KSBi-BIML 2023

Bioinformatics & Machine Learning(BIML)
Workshop for Life Scientists, Data Scientists,
and Bioinformatians

생물정보학&머신러닝 워크샵(오프라인)

Introduction to Deep Learning

이상근 _ 고려대학교

SBi 한국생명정보학회
Korean Society for Bioinformatics

본 강의 자료는 한국생명정보학회가 주관하는 BIML 2023 워크샵 오프라인 수업을 목적으로 제작된 것으로 해당 목적 이외의 다른 용도로 사용할 수 없음을 분명하게 알립니다.

이를 다른 사람과 공유하거나 복제, 배포, 전송할 수 없으며 만약 이러한 사항을 위반할 경우 발생하는 **모든 법적 책임은 전적으로 불법 행위자 본인에게 있음을 경고**합니다.

KSBi-BIML 2023

Bioinformatics & Machine Learning (BIML)
Workshop for Life Scientists, Data Scientists, and Bioinformatians

안녕하십니까?

한국생명정보학회가 개최하는 동계 교육 워크샵인 BIML-2023에 여러분을 초대합니다. 생명정보학 분야의 연구자들에게 최신 동향의 데이터 분석기술을 이론과 실습을 겸비해 전달하고자 도입한 전문 교육 프로그램인 BIML 워크샵은 2015년에 시작하여 올해로 9차를 맞이하게 되었습니다. 지난 2년간은 심각한 코로나 대유행으로 인해 아쉽게도 모든 강의가 온라인으로 진행되어 현장 강의에서만 가능한 강의자와 수강생 사이에 다양한 소통의 기회가 없음에 대한 아쉬움이 있었습니다. 다행히도 최근 사회적 거리두기 완화로 현장 강의가 가능해져 올해는 현장 강의를 재개함으로써 온라인과 현장 강의의 장점을 모두 갖춘 프로그램을 구성할 수 있게 되었습니다.

BIML 워크샵은 전통적으로 크게 인공지능과 생명정보분석 두 개의 분야로 구성되었습니다. 올해 AI 분야에서는 최근 생명정보 분석에서도 응용이 확대되고 있는 다양한 심층학습(Deep learning) 기법들에 대한 현장 강의가 진행될 예정이며, 관련하여 심층학습을 이용한 단백질구조예측, 유전체 분석, 신약개발에 대한 이론과 실습 강의가 함께 제공될 예정입니다. 또한 싱글셀오믹스 분석과 메타유전체분석 현장 강의는 많은 연구자의 연구 수월성 확보에 큰 도움을 줄 것으로 기대하고 있습니다. 이외에 다양한 생명정보학 분야에 대하여 30개 이상의 온라인 강좌가 개설되어 제공되며 온라인 강의의 한계를 극복하기 위해서 실시간 Q&A 세션 또한 마련했습니다. 특히 BIML은 각 분야 국내 최고 전문가들의 강의로 구성되어 해당 분야의 기초부터 최신 연구 동향까지 포함하는 수준 높은 내용의 강의가 될 것입니다.

이번 BIML-2023을 준비하기까지 너무나 많은 수고를 해주신 BIML-2023 운영위원회의 남진우, 우현구, 백대현, 정성원, 정인경, 장혜식, 박종은 교수님과 KOBIC 이병욱 박사님께 커다란 감사를 드립니다. 마지막으로 부족한 시간에도 불구하고 강의 부탁을 흔쾌히 허락하시고 훌륭한 현장 강의와 온라인 강의를 준비하시는데 노고를 아끼지 않으신 모든 연사분께 깊은 감사를 드립니다.

2023년 2월

한국생명정보학회장 이 인 석

강의 시간표

DAY1 (2.6 월)

시간	강 의 서울대 자연과학대학 26동B101호	강사	강 의 서울대 자연과학대학 26동B102호	강사
09:00-09:20 (20)	등록			
09:20-09:30 (10)	개회사/공지사항전달			
09:30-10:50 (80)	**Best practice for single-cell data analysis**	**박종은 교수**	**Introduction to ML & DNN (이론)**	**이상근 교수**
10:50-11:00 (10)	휴식			
11:00-12:10 (70)	**Practice1: Scanpy basic workflow**	**김우석 김성룡 조교**	**CNN (이론)**	**이상근 교수**
12:10-13:40 (90)	점심 (KOBIC 세미나)			
13:40-15:10 (90)	**Public data, batch correction, cell annotation**	**박종은 교수**	**RNN, GAN, XAI (이론)**	**이상근 교수**
15:10-15:20 (10)	휴식			
15:20-16:50 (90)	**Practice2: Advanced single-cell analysis**	**김우석 김성룡 조교**	**AI 모델 구조 정의, 학습 알고리즘 적용, 성능 평가, 시각화 방법 (Tensorflow 실습)**	**이정현 한성민 조교**

DAY2 (2.7 화)

시간	강 의 서울대 자연과학대학 26동B101호	강사	강 의 서울대 자연과학대학 26동B102호	강사
09:00-09:20 (20)	등록			
09:20-09:30 (10)	공지사항전달			
09:30-10:50 (80)	**Introduction to protein structure prediction - Homology modeling - Coevolution-guided modeling Early AI-based approaches**	**백민경 교수**	**Pre-trained Models for Transfer Learning (이론)**	**전민지 교수**
10:50-11:00 (10)	휴식			
11:00-12:10 (70)	**단백질 구조 예측 실습 - MSA generation, template search - homology modeling contact prediction & modeling**	**백민경 교수**	**Pre-trained Models for Transfer Learning (실습)**	**정민수 조교**
12:10-13:40 (90)	점심			
13:40-15:10 (90)	**AI-based protein structure prediction - AlphaFold/RoseTTAFold Applications to PPI prediction & protein design**	**백민경 교수**	**Deep learning in Bioinformatics**	**노미나 교수**
15:10-15:20 (10)	휴식			
15:20-16:50 (90)	**단백질 구조 예측 실습 II AlphaFold, RoseTTAFold 실습 및 응용**	**백민경 교수**	**Deep learning model을 이용한 실습**	**곽호진 박예솔 조교**

DAY3 (2.8 수)

시간	강 의 서울대 자연과학대학 26동B101호	강사	강 의 서울대 자연과학대학 26동B102호	강사
09:00-09:20 (20)	등록			
09:20-09:30 (10)	공지사항전달			
09:30-10:50 (80)	**화학정보학 기초(Cheminformatics) 약물특성 및 약물다움(druglikeness) Molecular Notations & Descriptors AI 신약개발을 위한 Databases AI 신약개발을 위한 Programming 기초**	**김동섭 교수**	**마이크로바이옴 기본 이론**	**이선재 교수**
10:50-11:00 (10)	휴식			
11:00-12:10 (70)	**Google Colab에 RDKit 설치 화합물 정보 읽기 실습 Bioactivity database 검색 및 정보 읽기 실습 Molecular descriptor (fingerprint) 생성 및 similarity 계산 실습**	**문채영 나민주 조교**	**16S rRNA amplicon seq. - DADA2**	**서영창 조준우 조교**
12:10-13:40 (90)	점심 (KOBIC 세미나)			
13:40-15:10 (90)	**AI 신약개발을 위한 기계학습법 기초 QSAR 모델링 기초 AI 신약개발을 위한 딥러닝 모델 Virtual screening (ligand-based, structure-based) 및 de novo design**	**김동섭 교수**	**최신 메타지놈 분석 기법의 현황**	**이선재 교수**
15:10-15:20 (10)	휴식			
15:20-16:50 (90)	**QSAR modeling 전체 과정 실습 화합물의 Bioactivity 예측 모델 개발 Virtual screening 과정을 통한 신약후보물질 발굴 실습**	**문채영 나민주 조교**	**Shotgun metagenome 분석 (Linux)**	**서영창 조준우 조교**

강의개요

Introduction to Deep Learning

딥러닝은 이미지/영상 처리, 시계열 예측 등 다양한 분야에서 다량의 데이터를 기반으로 분류 등 문제를 해결하기 위한 기계학습 기법이다. 본 과정에서는 기계학습과 딥러닝의 개념적 이해를 바탕으로, 최근 많이 활용되고 있는 CNN (Convolutional Neural Network), RNN (Recurrent Neural Network)의 구조와 활용 방법에 대해 소개한다. 또한 GAN 등 생성 모델과 최근 각광받고 있는 XAI (eXplainable AI) 기술에 대해 간단히 소개한다. 본 과정은 각 기법의 개념적 이해와 직관적인 수학적 이해를 통해 수강생이 각 기법의 동작 원리와 장단점에 대해 파악할 수 있도록 하며, 또한 구글의 딥러닝 소프트웨어인 Tensorflow를 이용한 실습을 통해 딥러닝 기법 적용을 위한 기초 소양을 다지고자 한다.

강의는 다음의 내용을 포함한다:

- 기계학습 및 딥러닝의 기초
- DNN (Deep Neural Network), CNN (Convolutional Neural Network), RNN (Recurrent Neural Network) 이해
- GAN 등 생성 기법 소개
- XAI 기법 소개

* 참고강의교재:

  Deep learning, Goodfellow, Bengio & Courville, MIT Press, 2016

* 교육생준비물:

  노트북 (메모리 8GB 이상, 디스크 여유공간 30GB 이상), 구글 크롬 웹 브라우저

  실습 시 구글 Colaboratory 사용 예정 (설치 필요 없음, 구글 개인 계정 생성 필수)

  https://colab.research.google.com/notebooks/welcome.ipynb

* 강의 난이도: 초급~중급

* 강의: 이상근 교수 (고려대학교 정보보호대학원)

Curriculum Vitae

Speaker Name: Sangkyun Lee, Ph.D.

▶ **Personal Info**

Name	Sangkyun Lee
Title	Associate professor
Affiliation	Korea University

▶ **Contact Information**

Address	145, Anam-ro, Seongbuk-gu, Seoul, 02841, Korea
Email	sangkyun@korea.ac.kr
Phone Number	02-3290-4890

Research Interest

Trustworthy AI, Robust deep learning methods, AI for security, Data analysis

Educational Experience

2003	B.S., Seoul National University
2005	M.S., Seoul National University
2011	Ph.D., University of Wisconsin-Madison, USA

Professional Experience

2011-2014	Post-doc Researcher, SFB 876, TU Dortmund University, Germany
2015-2017	Principal Investigator, SFB 876, TU Dortmund University, Germany
2017-2019	Assistant Professor, Department of Computer Science, Hanyang University ERICA
2020-2021	Assistant Professor, School of Cybersecurity, Korea University
2022-current	Associate Professor, School of Cybersecurity, Korea University

Selected Publications (5 maximum)

1. Libra-CAM: An Activation-Based Attribution Based on the Linear Approximation of Deep Neural Nets and Threshold Calibration, Sangkyun Lee, Sungmin Han, IJCAI, 2022
2. Model Stealing Defense against Exploiting Information Leak Through the Interpretation of Deep Neural Nets, Jeonghyun Lee, Sungmin Han, Sangkyun Lee, IJCAI, 2022
3. Hunt for Unseen Intrusion: Multi-Head Self-Attention Neural Detector, Seongyun Seo, Sungmin Han, Janghyeon Park, Shinwoo Shim, Han-Eul Ryu, Byoungmo Cho, and Sangkyun Lee, IEEE Access, 2021
4. Fast Saddle-Point Algorithm for Generalized Dantzig Selector and FDR Control with the Ordered l1-Norm, Sangkyun Lee, Damian Brzyski and Malgorzata Bogdan, AISTATS, 2016
5. Co-author, Mutational dynamics between primary and relapse neuroblastomas. In Nature Genetics, Vol. 47, No. 8, pages 872-877, 2015.

Introduction to Deep Learning

Introduction to ML & DNN

고려대학교 정보보호대학원 인공지능연구실 이상근

KSBi-BIML 2023

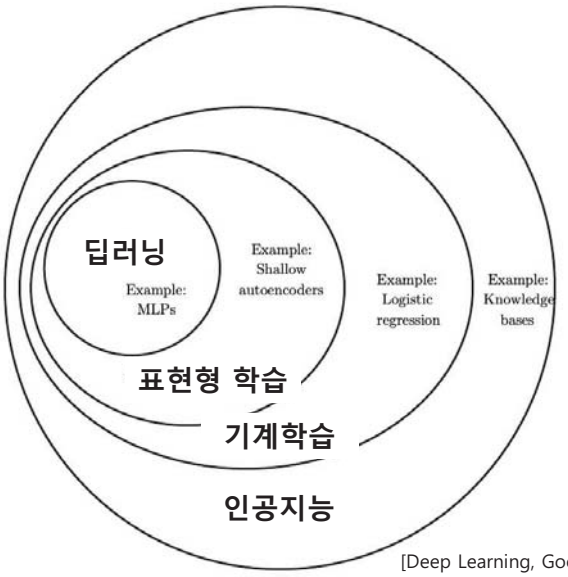
Machine Learning

- Arthur Lee Samuel (1901~1990, 1959)
  - A pioneer in AI
  - AI: a field of study that gives computers the ability to learn without being explicitly programmed

- Vladimir Vapnik (1936~)
  - The father of ML
  - Statistical Learning Theory (Wiley, 1998)

기계학습

AI: 학습이나 문제해결 등, 인간의 인지와 관련된 기능을 모사하는 SW/HW



[Deep Learning, Goodfellow et al., 2016]

인공지능 AI
: 전문가 시스템, Cybernetics

기계학습 Machine Learning
: SVM, 로지스틱 회귀

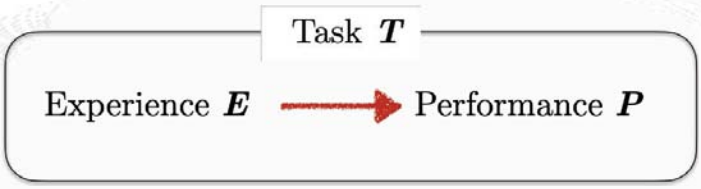
표현형 학습 Representation Learning
: Autoencoder

딥러닝 Deep Learning
: 자연어 처리, computer vision, …

Machine Learning

- Tom Mitchell (1997)

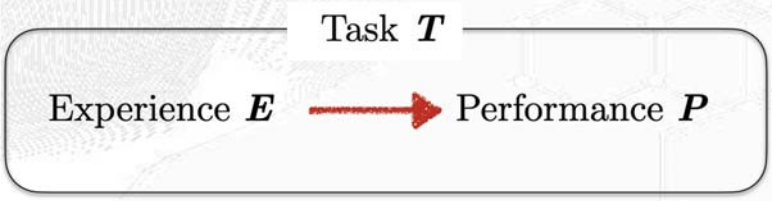
  "A computer program is said to learn from experience *E* w.r.t. some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."



Side: AGI (Artificial General Intelligence)

- No limitation on the task T

Machine Learning



Task T
- Classification
- Regression
- Machine translation
- Outlier detection
- Synthesis

.
.
.

Experience E
- Supervised Learning
- Unsupervised Learning
- Semi-Supervised
- Reinforcement Learning

.
.
.

Performance P
- Training Error
- Test Error
- Generalization Error

.
.
.

