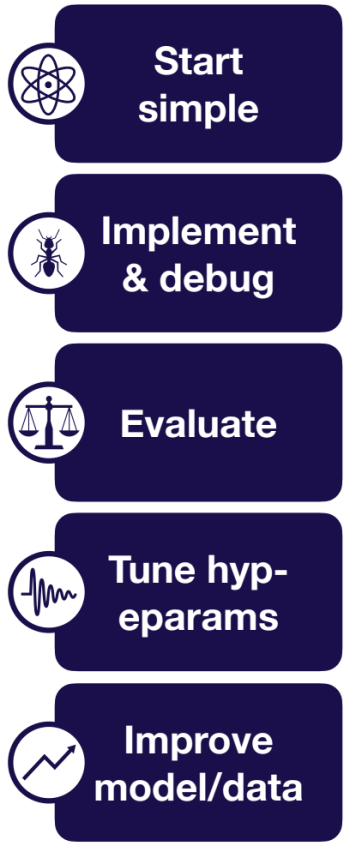
- 확률적 모델을 이용
- 가장 효율적인 방법
- 구현이 어려움 (오픈소스 있음)

https://towardsdatascience.com/hyperparameter-tuning-always-tune-your-models-7db7aeaf47e9

Training & Debugging - 결론



- 많은 에러의 원인들이 DL 디버깅을 어렵게 만든다.
- 버그없는 모델의 학습을 위해, 모델 구현을 반복적인 프로세스로 수행해야 한다.
- 이러한 과정들이 쉽고 빠르게 최대한 버그를 잡게 도와줄 것이다.



- **Choose the simplest model & data possible (e.g., LeNet on a subset of your data)**
- **Once model runs, overfit a single batch & reproduce a known result**
- **Apply the bias-variance decomposition to decide what to do next**
- **Use coarse-to-fine random searches**
- **Make your model bigger if you underfit; add data or regularize if you overfit**

Deploying & Monitoring

Deploying & Monitoring

- **모델을 배포하려면** 사용자에게 영향을 미칠 수 있는 **실패 사례**를 자세히 분석해야 함.
  1. 모델 배포 시 **데이터** 수집과 사용 시 고려할 점
  2. 모델 배포 시 **모델링** 시 고려할 점

Deploying & Monitoring

- **모델 배포 시 데이터 수집과 사용 시 고려 사항**

1. 데이터 소유권

- 사용자로부터 데이터 수집에는 도덕적, 법적 책임 수반, 데이터 수집 정책을 명확히 정의하고 사용자에게 공유
- **데이터 수집:** 모델을 훈련하기 위해 필요한 데이터셋을 수집하고 사용할 법적 권한을 가지고 있나요?
- **데이터 사용 및 허락:** 사용자의 데이터가 필요한 이유와 사용 방법을 명확하게 설명했나요? 사용자가 이에 동의했나요?
- **데이터 저장:** 데이터를 어떻게 저장하나요? 누가 데이터에 접근할 수 있나요? 언제 데이터를 삭제하나요?

Deploying & Monitoring

- **모델 배포 시 데이터 수집과 사용 시 고려 사항**

2. 데이터 편향

- 데이터 수집이 편향되어 머신러닝 모델도 이런 편향이 재현됨.
- 모든 데이터 셋은 편향되어 있다고 가정하고, 이 편향이 모델에 얼마나 영향을 미칠지 추정 필요.
- **측정 오류 또는 오염된 데이터:** 각 데이터는 불확실성이 있는데 이를 무시하여 측정 오류가 전파된다.
- **대표성:** 대부분의 데이터셋은 모집단을 완전히 대표하지 못한다.
- **접근성:** 일부 데이터셋은 다른 데이터셋에 비해 구하기 어려울 수 있다.

3. 시스템 편향

- 일부 집단을 부당하게 차별하도록 만드는 제도적, 구조적 정책에 의해 편향이 생길 수 있다.

Deploying & Monitoring

모델 배포 시 모델링 고려 사항

1. 피드백 루프 (추천시스템)

- 유저가 클릭한 영상은 추천시스템의 결과로 인한 것이고, 추천 시스템은 유저의 클릭을 기반으로 모델을 다시 학습
- 이로 인해 원치 않는 방향으로 편향이 생길 수 있음.

2. 포괄적인 모델 성능

- 기존 모델의 새 버전의 배포 여부 결정 시 요약된 성능 지표로만 비교하면, 일부 데이터에서의 성능 감소를 감지하지 못함.
- 예상 가능한 다양한 케이스에 대한 성능 검증이 필요.

3. 모델에 대한 정보

- 사용자에게 모델에 대한 정보를 제공함으로써 사용자가 이 정보를 어떻게 활용할 지 결정을 내리는데 도움을 주어야 함.
- **모델에 대한 정보:** 모델, 훈련방법, 테스트 데이터, 예상 사용 방법, 배포 시 모델이 잘 동작할 것으로 기대되는 입력 종류 등.