Data

- Supervised Learning: X (input), Y (output)
- Unsupervised Learning: X (input), no Y
- Semi-supervised Learning: (X1,Y1) and X2
- Self-supervised Learning: X → (X', Y')

In Statistical Learning…

Sampling (i.i.d)

i : independent
i.d.: identical distribution

Data Set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$

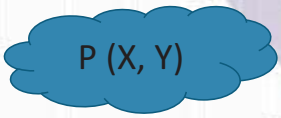
A set consists of m samples

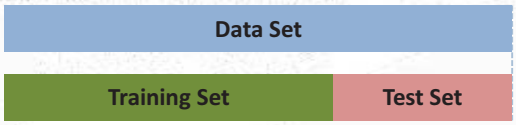
P (X, Y)

Probability distribution
(usually, unknown)

Note:
- There can be many possible data sets
- We're dealing with just one set of samples

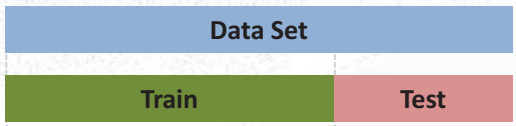
Performance





- **Training Error (Rate)** : error on the training set $\frac{1}{|\text{tr}|}\sum_{i\in\text{tr}}\mathbf{1}[y^{(i)} \neq f_w(x^{(i)})]$
- **Test Error (Rate)** : error on the test set $\frac{1}{|\text{tt}|}\sum_{i\in\text{tt}}\mathbf{1}[y^{(i)} \neq f_w(x^{(i)})]$
- **Generalization Error**: error on the **all possible** data $\mathbb{E}_{(X,Y)}\left[\mathbf{1}[Y \neq f_w(X)]\right]$

Model Selection



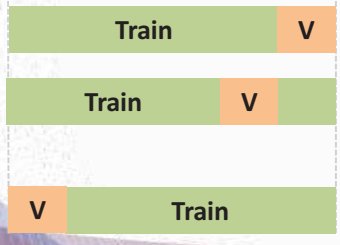
Model selection is a part of hyper-parameter tuning

- Hold-out method

Model A ⟷ Model B

Perf A ⟷ Perf B

- K-Fold CV (cross validation)

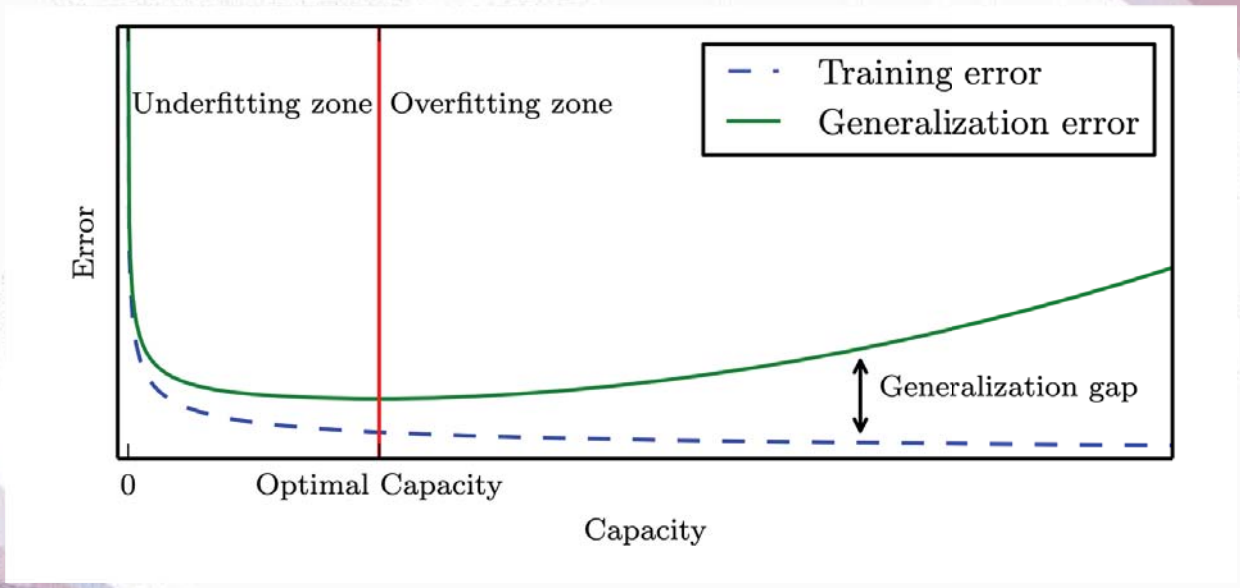


Perf A Fold 1 ⋮ Perf A Fold K

Perf B Fold 1 ⋮ Perf B Fold K

Avg Perf A ⟷ Avg Perf B

Generalization & Capacity



Statistical Learning

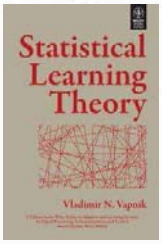
A dataset consists of
n samples (x, y)

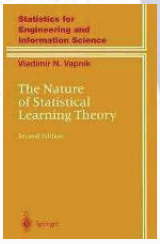
(Input, Label) Space

Unknown Probability Distribution: D(X,Y)

Predictor (Hypothesis) $h : X \to Y$ $h \in \mathcal{H}$ Hypothesis space

Loss $\ell_h : X \times Y \to \mathbb{R}$ e.g. set of linear functions, etc.





Risk and Empirical Risk

A dataset consists of n samples (x, y)

Predictor (Hypothesis) $h : X \to Y, h \in \mathcal{H}$

Loss $\ell_h : X \times Y \to \mathbb{R}$

Risk $$r(h) := \mathbb{E}_{(X,Y)\sim D}[\ell_h(X, Y)]$$

Empirical Risk $$\hat{r}(h) := \frac{1}{n}\sum_{i=1}^{n} \ell_h(x_i, y_i)$$

PAC (Probably Approximately Correct) Learning

PAC Bound : $$\mathbb{P}(|r(h^*) - r(\hat{h})| \leq \epsilon) \geq 1 - \delta$$

i) Finite hypothesis space: $|\mathcal{H}| = k$

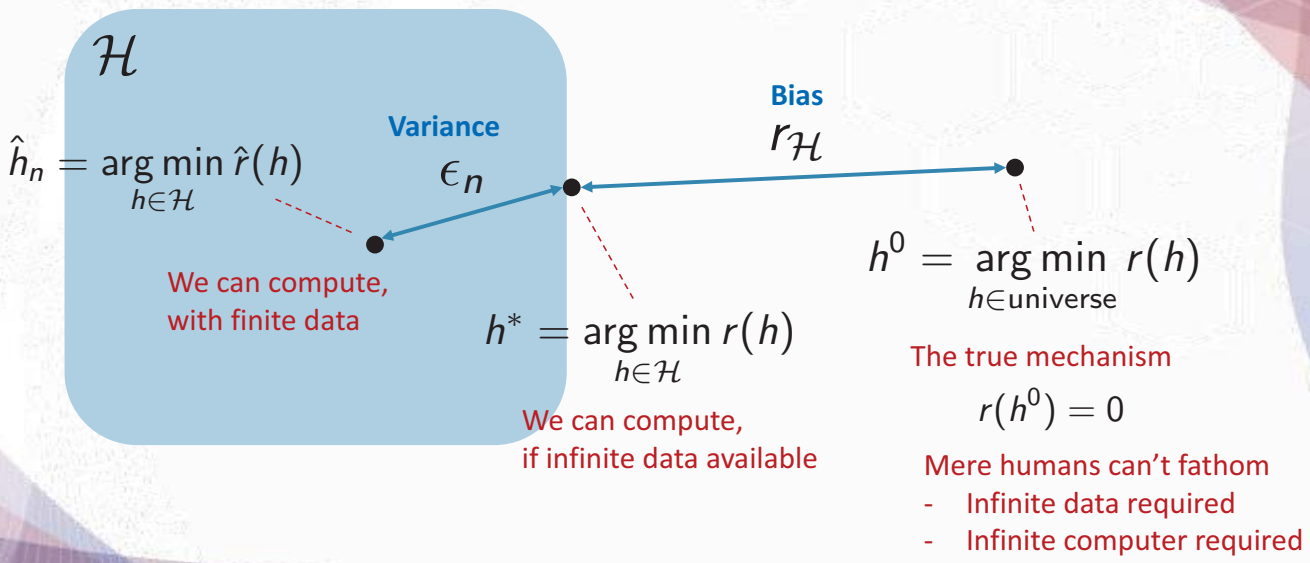
$$r(\hat{h}) \leq \underbrace{\left(\min_{h\in\mathcal{H}} r(h)\right)}_{\text{Bias}(\mathcal{H})} + \underbrace{2\sqrt{\frac{1}{2n}\log\frac{2k}{\delta}}}_{\text{Variance }(\mathcal{H})}$$

Bias-Variance Tradeoff

ii) Infinite hypothesis space: $\text{VC}(\mathcal{H}) = d$ Vapnik-Chervonenkis Dimension

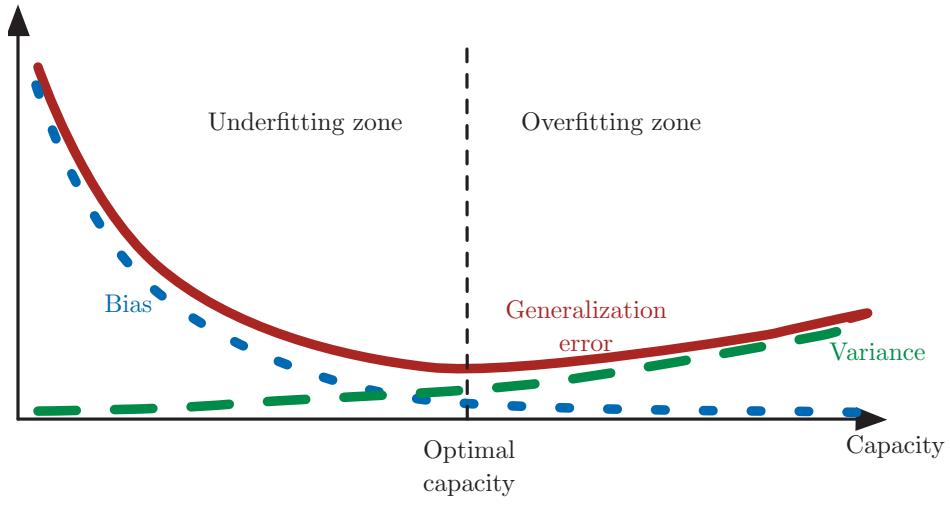
$$r(\hat{h}) \leq \left(\min_{h\in\mathcal{H}} r(h)\right) + \mathcal{O}\left(\sqrt{\frac{d}{n}\log\frac{n}{d} + \frac{1}{n}\log\frac{1}{\delta}}\right)$$

PAC (Probably Approximately Correct) Learning

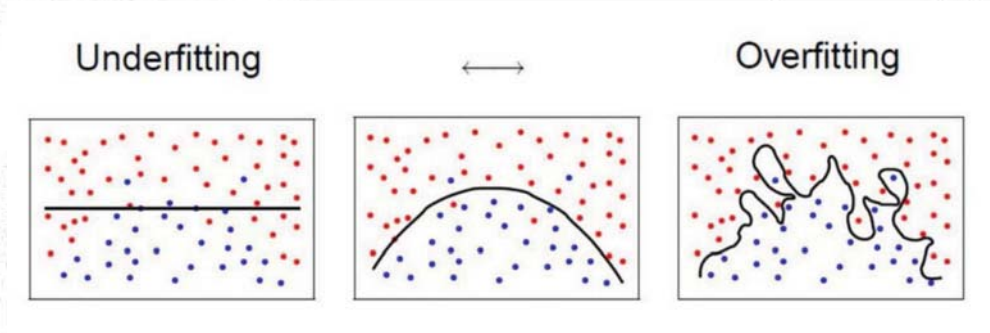


Mere humans can't fathom
- Infinite data required
- Infinite computer required

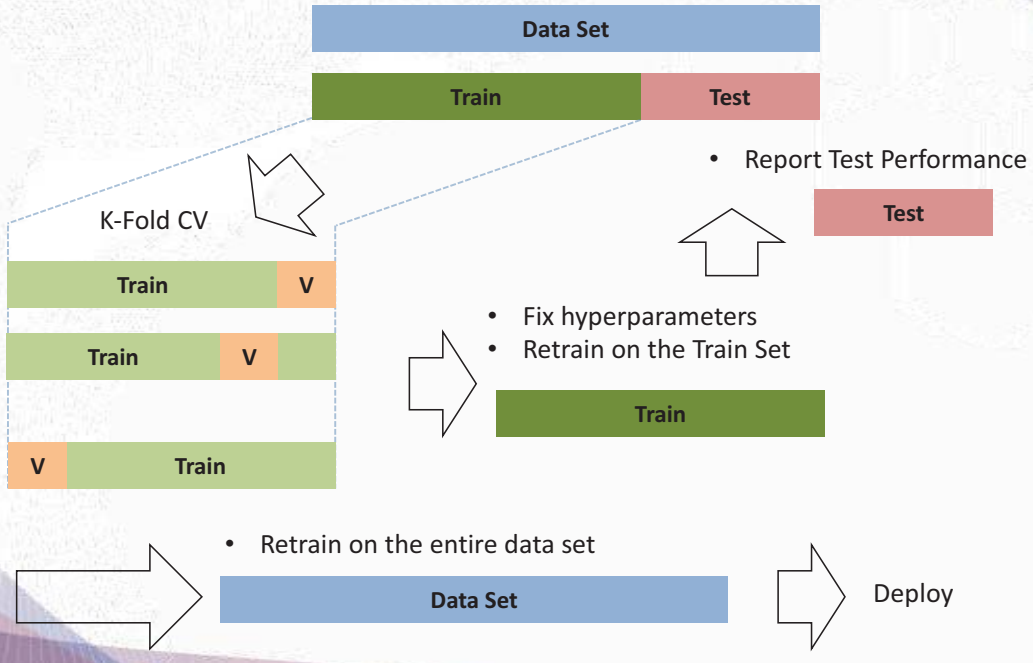
Bias-Variance Tradeoff



Overfitting Issue



ML Development Cycle

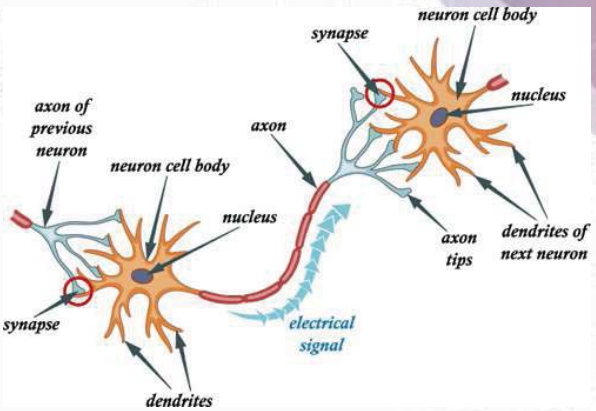


Neuron

Camillo Golgi

Santiago Ramón y Cajal

Nobel prize 1906
Structure of nerve system



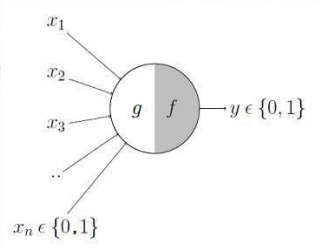
In Human Brain…

- Neurons: 약 100억
- 약 7000 synapse per neuron (Total 100조 이상)

Neural Net

Neuron [McCulloch & Pitts, 1943]

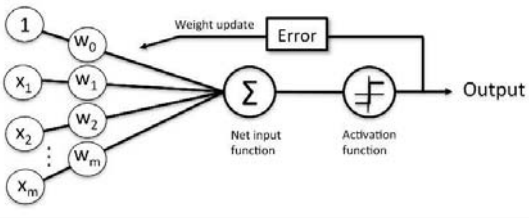
- The first computational model of a neuron



Walter Pitts, 1954 MIT

Perceptron [Rosenblatt, 1957]

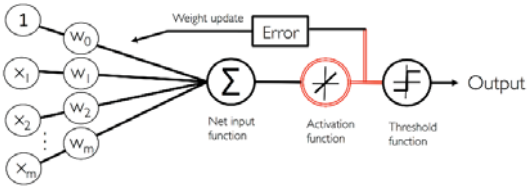
- The first neural net



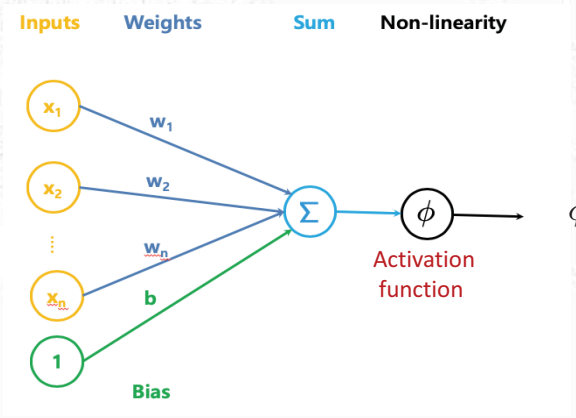
Frank Rosenblatt

ADALINE [Widrow & Hoff, 1960]

- Adaptive Linear Element



Perceptron



$$x_1^{(1)}\ x_2^{(1)}\ \cdots\ x_n^{(1)}$$

$$\phi\left(\sum_{i=1}^{n} w_i x_i\right)$$

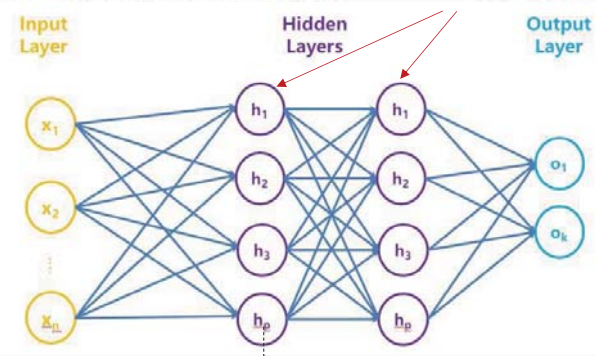
broadcasting

$$\begin{matrix} x_1^{(1)}\ x_2^{(1)}\ \cdots\ x_n^{(1)} \\ x_1^{(2)}\ x_2^{(2)}\ \cdots\ x_n^{(2)} \\ \vdots \\ x_1^{(m)}\ x_2^{(m)}\ \cdots\ x_n^{(m)} \end{matrix} \quad \begin{matrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{matrix} \quad + \quad b$$

$$X \quad * \quad w \quad + \quad b$$

ϕ Activation function → m

MLP (Multi-Layer Perceptron)
DNN (Deep Neural Network)



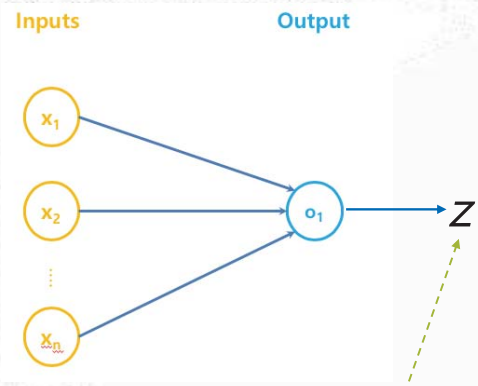
$$o_1 = \sum_{i=1}^{p} w_i \phi\left(\sum_{j=1}^{p} w'_j \phi\left(\sum_{k=1}^{n} w''_k x_k\right)\right)$$

$$\begin{matrix} x_1^{(1)}\ x_2^{(1)}\ \cdots\ x_n^{(1)} \\ x_1^{(2)}\ x_2^{(2)}\ \cdots\ x_n^{(2)} \\ \vdots \\ x_1^{(m)}\ x_2^{(m)}\ \cdots\ x_n^{(m)} \end{matrix} \quad \begin{matrix} h_1 & h_2 & & h_p \\ w_1 & w_1 & & w_1 \\ w_2 & w_2 & & w_2 \\ \vdots & \vdots & & \vdots \\ w_n & w_n & & w_n \end{matrix} \quad \begin{matrix} b_1 & b_2 & & b_p \\ \blacksquare & \blacksquare & & \blacksquare \end{matrix}$$

ϕ → $m \times p$

$$X \quad * \quad W \quad + \quad b$$

Output Transformation



Output without an activation
Called “logit”

Regression:

- $y \in \mathbb{R}$
- Use z as it is

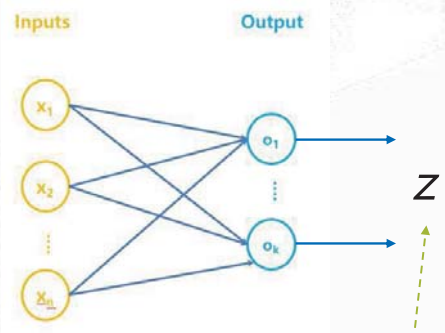
Classification:

- $y \in \{0, 1\}$
- Use

$$\mathbb{P}(Y = 1) = \sigma(z) = \frac{1}{1 + \exp(-z)}$$

Sigmoid function

Output Transformation



Output without an activation
Called “logit”

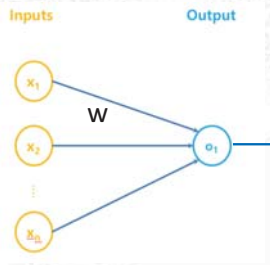
Classification:

- $y \in \{1, 2, \ldots, k\}$
- Use

$$\mathbb{P}(Y = i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}$$

Softmax(z)$_i$

Sigmoid / Softmax Output Transforms



z (logit) $\xrightarrow{\phi}$ $\mathbb{P}(Y=1) = \sigma(z) = \dfrac{1}{1+\exp(-z)}$

$\dfrac{p}{1-p}$ Odds ratio

$p = \mathbb{P}(Y=1; w)$

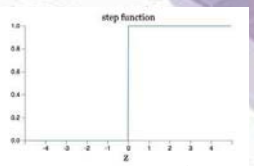
$\log \dfrac{p}{1-p}$ Log odds ratio = logit

Logistic regression:

$$\log \frac{p}{1-p} = z = w^T x \quad \Rightarrow \quad p = \sigma(z) = \sigma(w^T x)$$

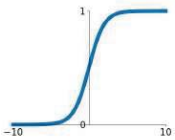
Softmax function can be derived in a similar way.

Activation Functions

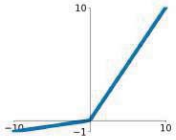


Alternatives to the step function

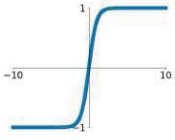
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

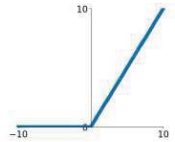


tanh
$\tanh(x)$

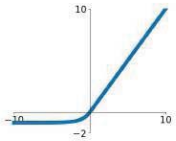


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

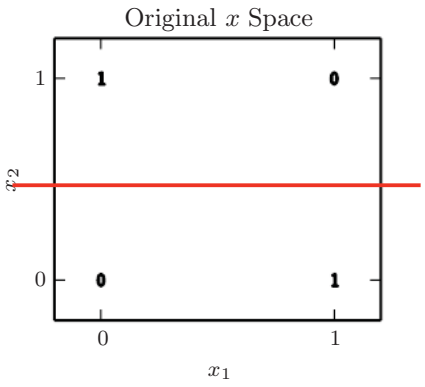


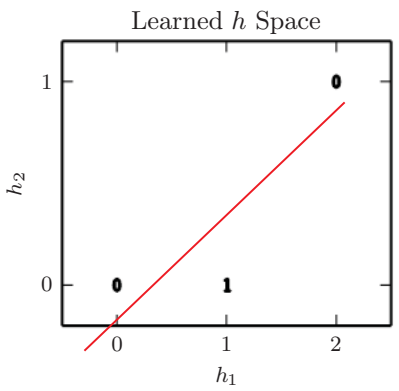
(Rectifier Linear Unit)

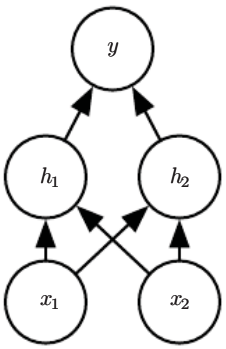
(Exponential Linear Unit) $\alpha > 0$

Benefit of Depth

- Solving XOR problem







Benefit of Depth

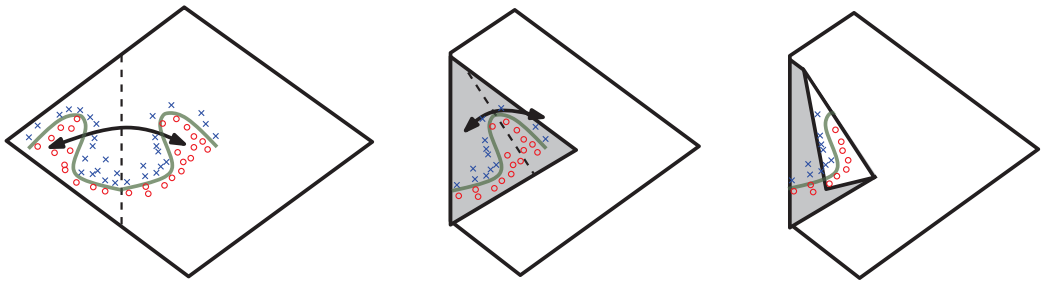
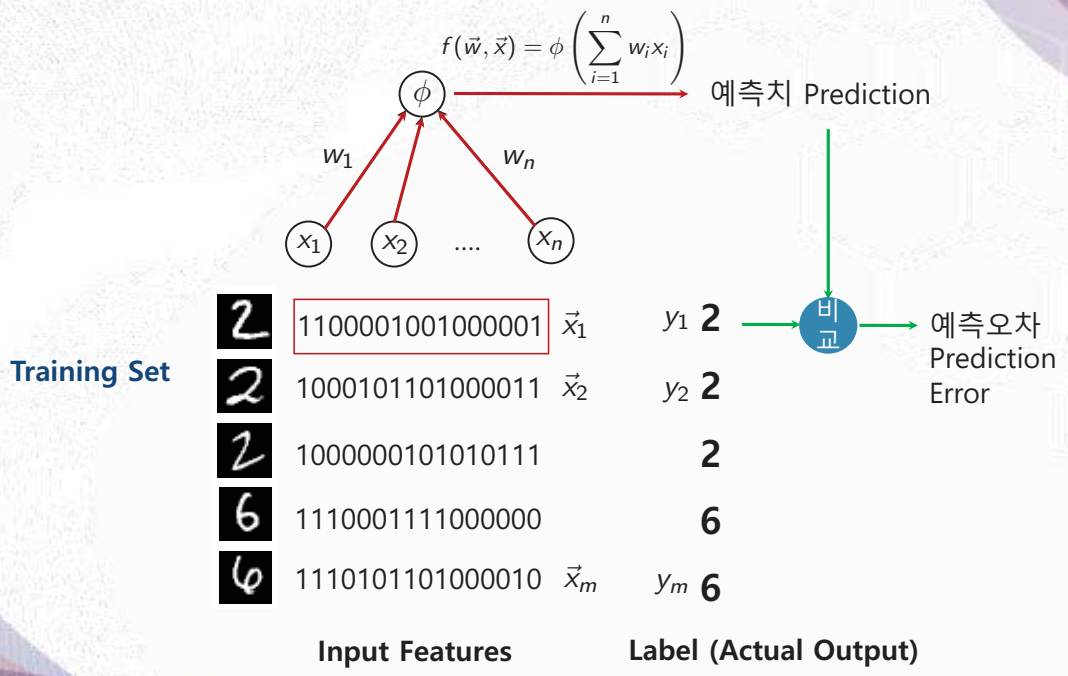


Figure 6.5: An intuitive, geometric explanation of the exponential advantage of deeper rectifier networks formally shown by Pascanu *et al.* (2014a) and by Montufar *et al.* (2014). *(Left)* An absolute value rectification unit has the same output for every pair of mirror points in its input. The mirror axis of symmetry is given by the hyperplane defined by the weights and bias of the unit. A function computed on top of that unit (the green decision surface) will be a mirror image of a simpler pattern across that axis of symmetry. *(Center)* The function can be obtained by folding the space around the axis of symmetry. *(Right)* Another repeating pattern can be folded on top of the first (by another downstream unit) to obtain another symmetry (which is now repeated four times, with two hidden layers).

Training



Training = Numerical Optimization

- Training (학습): 주어진 training 데이터에서 예측오차를 최소화하는 최적 기계학습 파라미터 $\vec{w}^* = (w_1^*, \ldots, w_n^*)$의 값을 찾는 문제

m: Training 데이터 포인트 수

$$\vec{w}^* \in \arg\min_{\vec{w}\in\mathbb{R}^n} J(\vec{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, f(\vec{w}, \vec{x}_i))$$

n: 학습 모델의 파라미터 수 (최적화 문제의 차원)

Loss function

- n 또는 m 이 큰 경우 (Big Data), 또는 loss function이 다루기 어려운 경우 (e.g. 미분불가), 효율적인 **수치 최적화** 알고리듬이 필요함

Loss Functions

Binary Cross Entropy: $y_i \in \{0, 1\}$

$$-\ell(y_i, f(\vec{w}, \vec{x}_i)) = y_i \log(f(\vec{w}, \vec{x}_i)) + (1 - y_i) \log(1 - f(\vec{w}, \vec{x}_i))$$

Mean Square Error (MSE): $y_i \in \mathbb{R}$

$$\ell(y_i, f(\vec{w}, \vec{x}_i)) = (y_i - f(\vec{w}, \vec{x}_i))^2$$

Likelihood Function

Likelihood

$$\mathbb{P}(o_1, o_2, \ldots, o_n; \theta)$$

- **Joint probability observations under the model**
- **Probability that the model has generated the observations**
- **A function in θ**

MLE (Maximum Likelihood Estimation)

- A coin toss problem in elementary school:
  - Assume a fair dice: *P(X = i) = 1/6, i = 1,2,…,6*
  - Toss the dice 10 times, where each toss is independent
  - What is the probability of the event, $\{3, 6, 1, 4, 2, 2, 4, 5, 6, 3\}$?

- Given the observations: $\{3, 6, 1, 4, 2, 2, 4, 5, 6, 3\}$
  - What will be a good guess of *P(X = i)* ?

MLE (Maximum Likelihood Estimation)

- Given observations: $\{o_1, o_2, \ldots, o_n\}$ $\quad o_i \overset{i.i.d.}{\sim} \mathbb{P}(O; \theta)$

- Likelihood function $\quad L(\theta) = \mathbb{P}(o_1, o_2, \ldots, o_n; \theta)$
  - Joint probability of observations under the model (parameter: θ)
  - Probability that the model has generated the observations
  - A function in θ

- MLE: find the θ that maximizes the likelihood

$$\max_{\theta} L(\theta) \qquad \min_{\theta} -LL(\theta) = -\log L(\theta)$$

Negative log likelihood (NLL)

  - MLE is *efficient*: given n examples, MLE is the most accurate procedure to estimate the parameters

MLE for Binary Classification

Conditional Bernoulli model of labels:

$$P(Y=1|X=x;w) = \sigma(f(w,x_i)) = \frac{1}{1+\exp(-f(w,x_i))}$$
$$P(Y=0|X=x;w) = 1-\sigma(f(w,x_i))$$

(Conditional) Log likelihood function:

i.i.d

$$\log P(y_1,\ldots,y_n|x_1,\ldots,x_n;w) = \log \prod_{i=1}^{n} P(y_i|x_i;w)$$

$$= \log \prod_{i=1}^{n} P(y_i=1|x_i;w)^{y_i} P(y_i=0|x_i;w)^{1-y_i}$$

$$= \sum_{i=1}^{n} \{y_i \log \sigma(f(w,x_i)) + (1-y_i)\log(1-\sigma(f(w,x_i)))\}$$

MLE for Multi-Class Classification

Softmax function: $\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \quad i=1,2,\ldots,k$

$$P(Y=k|x) = \text{softmax}(f(w,x))_k$$

Log likelihood function:

$$= \log \prod_{i=1}^{n} P(y_i=1|x_i;w)^{I_{y_i=1}} \ldots P(y_i=K|x_i;w)^{I_{y_i=K}}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K} I_{y_i=k} \log \text{softmax}(f(w,x_i))_k$$

Training Problem (Classification)

$$\max_{\theta} \log \mathbb{P}(o_1, o_2, \ldots, o_n; \theta)$$

Minimization of Negative Log Likelihood function:

$$-\min_{\theta} -\log \mathbb{P}(o_1, o_2, \ldots, o_n; \theta)$$

Training Problem

Minimization of Negative Log Likelihood function:

$$\min_{w \in \mathbb{R}^p} -\log P(y_1, \ldots, y_n | x_1, \ldots, x_n; w)$$

$$\min_{w \in \mathbb{R}^p} -\sum_{i=1}^{n} \{y_i \log f(w, x_i) + (1 - y_i) \log(1 - f(w, x_i))\}$$

In general, the training of DNN (and many other ML models) can be written as:

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} \ell(w; x_i, y_i)$$

Loss function

Training Problem

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} \ell(w; x_i, y_i)$$

Task	Labels	Loss function
Regression	$y_i \in \mathbb{R}$	$\ell(w; x_i, y_i) = (y_i - f(w, x_i))^2$
Classification (Binary)	$y_i \in \{0, 1\}$	$-\ell(w; x_i, y_i) = y_i \log f(w, x_i) + (1 - y_i) \log(1 - f(w, x_i))$
Classification (Multi-Class)	$y_i \in \{1, \ldots, K\}$	$-\ell(w; x_i, y_i) = \sum_{k=1}^{K} I_{y_i=k} \log \text{softmax}(f(w, x_i))_k$

Stochastic Gradient Descent (SGD)

$$\vec{w}^* \in \arg\min_{\vec{w} \in \mathbb{R}^n} J(\vec{w}) = \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, f(\vec{w}, \vec{x}_i))$$

- Initialize w randomly
- For N epochs
  - For a random training example $J_i(w) = \ell(y_i, f(\vec{w}, \vec{x}_i))$
  - Compute stochastic (sub)gradient of loss: $\frac{\partial J_i(w)}{\partial w}$
  - Update w: $w = w - \eta \frac{\partial J_i(w)}{\partial w}$

Learning rate

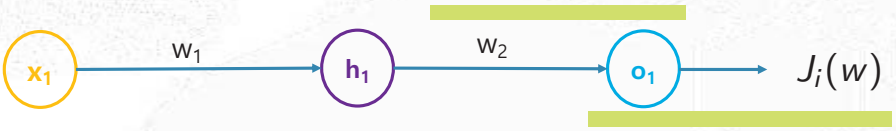
Mini-Batches

Use a small subset of examples per update, to reduce variance of gradient estimates

- Initialize w randomly
- For N epochs
  - For minibatch samples $J_i(w) = \ell(y_i, f(\vec{w}, \vec{x}_i))$
  - Compute stochastic (sub)gradient of loss: $\frac{\partial J(w)}{\partial w} \approx \frac{1}{B} \sum_i^B \frac{\partial J_i(w)}{\partial w}$
  - Update w:

$$w = w - \eta \frac{\partial J(w)}{\partial w}$$

Computing Gradient: Back Propagation

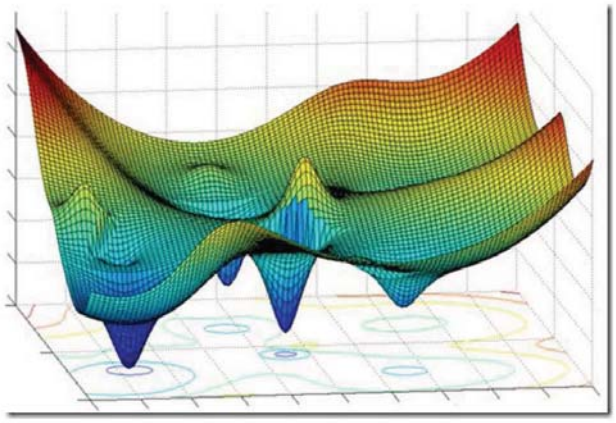


$$\frac{\partial J_i(w)}{\partial w_2} =$$

$$\frac{\partial J_i(w)}{\partial w_1} = \frac{\partial J_i(w)}{\partial o_1} \frac{\partial o_1}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

DNN Objective Functions

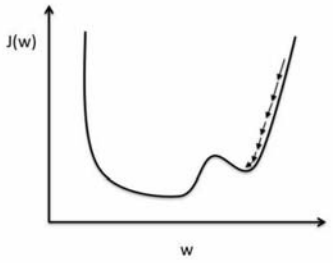
Objective functions of DNN are typically nonconvex, with lots of local minimizers and saddle points



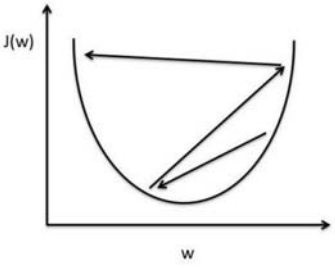
Learning Rate is Important

$$w = w - \eta \frac{\partial J(w)}{\partial w}$$

How to choose the learning rate?