Deploying & Monitoring

- **모델 배포 시 모델링 고려 사항**

4. 적대 공격

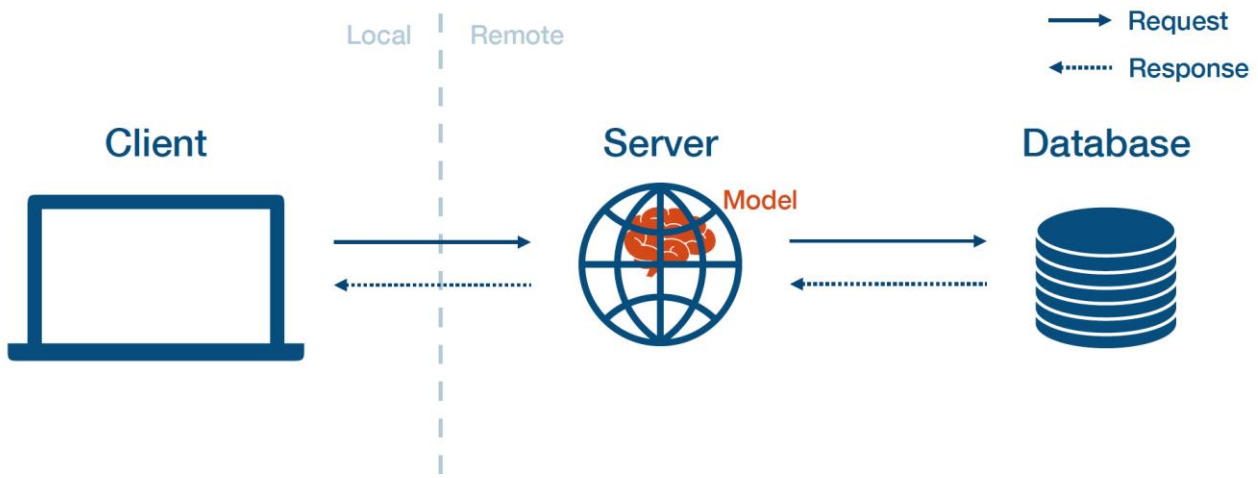
- 모델을 속이거나 교란 시키려는 시도 (adversarial attack)를 고려해서 모델링 해야 함.
- 적대 공격을 막기 위해 정기적으로 모델 업데이트
- 새로운 행동 패턴 감지를 위한 모니터링 시스템 필요

5. 이중 사용

- 하나의 목적으로 개발된 기술이 다른 목적으로 사용되는 것
- 얼굴인식 기술이 감시 기술로 사용
- 목소리/얼굴 합성 기술 (deep fake)이 정치적으로 악용되거나 타인 사칭

Deploying & Monitoring- Deployment (배포)의 형태

- **서버측 배포-** 스트리밍 애플리케이션 (API)
  - 웹서버에 배포 모델을 포함하여 실시간으로 예측함.
  - 속도에 대한 요구사항이 높을 때 필요.

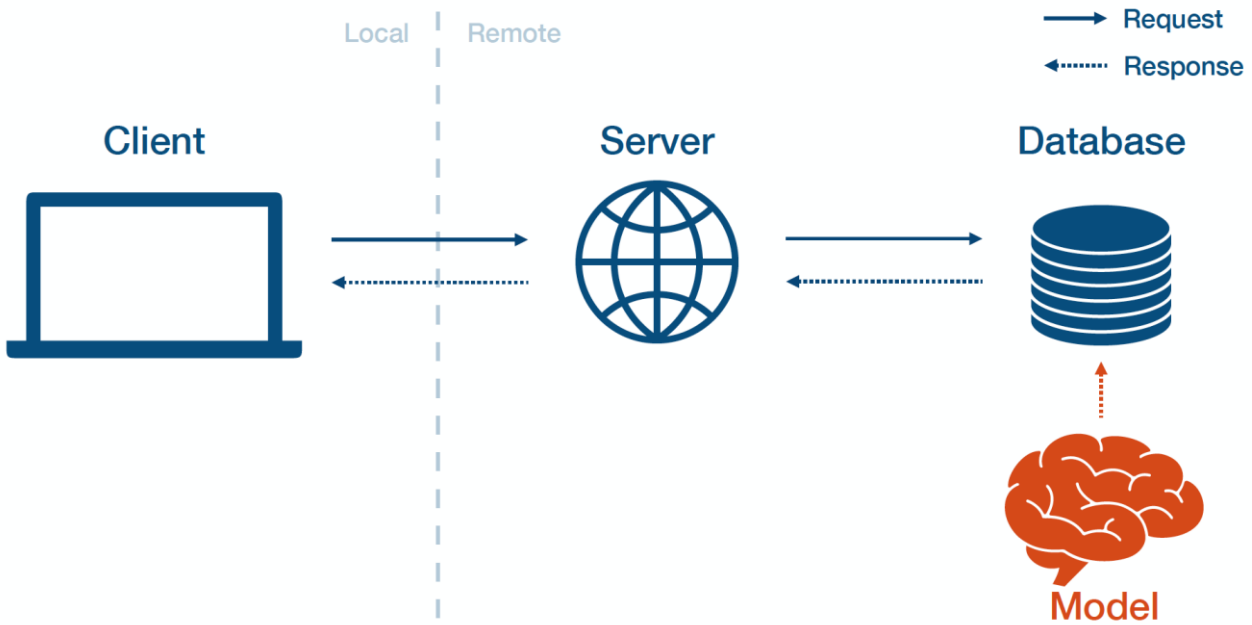


- **장점**
  - 기존의 인프라를 재사용하여 구축 가능.
- **단점**
  - 동시 사용자의 수가 늘어남에 따라 적절히 인프라를 늘려야 함.
  - 모델과 서버의 확장 정도가 달라질 수 있음.
  - 서버의 HW 가 모델에 최적화 되지 않을 수 있음 (서버에 GPU가 없을 때 등)