Small learning rate: Many iterations until convergence and trapping in local minima.



Large learning rate: Overshooting.

Adaptive Learning Rate

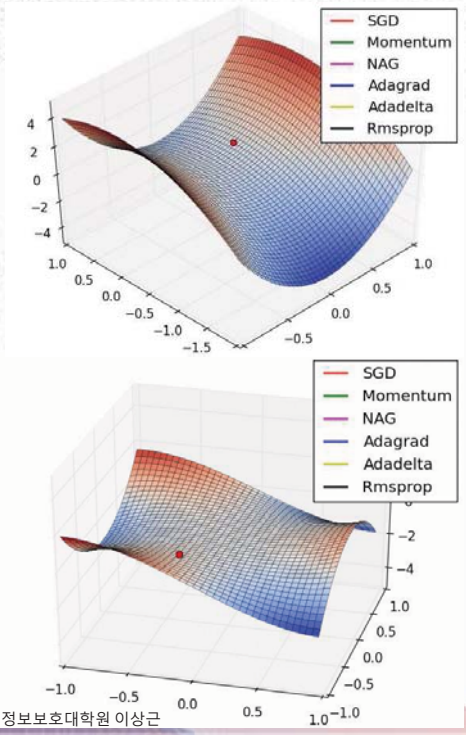
Learning rates can be chosen adaptively to:

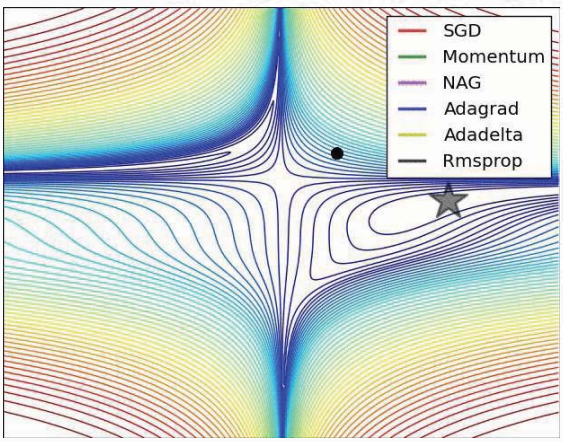
- How large the gradient is
- How fast learning is happening
- Magnitude of particular weights
- ....

Adaptive learning rate algorithms:

ADAM, Momentum, NAG, Adagrad, Adadelta, RMSProp, ...

Adaptive Learning Rate Algorithms





Regularization to Avoid Overfitting

Dropout: in training, randomly set some activations to zero

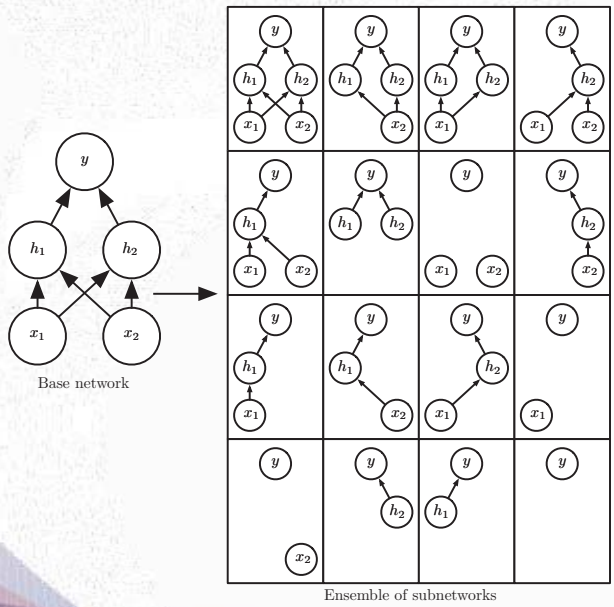
- Typically drop 50% of activations in layers
- Forces the network not to rely on small set of nodes

Early Stopping:

Weight regularization

$$\vec{w}^* \in \arg\min_{\vec{w}\in\mathbb{R}^n} J(\vec{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, f(\vec{w}, \vec{x}_i)) + \lambda \|w\|_2^2$$

Dropout [Srivastava et al., 2014]



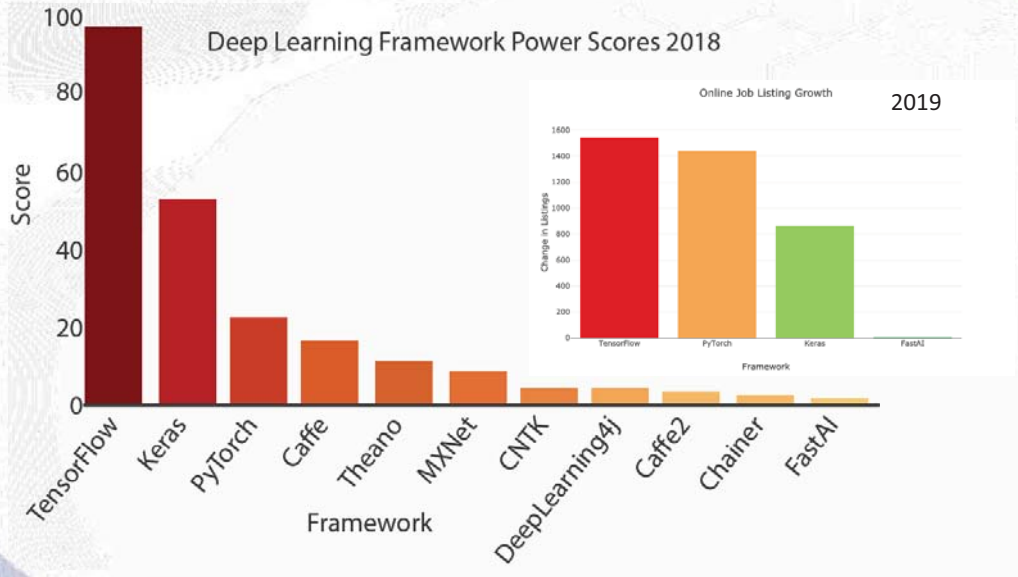
Inexpensive but powerful method of regularization

Dropout trains the ensemble consisting of all sub-networks that can be formed by removing non-output units from an underlying base network

In each step of the SGD, a different binary mask is sampled to apply to all input and hidden units

Large networks are preferred to apply dropout

Deep Learning Frameworks



Data Labeling Cost

Amazon SageMaker

amazon mechanical turk

LIONBRIDGE

Edgecase.ai

https://medium.com/cogitotech/what-is-the-pricing-for-data-labeling-and-annotations-47608da9073d

ML / DL Platforms (Python)

- ML : scikit-learn
- DL

Caffe (UC Berkeley) → Caffe2 (Facebook)

Torch (NYU / Facebook) → PyTorch (Facebook)

Theano (U Montreal) → TensorFlow (Google)

Paddle (Baidu)

CNTK (Microsoft)

MXNet (Amazon)
Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

And others...

- Home-brewed ML / DL toolkits?

Leaders in DL

Ian Goodfellow

- Google Brain
- OpenAI

Yoshua Bengio

- U Montreal
- Head of MILA
- Theano

Yann LeCun

- NYU
- Head of FAIR
- Inventor of CNN

Alex Smola

- CMU
- Head of Amazon AI

OpenAI

Mila

Facebook AI Research

Amazon AI

Adventures in Data Land

Leaving CMU

...

It has been wonderful to work with you and I dearly love CMU. So why the change?

Here's the reasoning that went into deciding to go to Amazon: **Our goal as machine learning researchers is to solve deep problems (not just in deep learning) and to ensure that this leads to algorithms that are actually used**. At scale. At sophistication. In applications. The number of people I could possibly influence personally through papers and teaching might be 10,000. In Amazon we have 1 million developers using AWS. Likewise, the NSF thinks that a project of 3 engineers is a big grant (and it is very choosy in awarding these grants). At Amazon we will be investing an order of magnitude more resources towards this problem. With data and computers to match this. This is significant leverage. Hence the change.

…

https://blog.smola.org/post/145983963411/leaving-cmu (2016)

Turing Award 2018

https://www.youtube.com/watch?v=HzilDIhWhrE

References

- For grad. level study
  - Elements of statistical learning, Springer, 2009
  - Machine learning, K. Murphy, MIT Press, 2012
    - https://github.com/probml/pml-book
- Theory
  - Foundations of machine learning, R. Mohri, MIT Press, 2018
  - Nature of statistical learning theory, V. Vapnik, Springer, 2000
  - Statistical learning theory, V. Vapnik, Wiley, 1998
- Links of ML to other fields
  - Information theory, inference, and learning algorithms, D. MacKay, Cambridge, 2017 (free pdf available)

Thank You

SBi 한국생명정보학회 Korean Society for Bioinformatics

고려대학교 KOREA UNIVERSITY

AI LAB

Introduction to Deep Learning

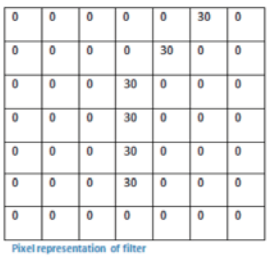
Convolutional Neural Networks

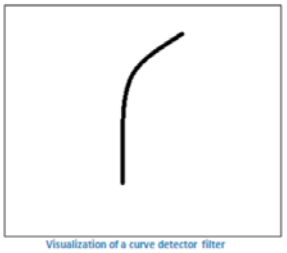
고려대학교 정보보호대학원 인공지능연구실 이상근

KSBi-BIML 2023

Convolution

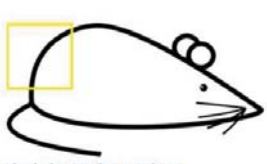
필터 Filter
(커널 Kernel)





학습대상
이미지

Convolution



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Convolution

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

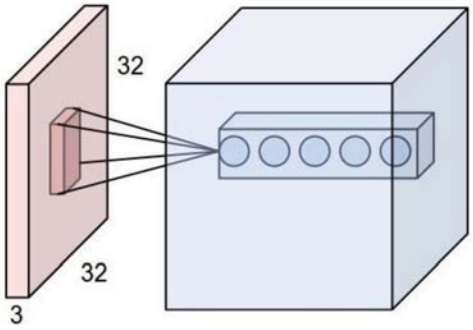
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

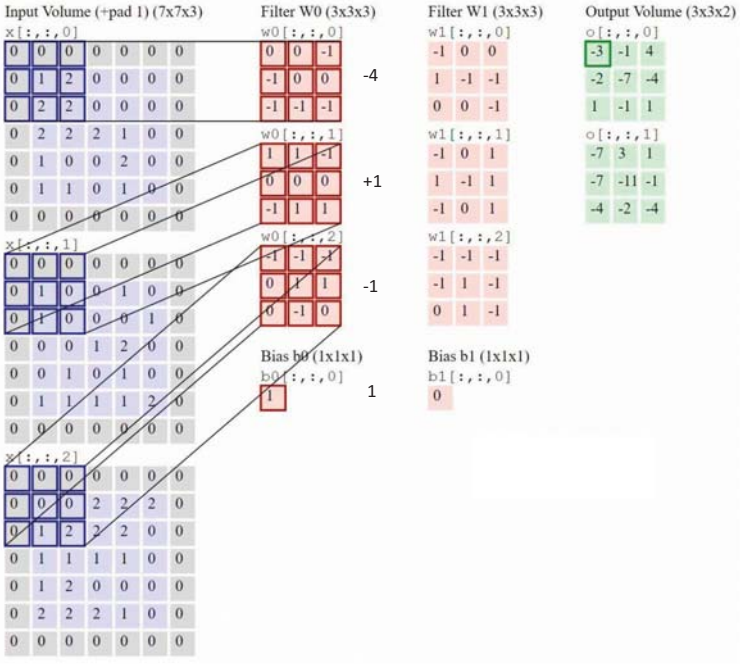
Multiplication and Summation = 0

Terminology

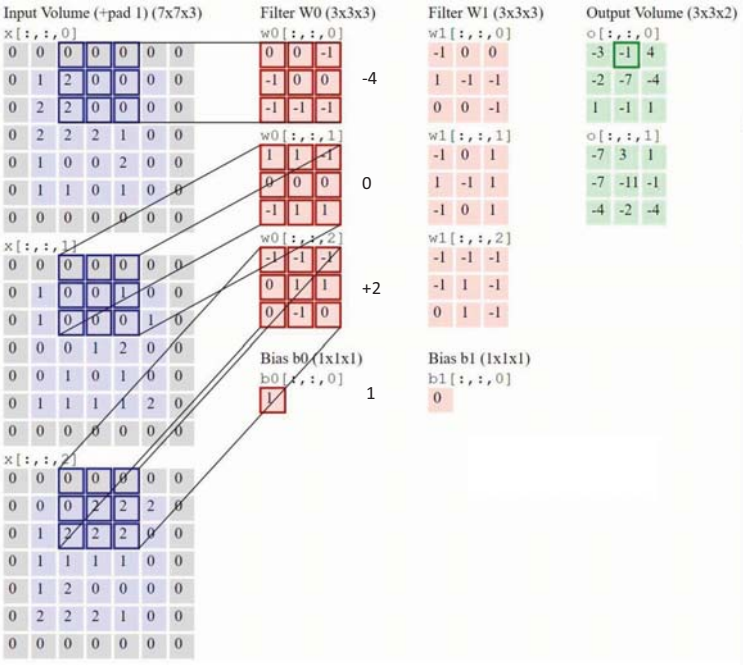


- **Depth:** number of filters
- **Stride:** filter step size (when we "slide" it)
- **Padding:** zero-pad the input

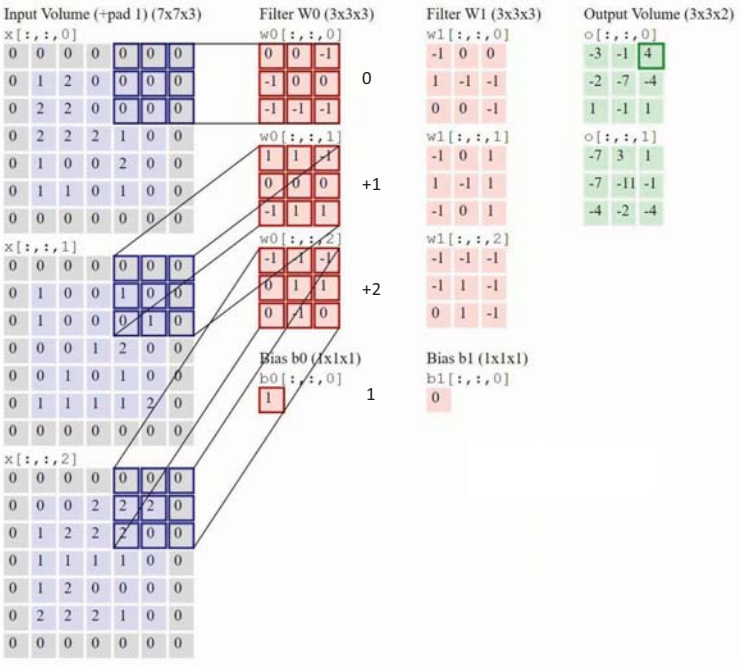
Convolution (Padding 1, Stride 2)



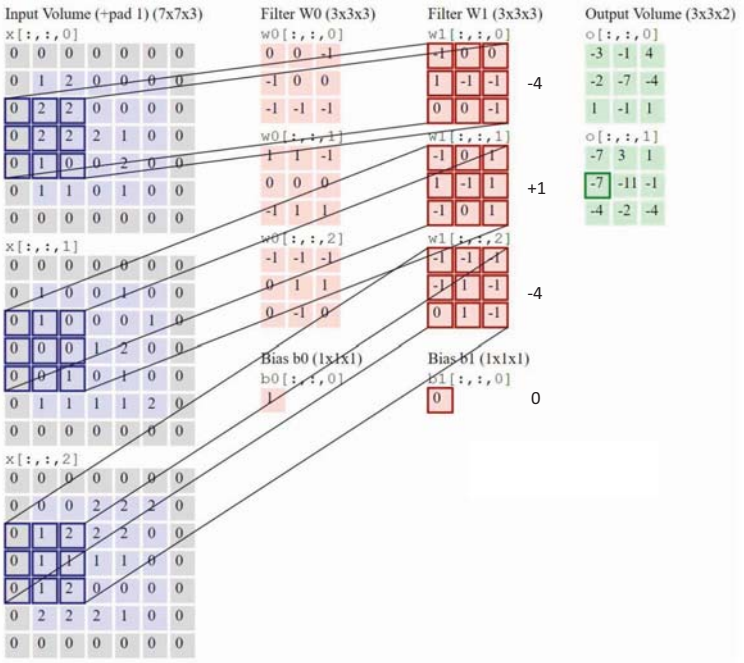
Convolution (Padding 1, Stride 2)



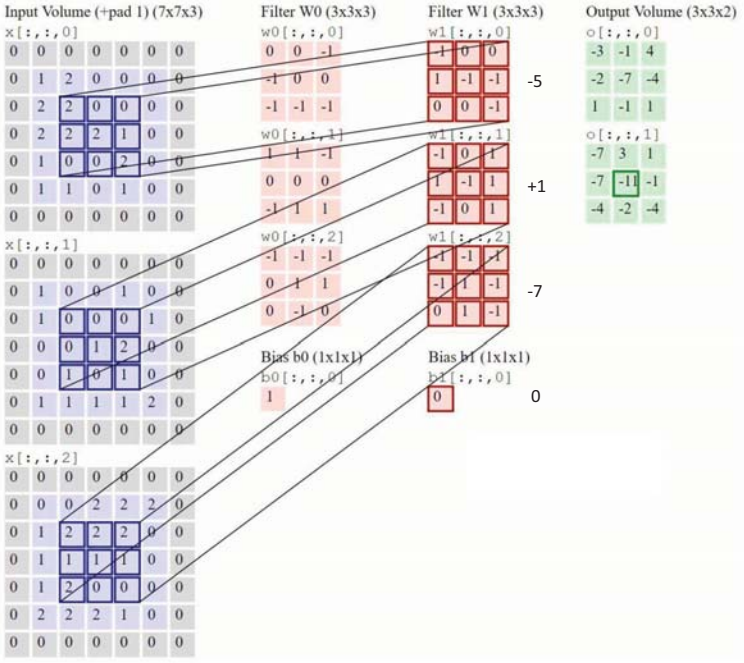
Convolution (Padding 1, Stride 2)



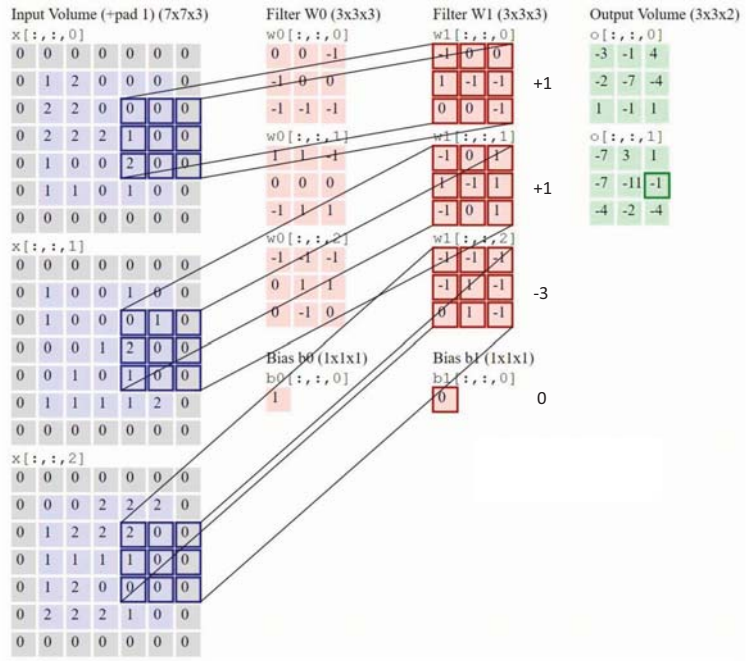
Convolution (Padding 1, Stride 2)



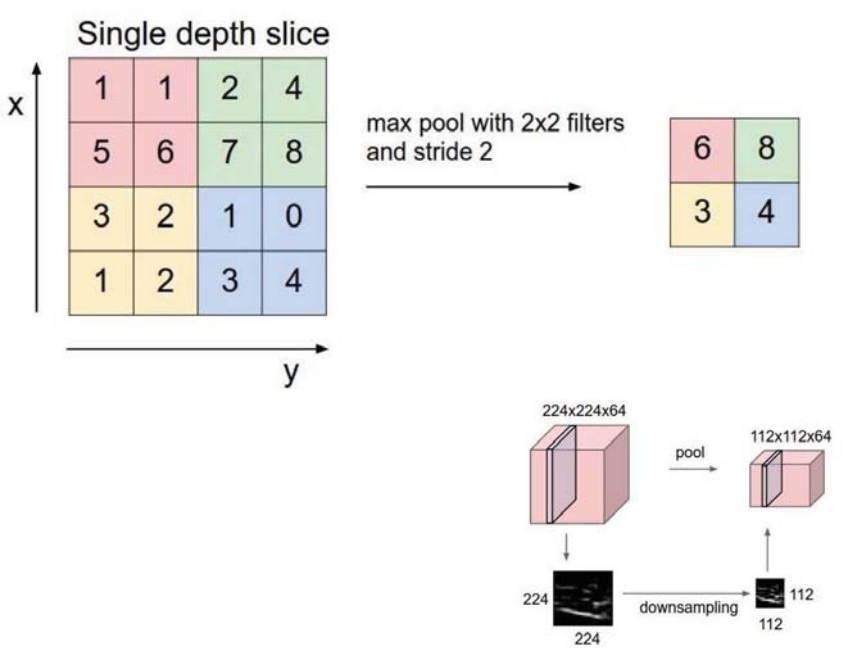
Convolution (Padding 1, Stride 2)



Convolution (Padding 1, Stride 2)

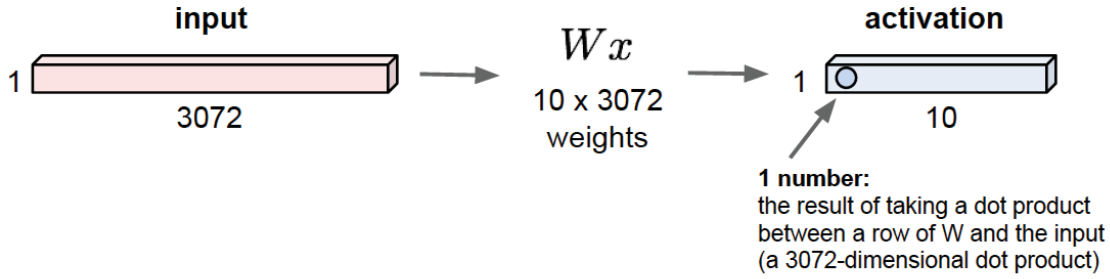


Pooling



FC (Fully Connected) Layer

32x32x3 image -> stretch to 3072 x 1

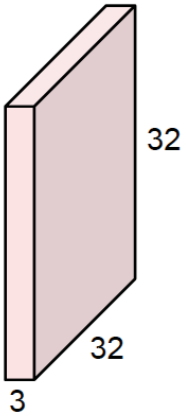


1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

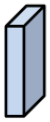
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

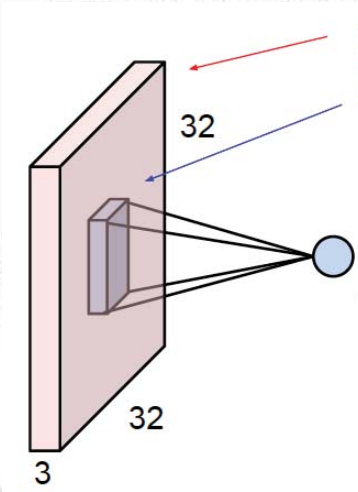


5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

Convolution Layer

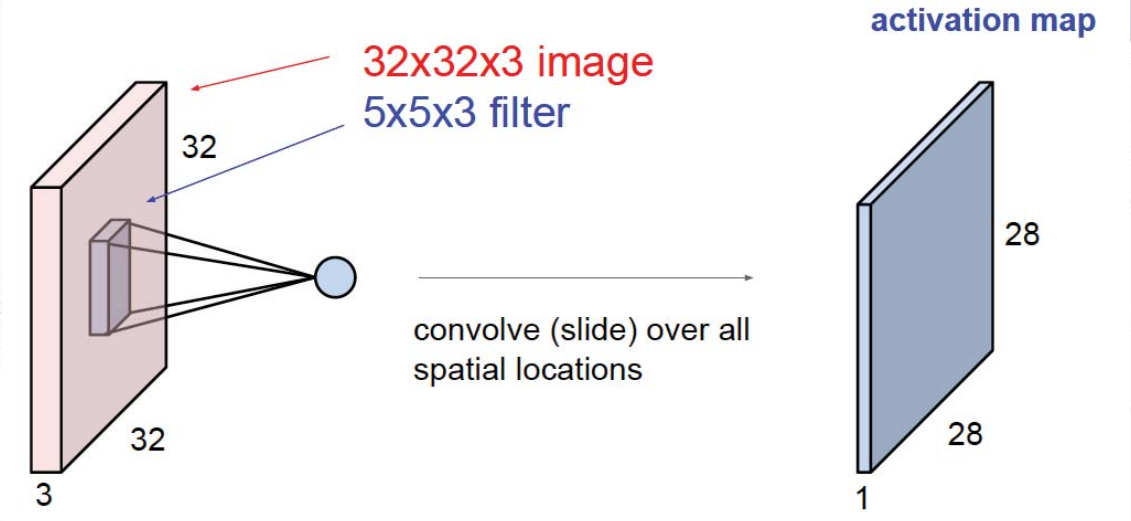


32x32x3 image
5x5x3 filter w

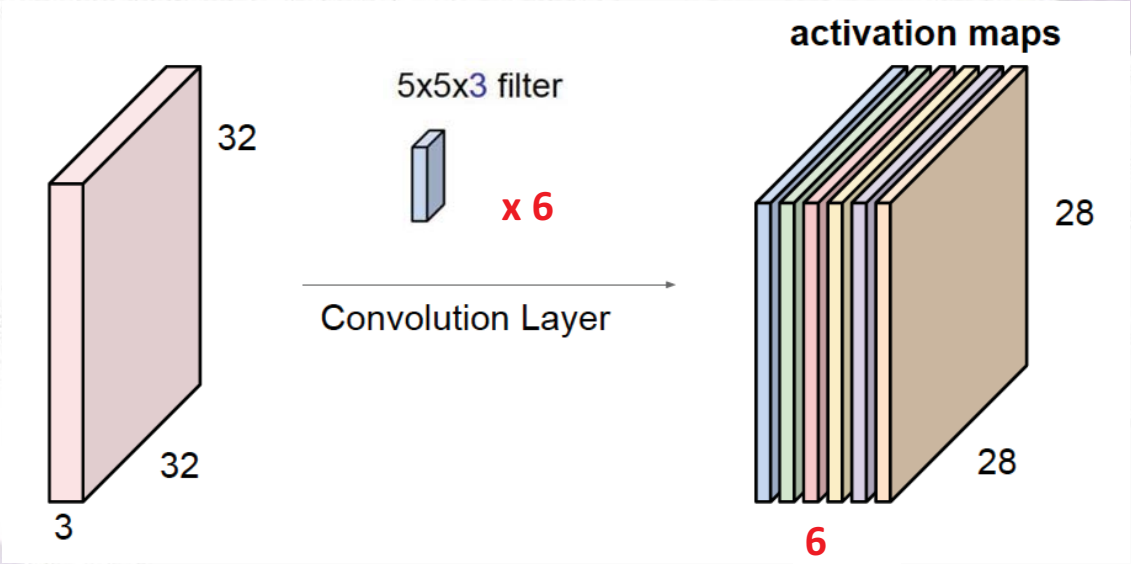
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

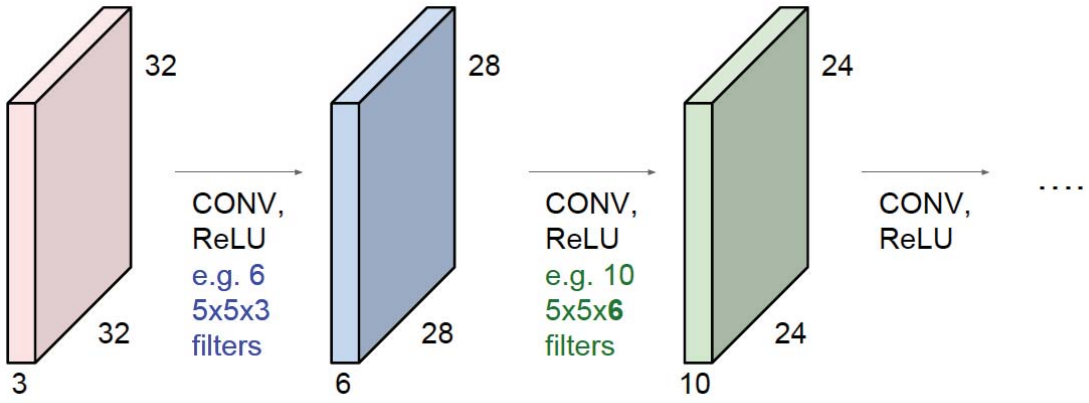


Convolution Layer

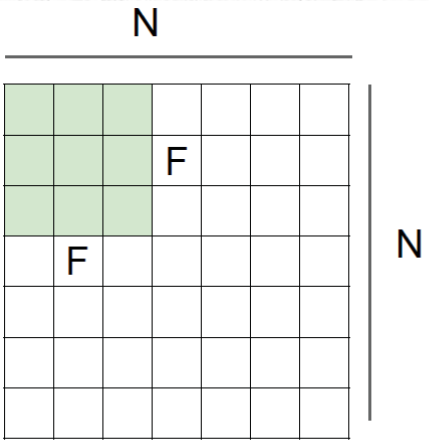


We stack these up to get a “new image” of size 28x28x6!