https://fullstackdeeplearning.com/spring2021/lecture-11/

Deploying & Monitoring- Deployment (배포)의 형태

- **서버측 배포-** 배치 예측 (Batch Prediction)
  - 주기적으로 새로운 데이터에 대해 모델 실행하여 결과 저장
  - 상대적으로 적은 입력 범위에 적합함 (1일에 1사용자당 1 예측)

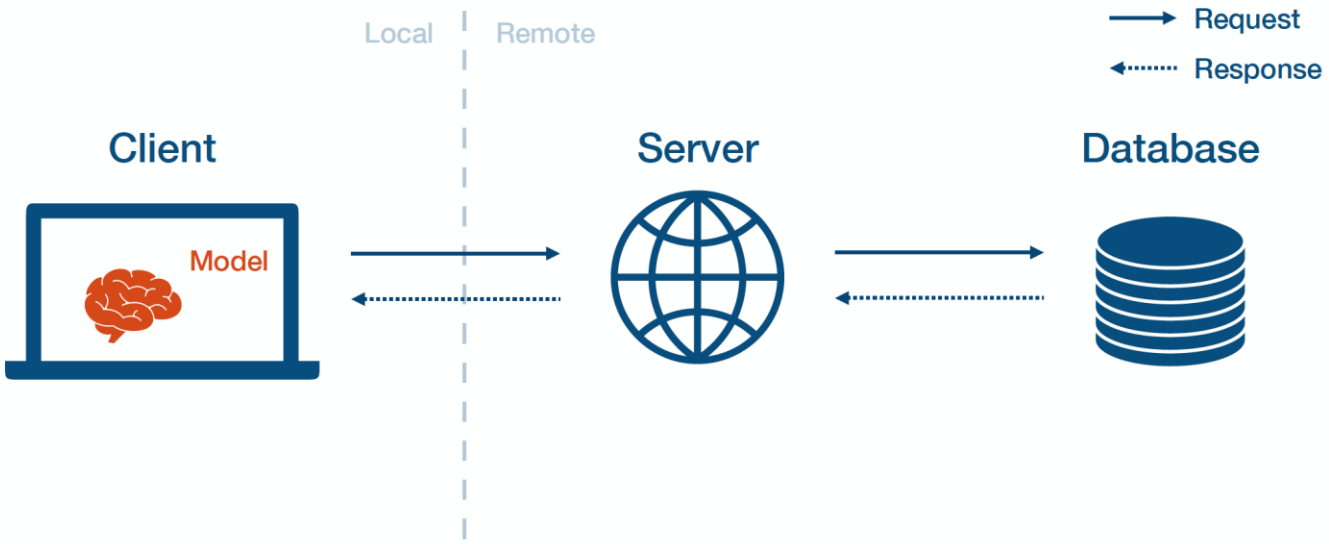


- **장점**
  - 구현이 쉬움.
  - 지정된 시간에 예측 수행하고 예측횟수를 알기 때문에 자원할당과 병렬화가 수월.
  - 추론 속도가 빠름(미리 계산하여 저장).
- **단점**
  - 사용자가 최신 예측 결과를 얻지 못함.

https://fullstackdeeplearning.com/spring2021/lecture-11/

Deploying & Monitoring- Deployment (배포)의 형태

- **클라이언트 측 배포–** 온 디바이스(On Device)
  - 장치에서 직접 모델 예측 수행

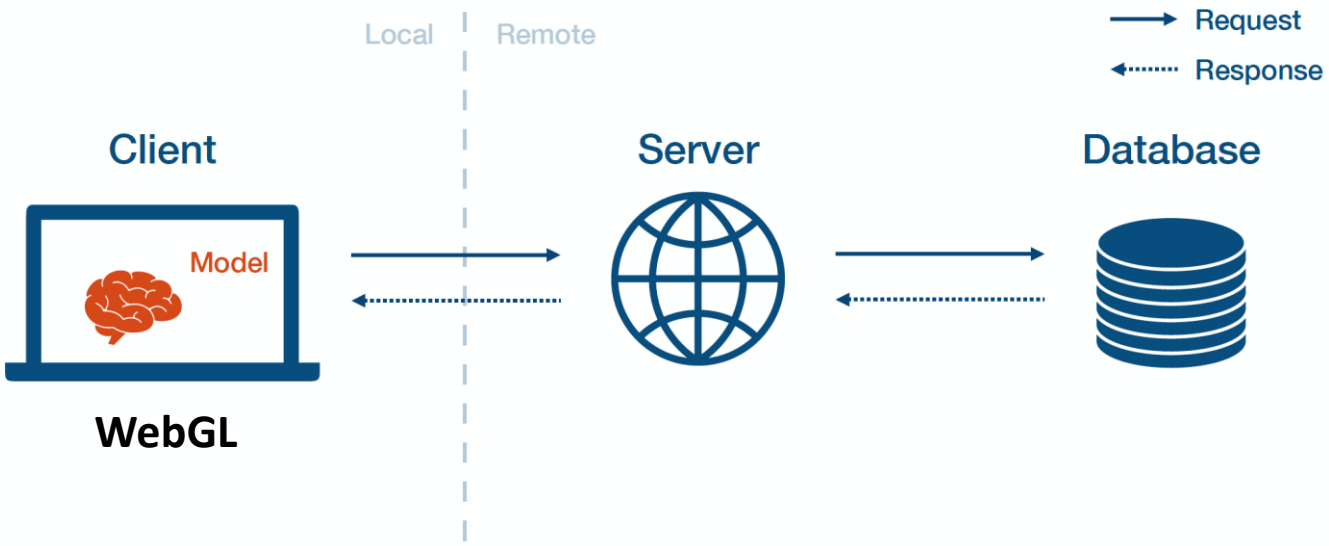


- **장점**
  - 대기시간 짧음.
  - 데이터 보안에 강점.
  - 네트워크가 없이 실행 가능함.
- **단점**
  - 모델의 예측 시간이 서버에서 보다 길어 짐.
  -
  - HW를 고려해 모델 경량화 작업이 필요함.
  - 모델 최적화에 시간과 노력이 많이 듦.
  - 예측 정확도의 손실을 가져올 수 있음.

https://fullstackdeeplearning.com/spring2021/lecture-11/

Deploying & Monitoring- Deployment (배포)의 형태

- **클라이언트 측 배포– 브라우저(WebGL)**
  - 스마트 장치의 브라우저의 그래픽 가속기를 사용해 모델 실행



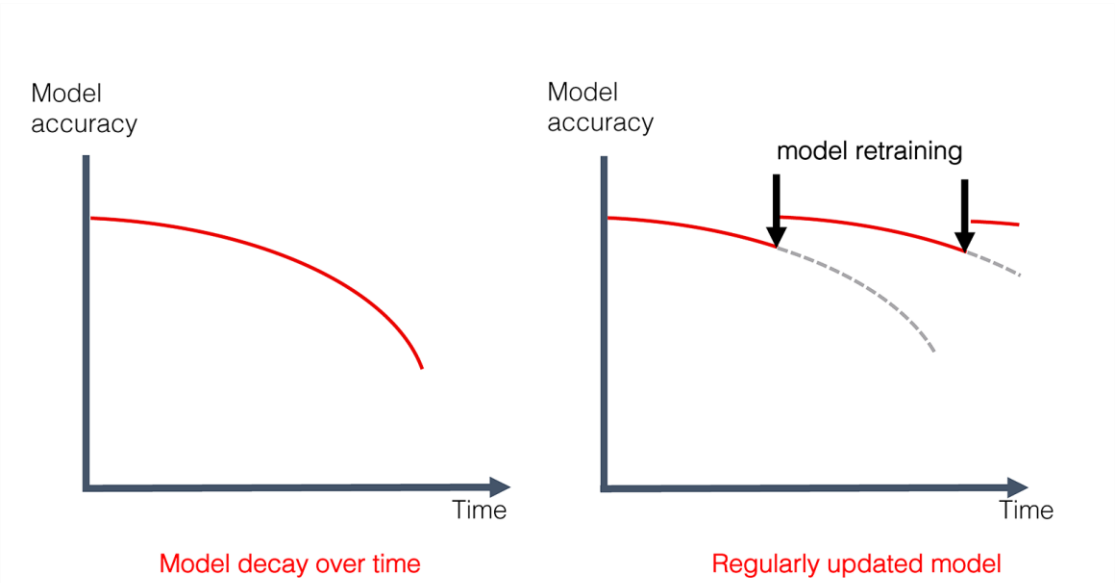
- **장점**
  - 추가적인 애플리케이션 설치 없이 사용 가능.
  - 브라우저를 사용해 머신러닝 작업을 수행하는 라이브러리가 많아지고 있음. (ex. TensorFlow.js)
- **단점**
  - 자바스크립트 기반의 언어에 익숙해야 함.
  - 라이브러리에서 호환되지 않는 함수 존재함.