CNN



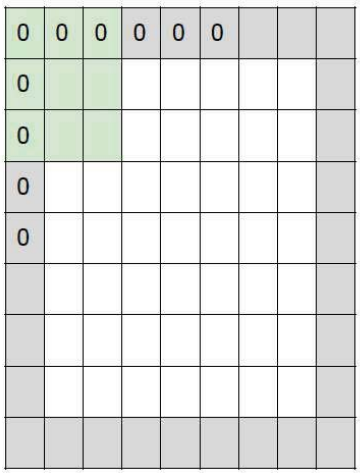
Output Size



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Output Size with Zero-Padding



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Output Size

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

Output Size

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

No. of Parameters

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

No. of Parameters

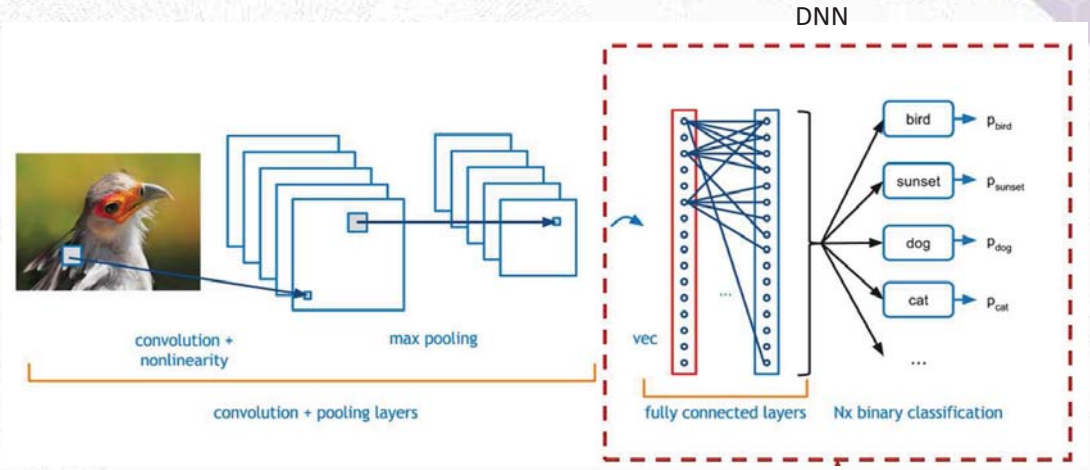
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params
=> 76*10 = **760**

Spatial Dim & No. Parameters

- Input volume: W1 x H1 x D1
- Filter (kernel)
  - No of filters K
  - Spatial extent F
  - Stride S
  - Amount of zero padding P
- Output volume: W2 x H2 x D2
  - W2 = (W1-F+2P)/S+1
  - H2 = (H1-F+2P)/S+1
  - D2 = K
- With weight sharing,
  - (F x F x D1) x K weights
  - K biases

CNN Architecture



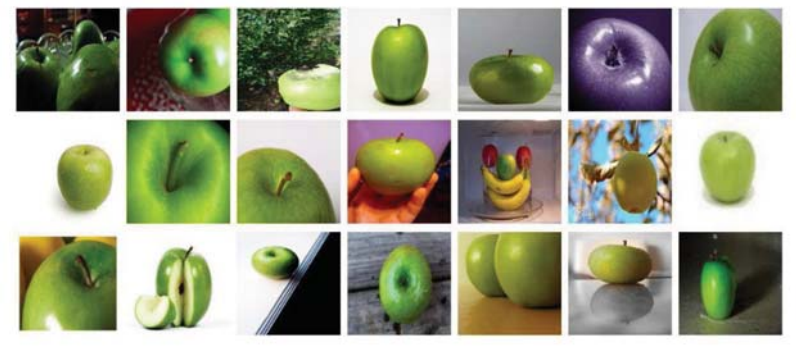
응용에 따라 Convolution 부분이나 DNN 구조를 바꿀 수 있음

ImageNet Data

Dataset of 14+ million images of 21,841 categories

Category “Fruit”: 180,000 images

- 1206 Granny Smith apples

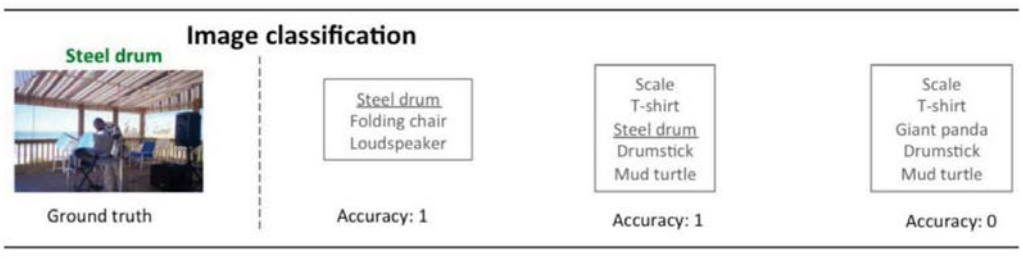


ILSVRC : ImageNet Large Scale Visual Recognition Challenge

ILSVRC Challenge

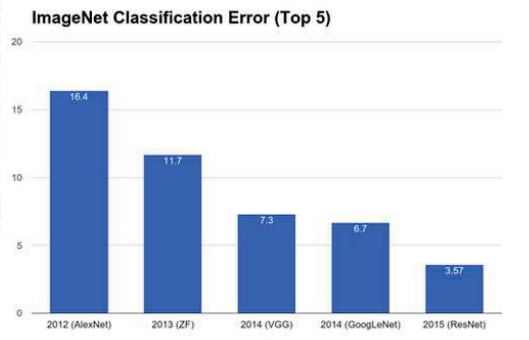
Top 5 error rate:

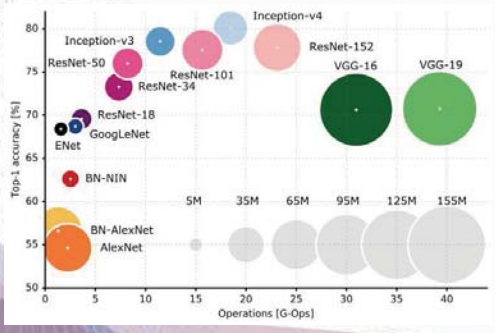
- Can make 5 guesses to get the correct label



Human annotation: binary (“apple” or “not apple”)

ILSVRC Challenge



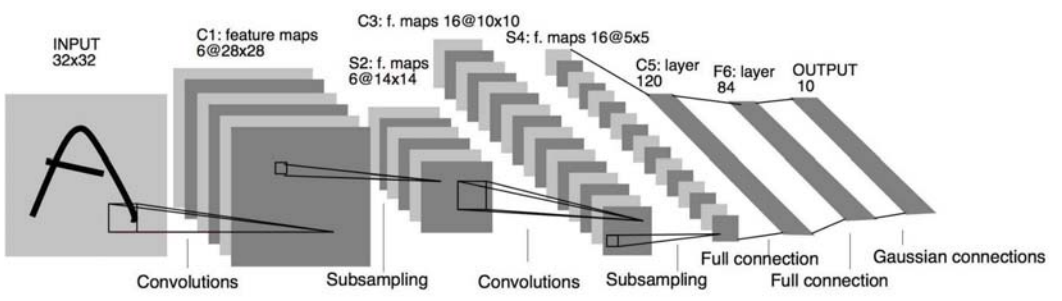


- **AlexNet (2012): First CNN for ILSVRC (16.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 16.4% to 11.7%**
  - 8 layers
  - More filters, denser stride.
- **VGGNet (2014): 11.7% to 7.3%**
  - Beautifully uniform:

  3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogleNet (2014): 11.7% to 6.7%**
  - Inception module
  - 22 layers
  - 5 million parameters

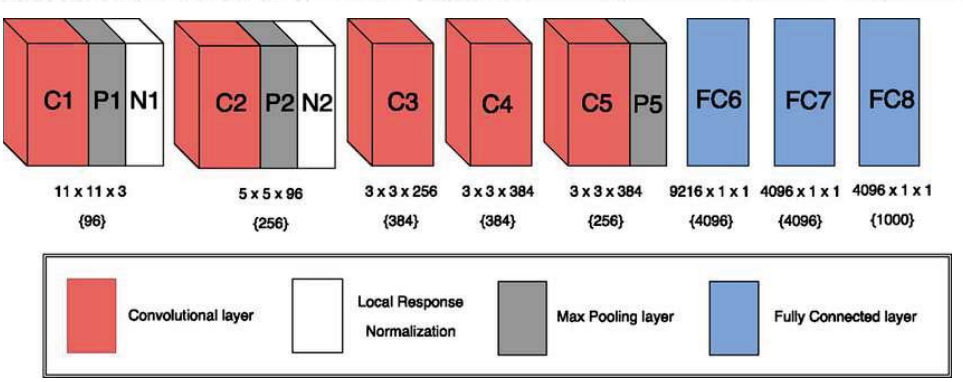
  (throw away FC layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

LeNet5 (1994)



- The very first CNN, by Yann LeCun (1994)
- No GPU computation, non-linearity = sigmoid / tanh
- Insight:
  - Image features are distributed across the entire image
  - Convolutions with learnable features

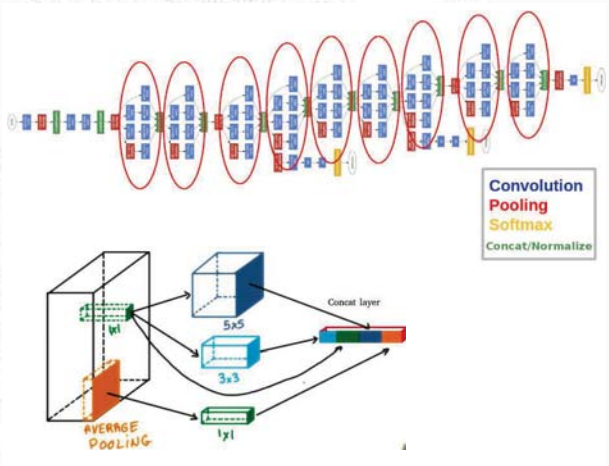
AlexNet (2012)



- Extension of LeNet5 to learn more complex objects and object hierarchy
- ReLU, dropout, overlapping max pooling, NVIDIA GTX 580

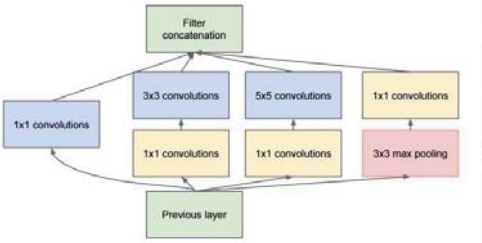
Krizhevsky et al. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

GoogLeNet (2014)



Goal: reduce no. of parameters by going deeper

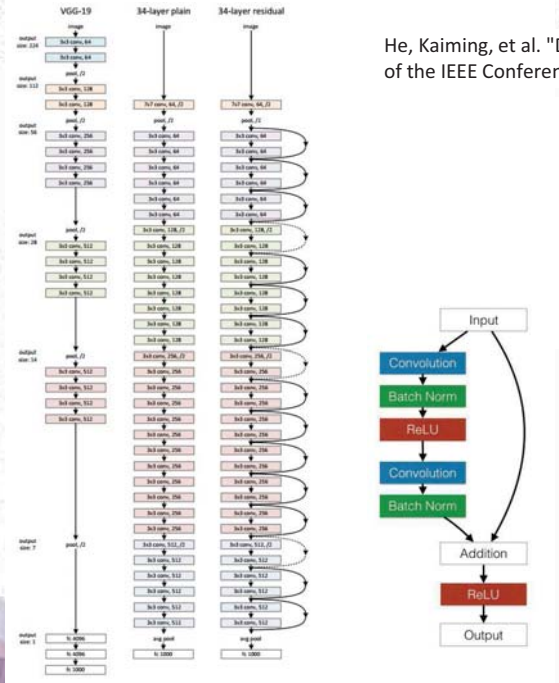
"Inception" module:



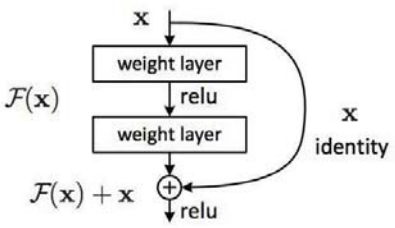
Szegedy et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

1x1 Convolution

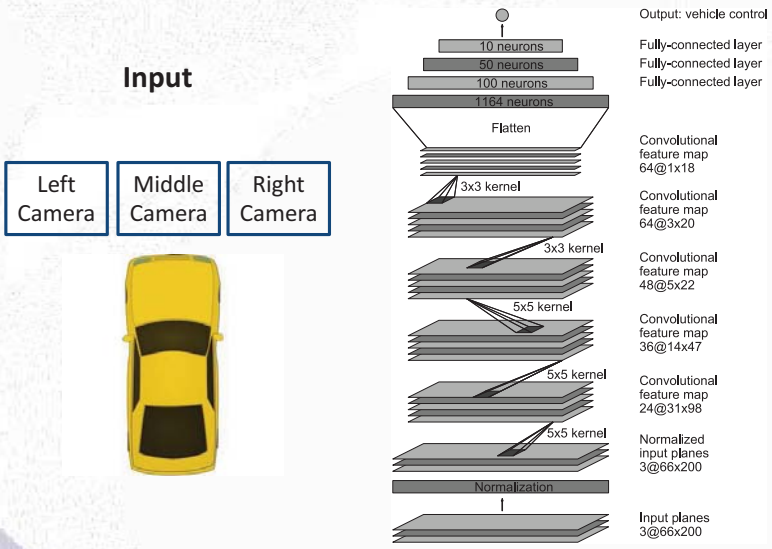
ResNet (2015)



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

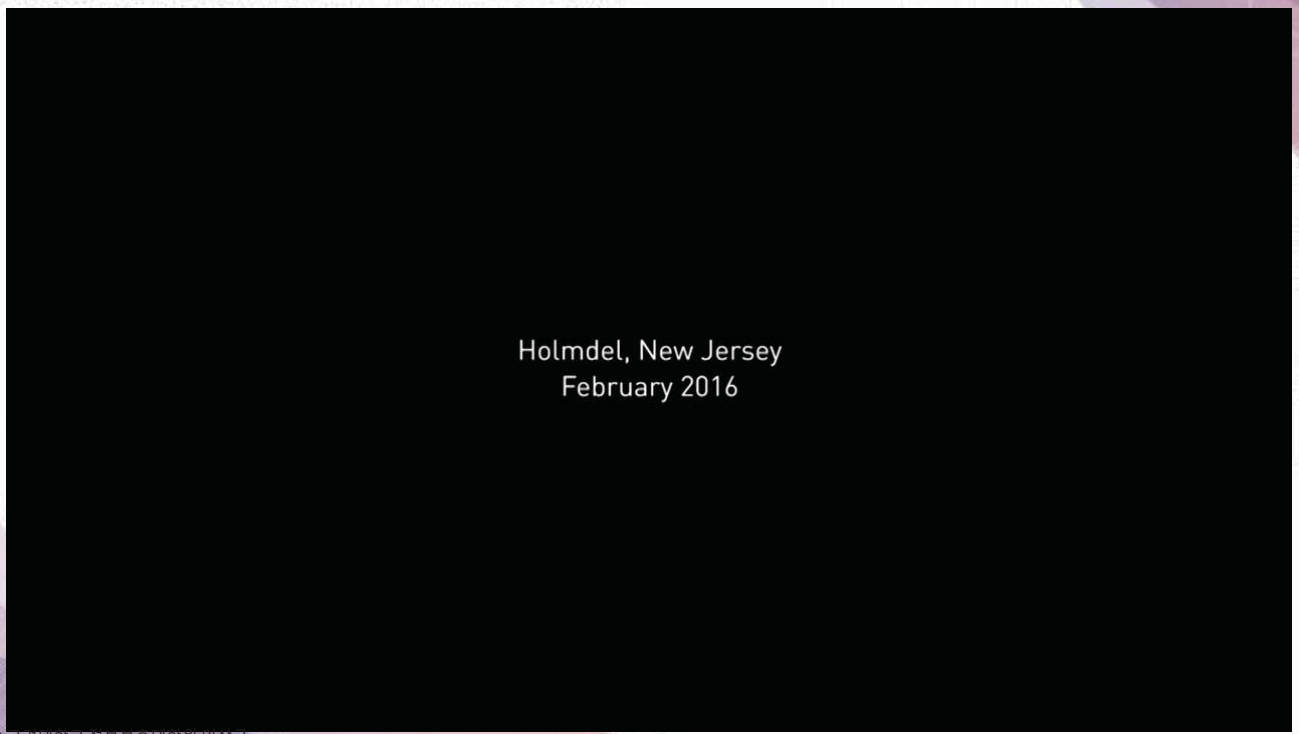


NVIDIA End-to-End CNN (2016)

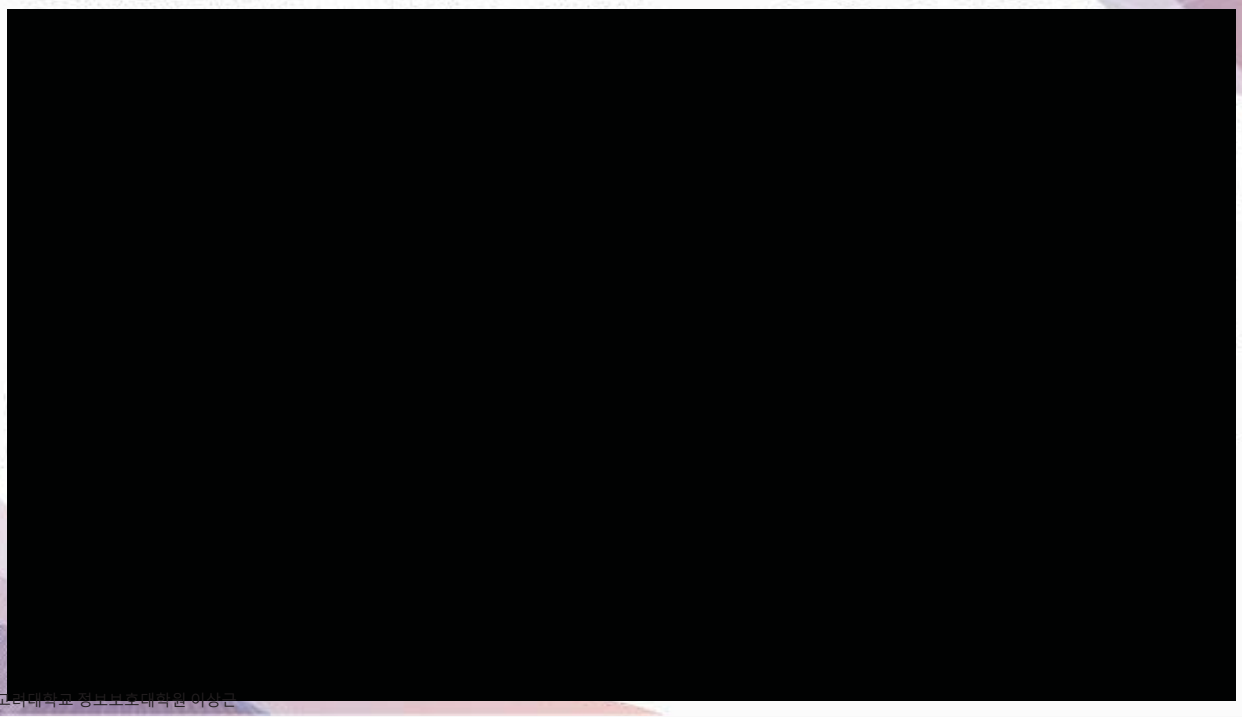


# of parameters: 250,000

NVIDIA Self-Driving (2016)