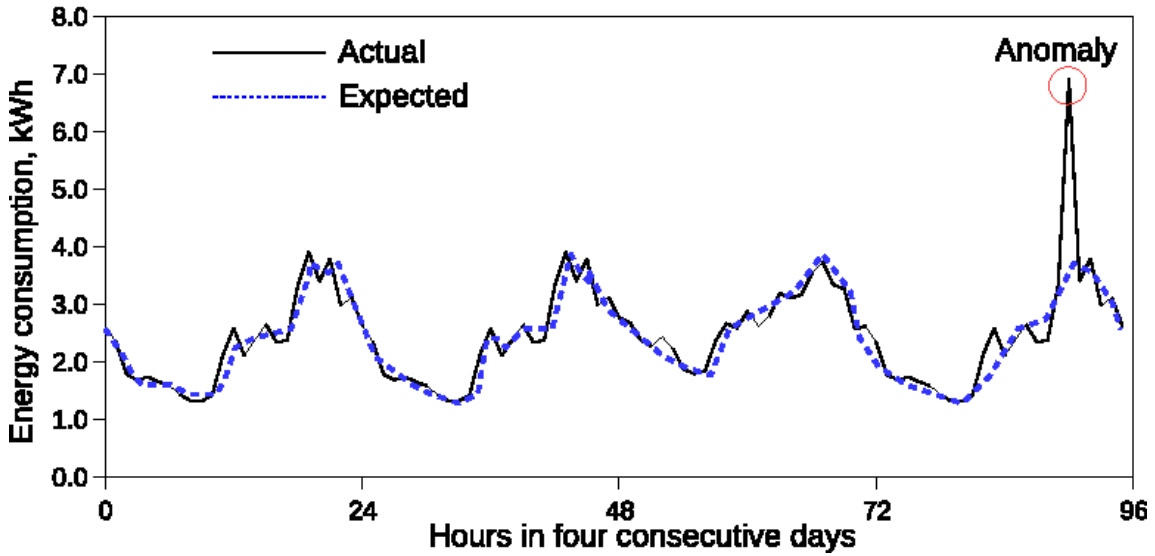
https://fullstackdeeplearning.com/spring2021/lecture-11/

Deploying & Monitoring- Monitoring

- **모니터링의 목적**
  - 모델 배포 후에 시스템의 상태를 추적하고, 모델의 성능과 예측 품질의 지속적으로 관리하기 위함.

- **모니터링의 역할**
  - 재훈련의 시기를 알려주는 모니터링 (ex. 사용자의 피드백으로 부터 정확도를 측정하여 하락 시 알람)
  - 이상치를 감지하는 모니터링 (ex. 시스템에 대한 공격을 감지, 갑자기 로그인 시도 횟수가 급격히 증가)





https://evidentlyai.com/blog/machine-learning-monitoring-data-and-concept-drift

Deploying & Monitoring- Monitoring

◆ 모델 성능 저하 시점을 감지하는 모니터링 지표

▪ 성능지표

1. **Data Drift**
   - 입력 데이터의 분포가 변하는 경우
   - 데이터 파이프라인의 버그를 통해 입력데이터가 잘 못 입력되는 경우
   - 악의적인 사용자에 의한 이상 입력

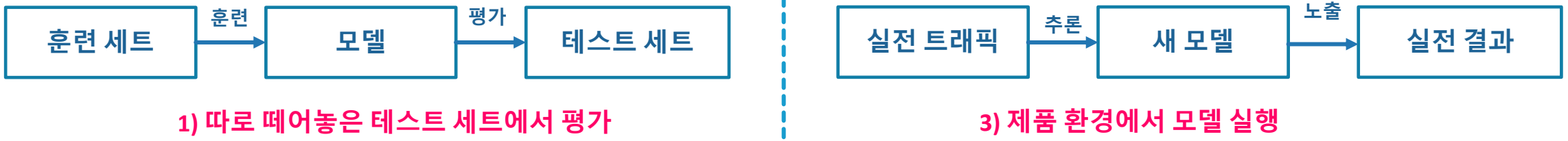
2. **Concept Drift**
   - 시간이 지남에 따라 모델에 대한 입력과 출력 사이의 통계적 특성이 변하는 경우
   - COVID 전후의 소비 패턴의 변화로 인한 예측 성능의 저하

▪ 비즈니스 지표

- 성능 지표가 정상이더라도, 비즈니스 지표가 떨어지면 제품측면에서 실패
- 검색이나 추천 시스템의 경우 모델 추천에 대한 클릭 비율인 CTR (Clickthrough Rate)을 비즈니스 지표로 활용

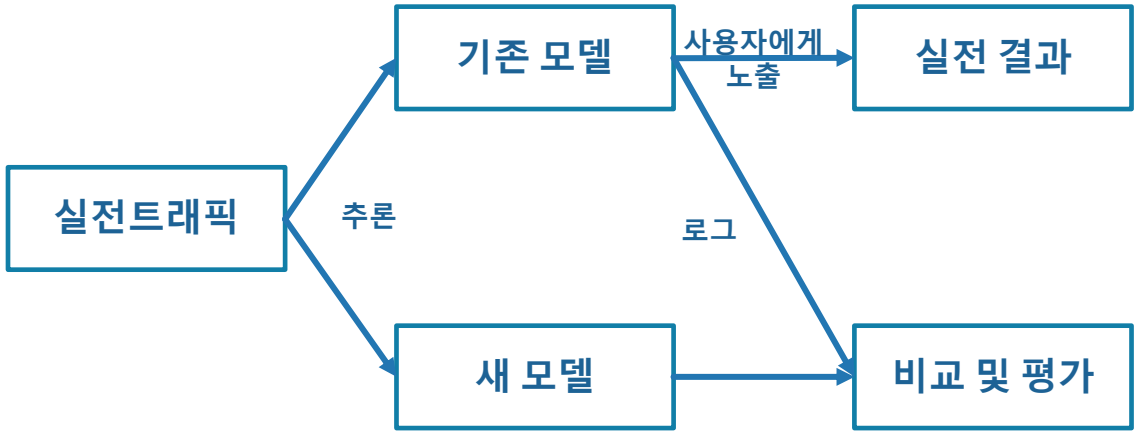
Deploying & Monitoring- 머신러닝을 위한 CI/CD

- **모델 평가 방법**



1) 따로 떼어놓은 테스트 세트에서 평가

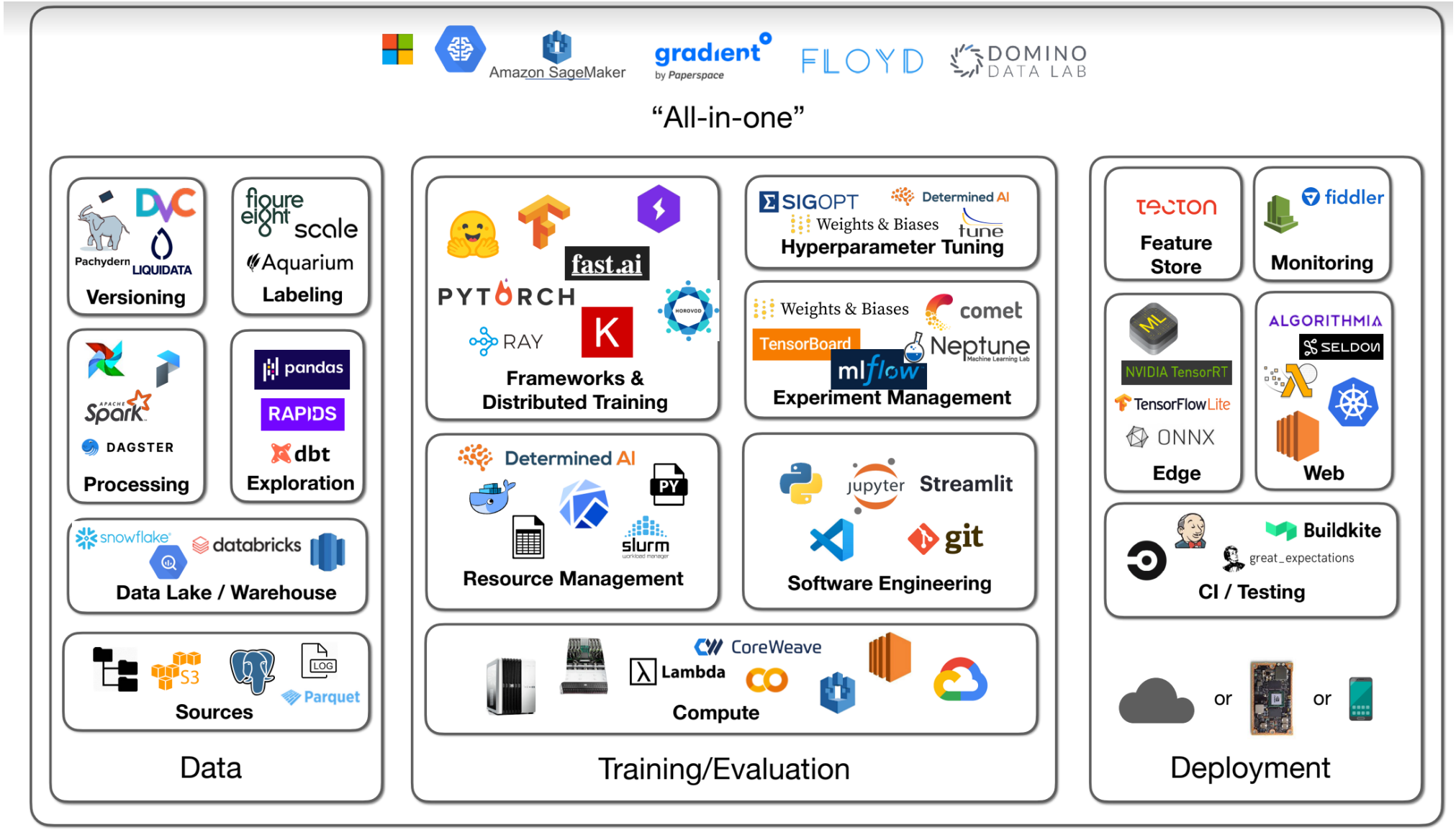
3) 제품 환경에서 모델 실행



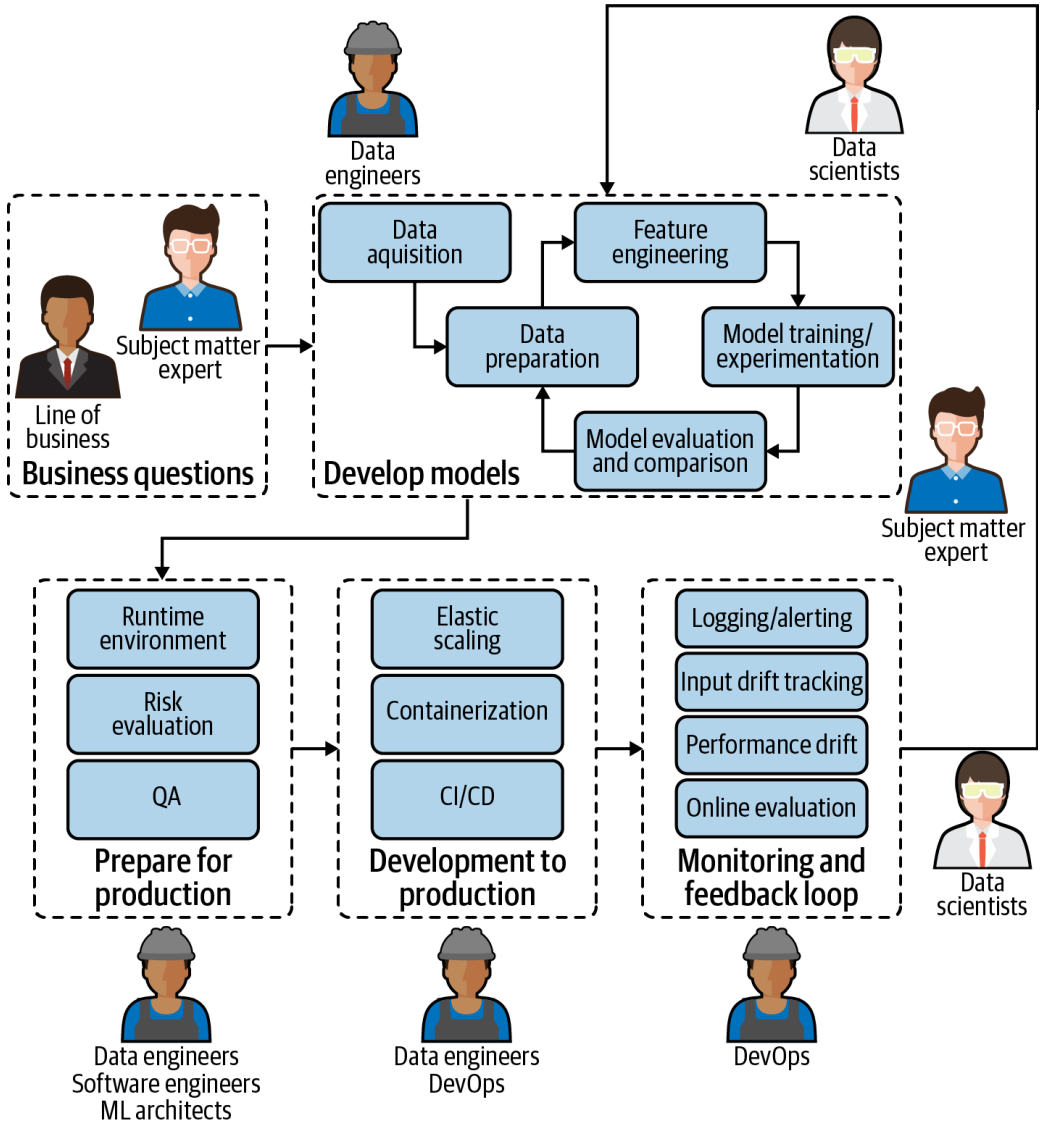
2) 섀도 모드에서 모델 실행

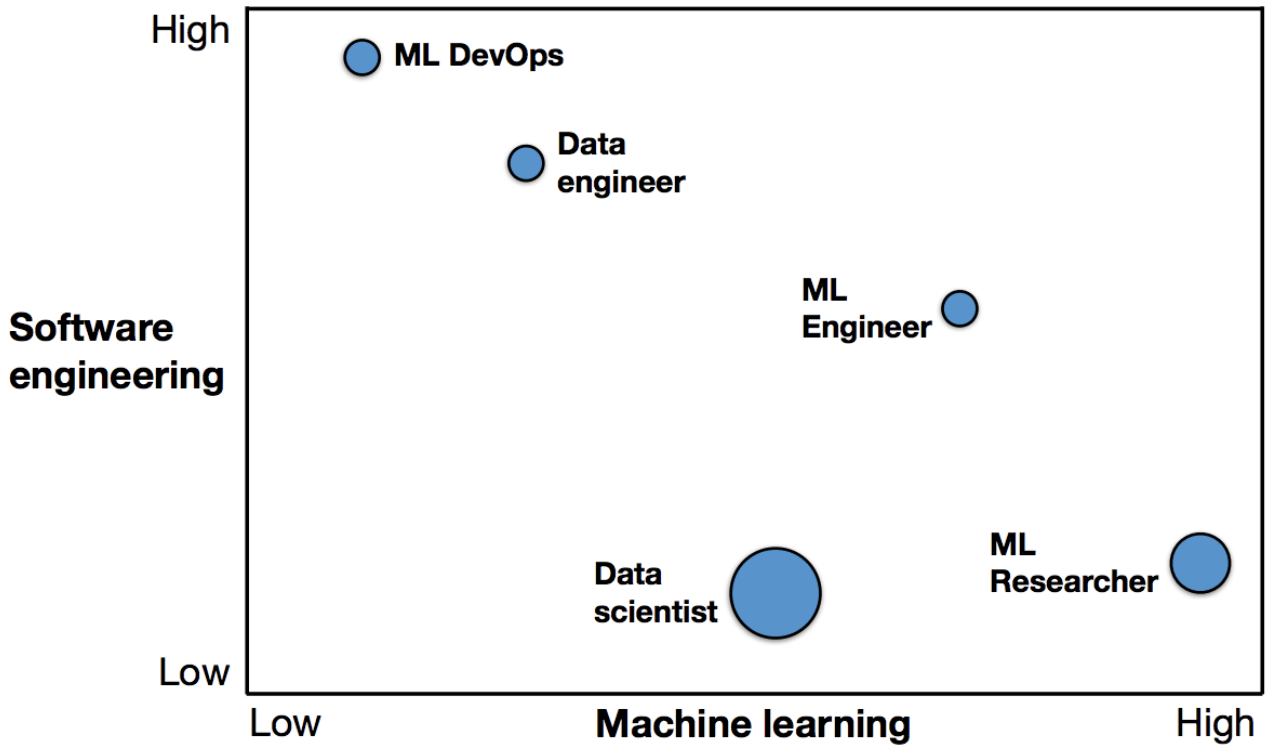
평가 정확도 & 평가 위험

MLOps Infrastructure & Tooling



AI 프로젝트의 개발팀 조직 구성 (1/4)





AI 프로젝트의 팀 조직 구성 (2/4)

- **Subject Matter Experts (SME)**
  - 해당직무 또는 과제를(지식, 기능, 태도 측면에서) **가장 잘 알고, 잘 수행하고 있는 사람**
  - ML 서비스에서 SME의 역할은 모델을 구성해야 하는 **비즈니스 목표 또는 KPI(핵심성과지표)를 제공**
  - **지속적으로 서비스를 평가**하고 **모델의 성능이 요구 사항과 일치하는지 확인**
  - MLOps 프로세스를 구축할 때 **모델의 성능을 비즈니스 관점에서 쉽게 이해할 수 있는 방법을 확보**

AI 프로젝트의 팀 조직 구성 (3/4)

- **데이터 어노테이션/퀄리티 매니저 (Data Annotation/Quality Manager)**