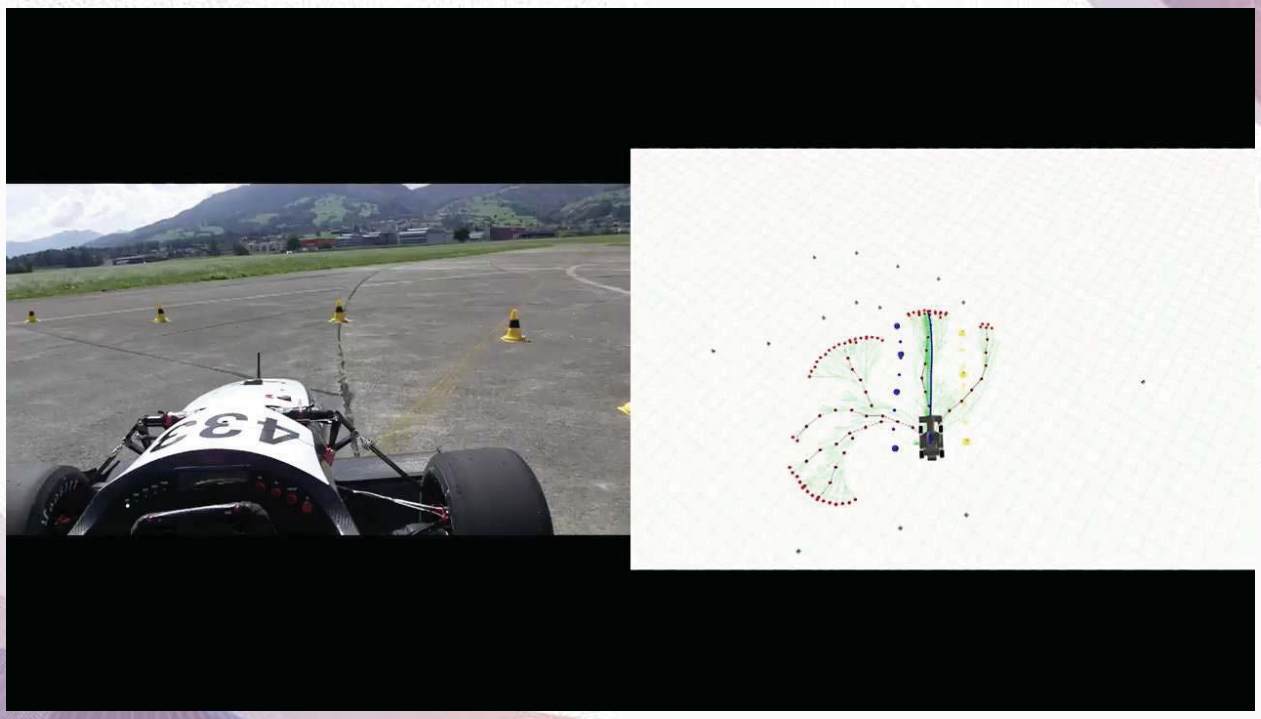


자율주행 시스템 (NVIDIA CES 2017)

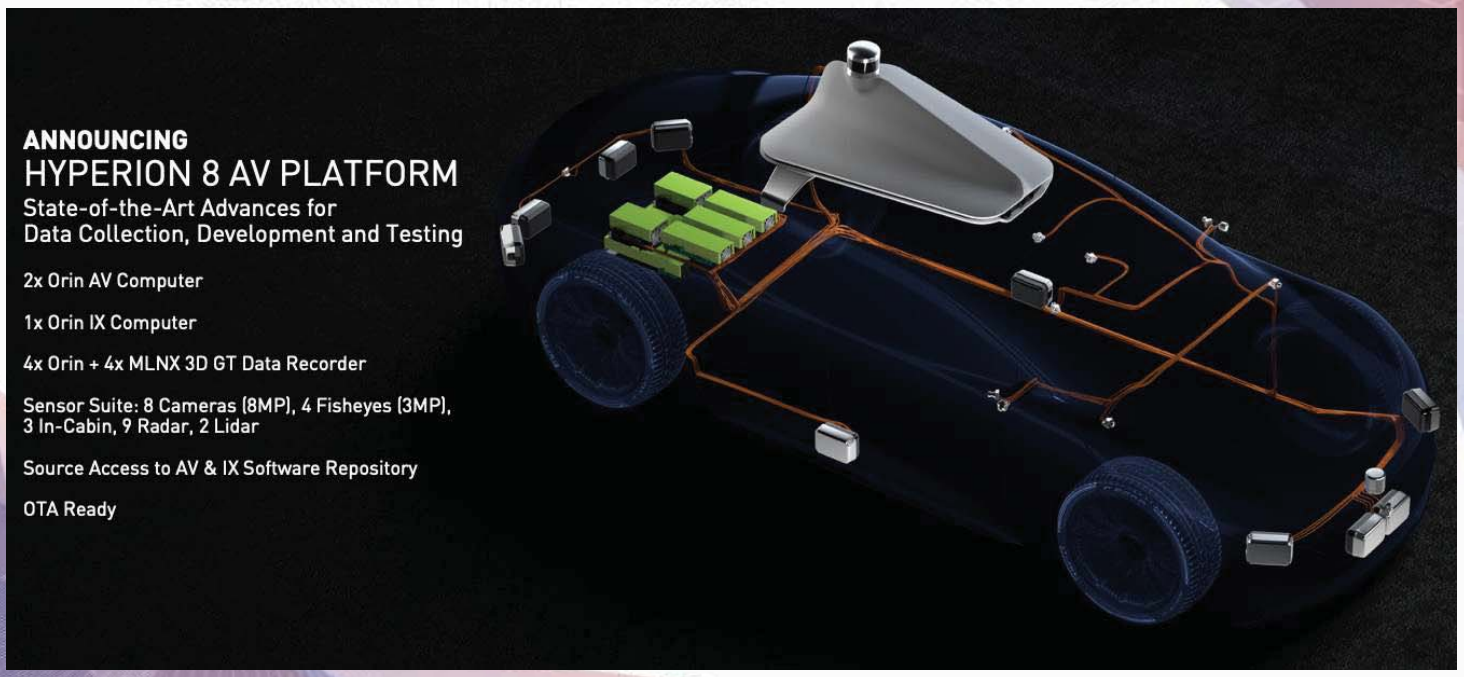


DEVBOT by Roborace

Formula Students 2017 (Füela)



자율주행 AI 개발 플랫폼 (NVIDIA, GTC 2021)



Thank You

Introduction to Deep Learning

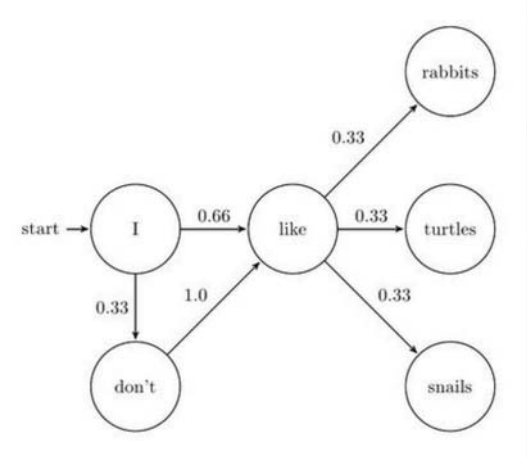
Recurrent Neural Networks

고려대학교 정보보호대학원 인공지능연구실 이상근

KSBi-BIML 2023

Sequence Modeling

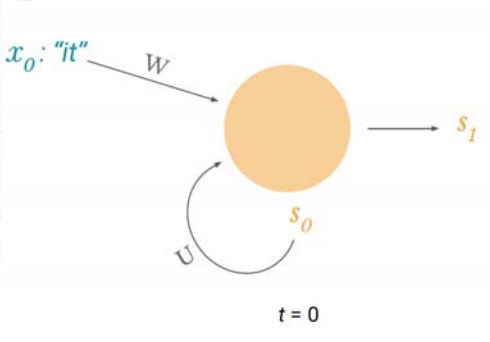
Markov models:



- Markov assumption: each state depends only on the last state
- We cannot model long-term dependencies:
  - In France, I had a good time and I learnt some of the _____ language

RNN

RNN hidden nodes “remember” their previous state:

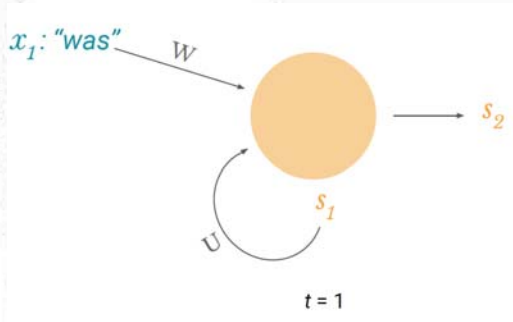


x_0 : vector representing first word
s_0 : cell state at $t = 0$ (some initialization)
s_1 : cell state at $t = 1$

$s_1 = tanh(Wx_0 + Us_0)$

RNN

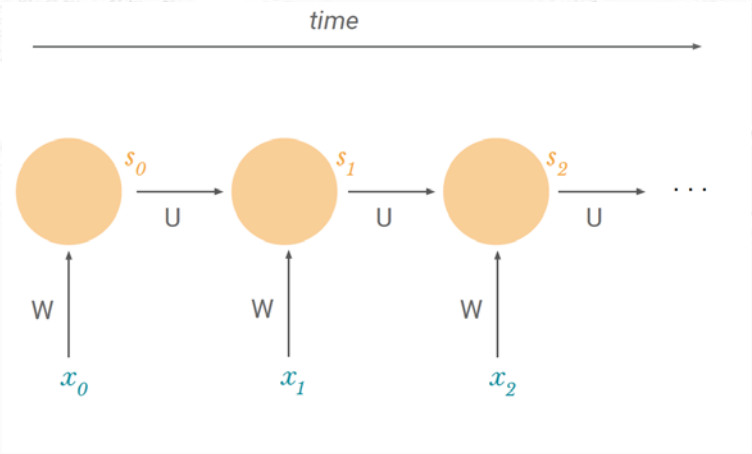
RNN hidden nodes “remember” their previous state:



x_1 : vector representing second word
s_1 : cell state at $t = 1$
s_2 : cell state at $t = 2$

$s_2 = tanh(Wx_1 + Us_1)$

Unfolding RNN Hidden States



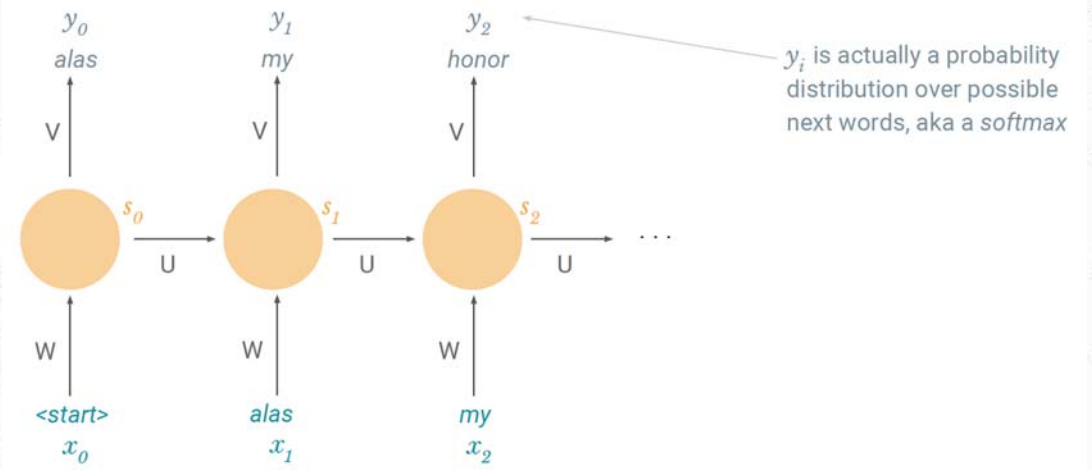
s_n can contain information from all previous states

Language Modeling

all the works of shakespeare → *language model* →

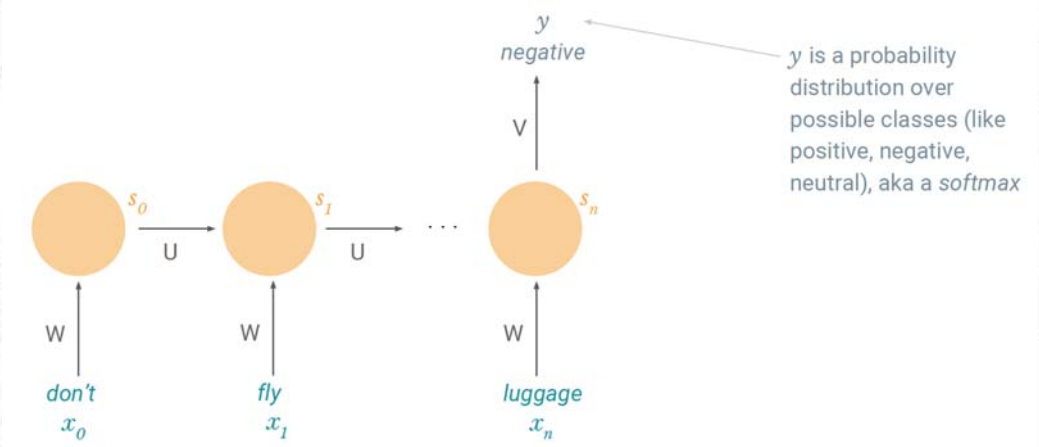
KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

Language Modeling

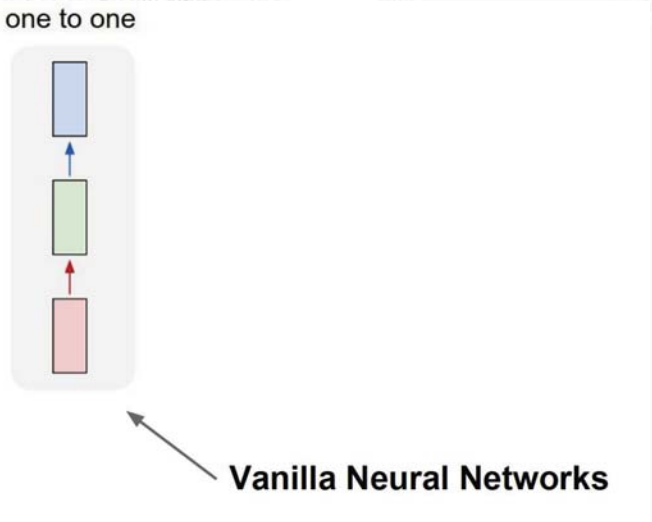


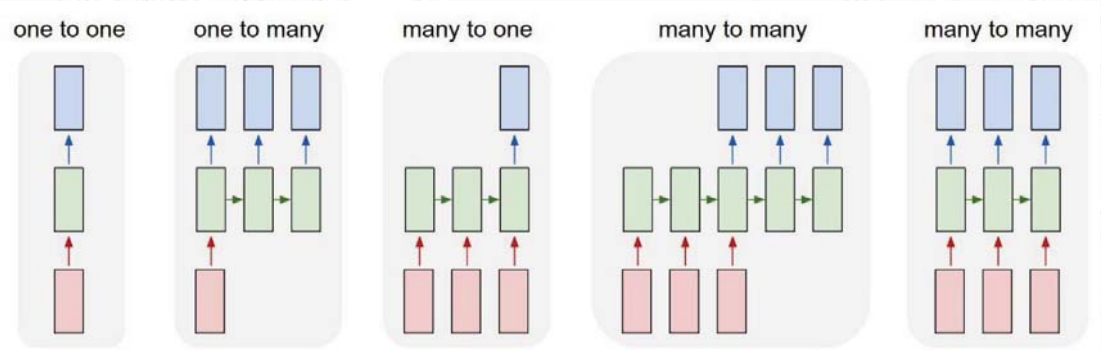
Sentiment Analysis

Sentiment Analysis



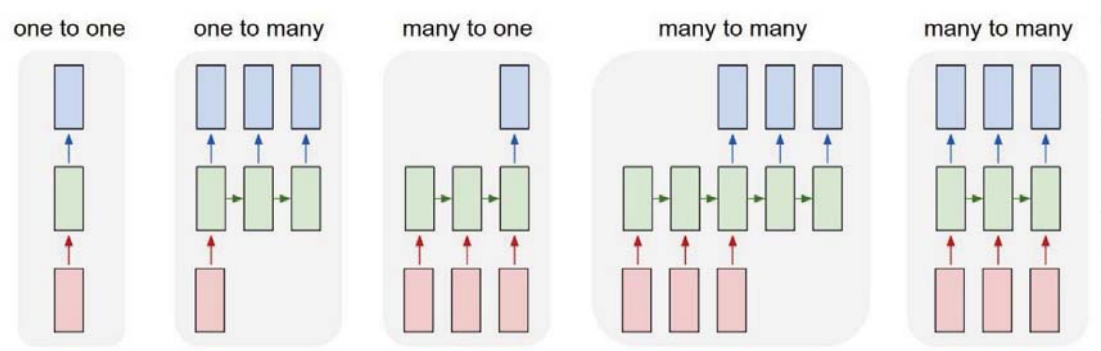
"Vanilla" NN





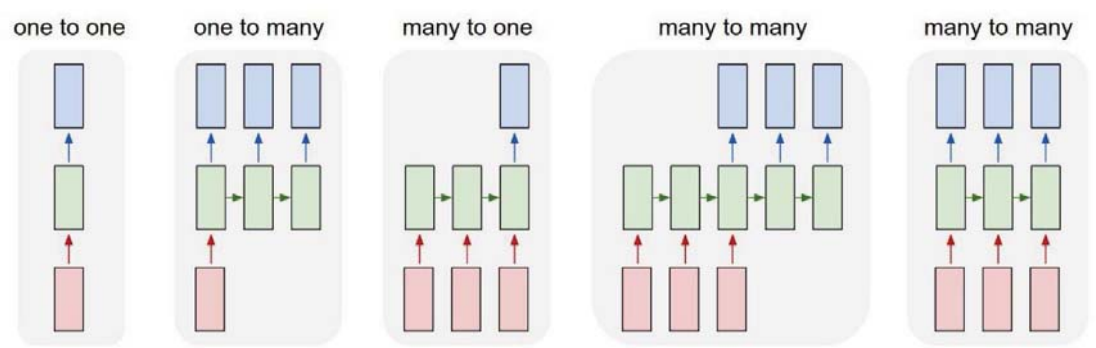
e.g. **Image Captioning**
image -> sequence of words