  - 데이터 어노테이션 계획(고객 요구 spec파악, 일정 계획 수립, 어노테이션 가이드라인 작성 및 협의)
  - 어노테이션 인력 채용 및 가이드라인 교육
  - 어노테이션 작업자 모니터링 및 진행률 관리

- **데이터 엔지니어 (Data Engineer)**
  - 데이터를 수집하거나, 방대한 데이터를 다루는 인프라를 구축하고 운영하는 역할을 담당

- **데이터 사이언티스트 (Data Scientist) / 머신러닝 연구자 (ML Research)**
  - 데이터 모델링을 통해 다양한 문제해결의 방법론을 개발
  - 머신러닝 모델 개발을 위한 이론적 개념을 탐구하고 증명

AI 프로젝트의 개발팀 조직 구성 (4/4)

- **머신러닝 엔지니어 (ML Engineer)**
  - 머신러닝 기반의 제품 및 서비스를 구현하거나 기업이 지닌 문제를 해결할 AI 솔루션을 구현
  - 컴퓨터공학, 수학 등에 기반하여 주요 AI분야에 대한 이해를 충분히 해야 한다.
  - 머신러닝 연구자와 데이터 사이언티스트가 개발한 머신러닝 모델을 실 서비스에 배포 가능한 규모의 시스템으로 구축
  - 모델 학습 및 테스트 자동화, 스케일링, 모델 배포 및 모니터링 업무

- **애플리케이션 개발자 (Software Engineer) / DevOps팀**
  - 최종 배포 모델을 개발중인 비즈니스 애플리케이션에 통합
  - ML모델의 보안, 성능테스트, 운영 시스템 구축 및 수행
  - CI/CD 파이프라인 구축 및 관리

- **인프라스트럭쳐 운영 팀 (MLOps Team)**
  - 모델 개발부터 트레이닝, 최적화, 프로덕션 환경에 배포하는 일련의 과정을 모두 지원
  - AI프로젝트 팀의 생산성에 직접적인 영향을 끼침

AI 인재 확보를 위한 지침

- **AI를 활용하는 목적이 분명해야 한다.**
  - AI를 활용하는 목적이 분명해야 이를 잘 수행하는 인재가 어떤 인재인지 알 수 있다.
  - 고려해야 할 변수가 매우 많기 때문에 목적 지향적인 인재 확보가 필요하다.

- **AI 인재 확보에는 균형을 유지해야 한다.**
  - AI 기술 구현만으로는 혁신이 완성되지 않는다.
  - 좋은 AI 제품을 만들어 놓아도 시장에서 주목도 못 받고 사장되는 일이 부지기수다.
  - 시장과 비즈니스를 아는 인력과 AI 기술인력의 적절한 균형점을 찾아야 한다.

- **AI 인재 확보에는 유연한 자세를 견지할 필요가 있다.**
  - AI는 학습과 시행착오가 필요하다.
  - AI는 만능이 아니기 때문에 제대로 된 솔루션이 나오기까지 많은 실패 과정이 필요하다.
  - AI는 학습에 기반하기 때문에 시스템이 성장하기 까지 인내심이 필요하다.
  - 이러한 인내심은 인재를 채용하는 경영자의 AI에 대한 충분한 이해가 있어야 가능하다.

https://www.technologyreview.kr/ai-talents-recruit/

Q & A