RNN: Model Sequences



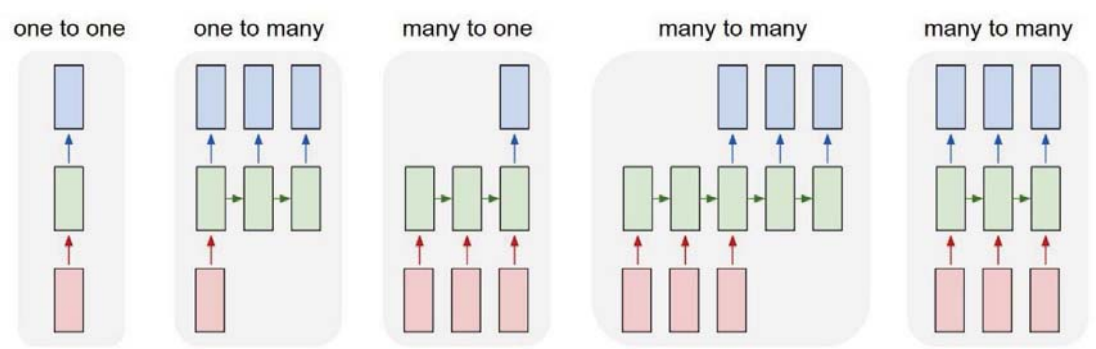
e.g. **Sentiment Classification**
sequence of words -> sentiment

RNN: Model Sequences



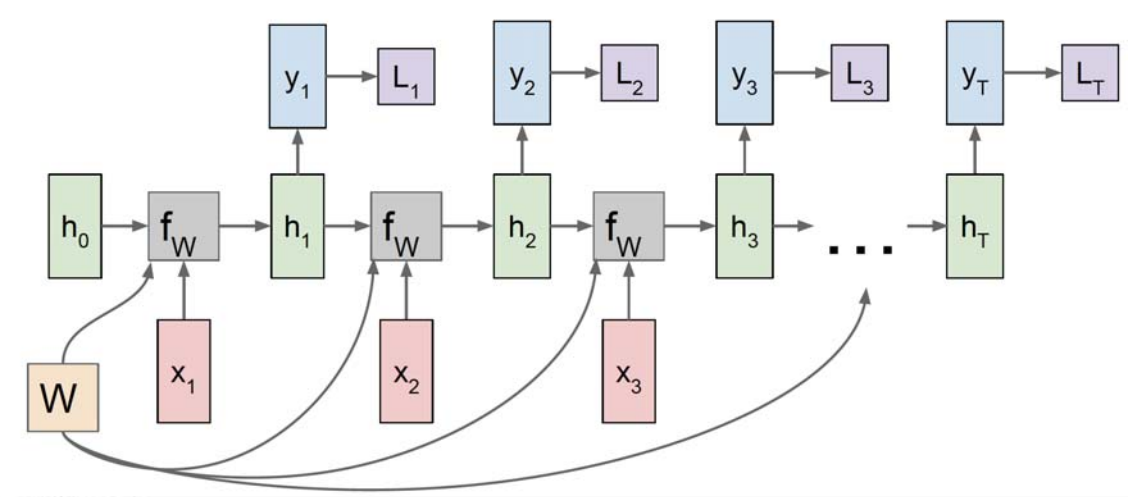
e.g. **Machine Translation**
seq of words -> seq of words

RNN: Model Sequences



e.g. **Video classification on frame level**

RNN Computation Graph: Many-to-Many

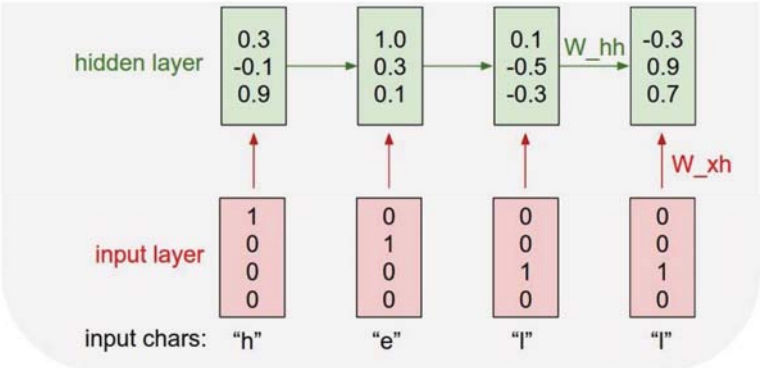


Ex. Character-Level Language Modelling

Vocaburary:
[h, e, l, o]

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

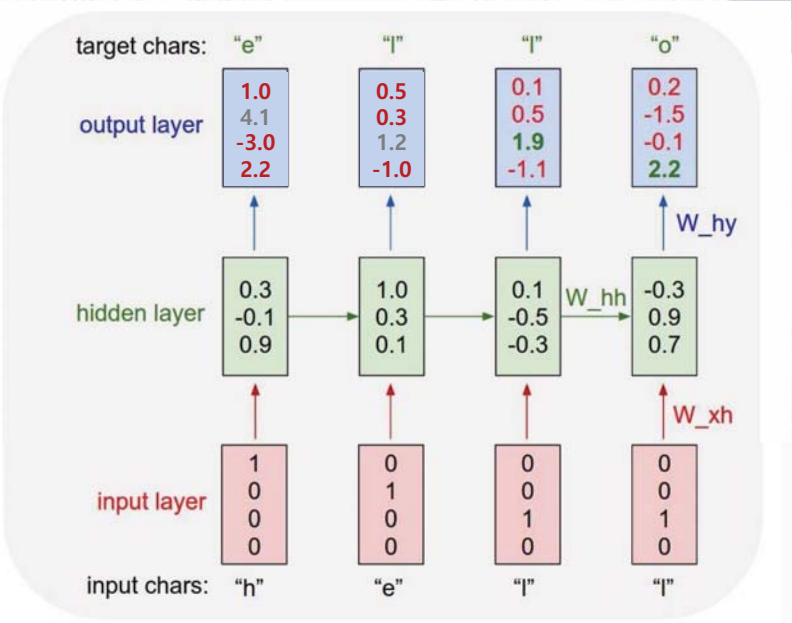
Training
seqex: “hello”



Ex. Character-Level Language Modelling

Vocaburary:
[h, e, l, o]

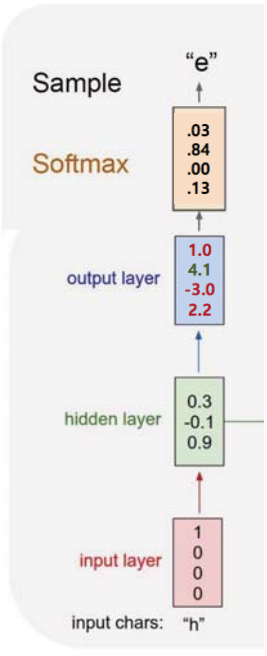
Training sequence exa
"hello"



Ex. Character-Level Language Modelling

Vocaburary:
[h, e, l, o]

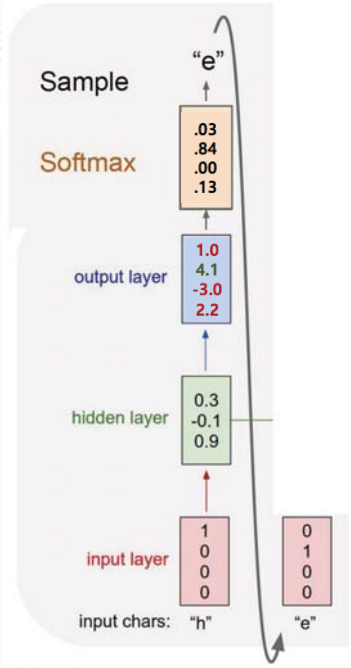
Test-time: sample one character at a time, feeding back to the model



Ex. Character-Level Language Modelling

Vocaburary:
[h, e, l, o]

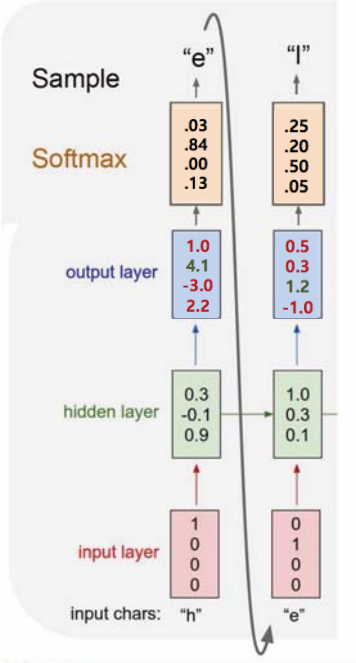
Test-time: sample one character at a time, feeding back to the model



Ex. Character-Level Language Modelling

Vocaburary:
[h, e, l, o]

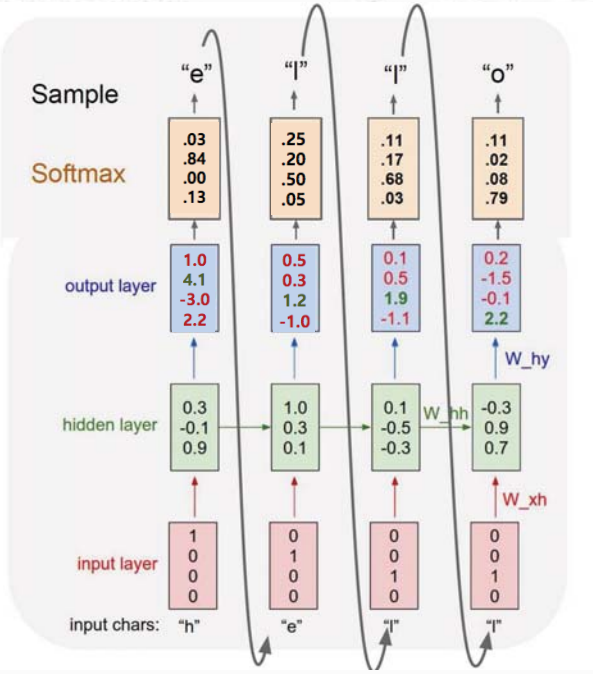
Test-time: sample one character at a time, feeding back to the model



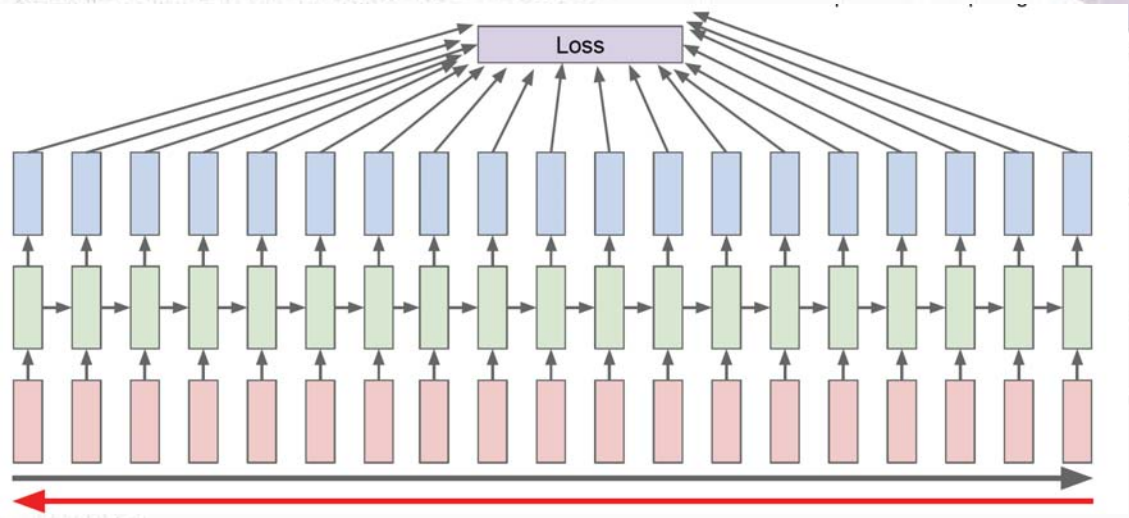
Ex. Character-Level Language Modelling

Vocaburary:
[h, e, l, o]

Test-time: sample one character at a time, feeding back to the model



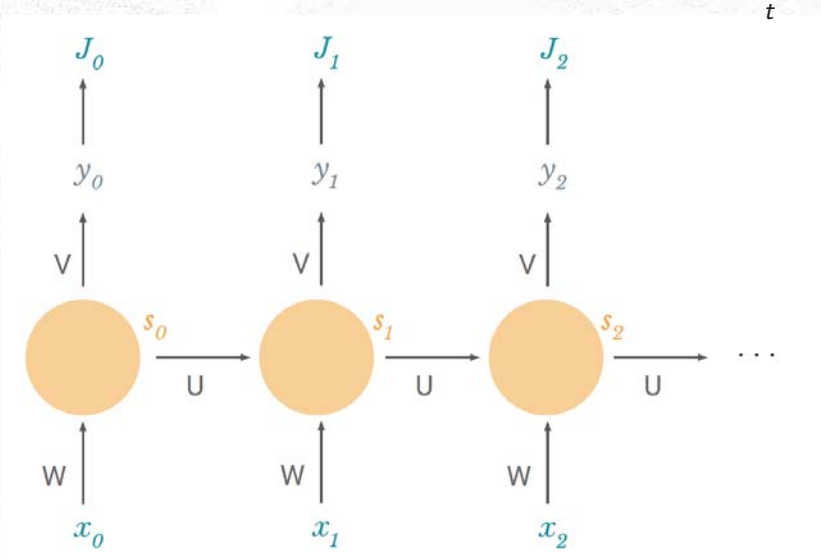
BPTT: Back-Propagation Through Time



Back Propagation

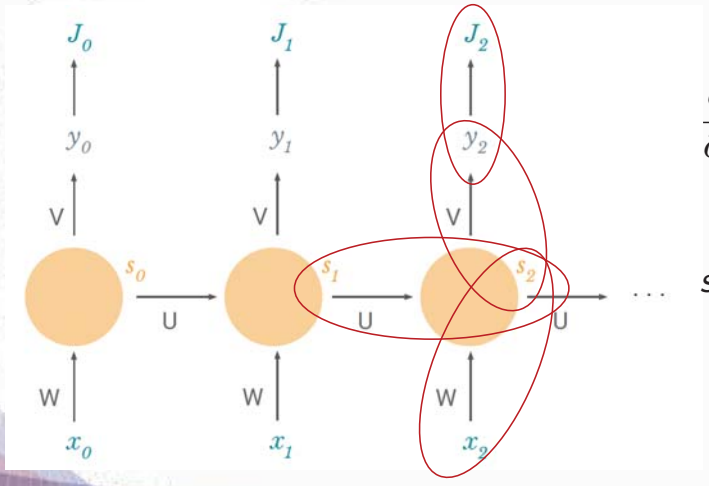
We have loss at each time step:

$$J(w) = \sum_t J_t(w)$$



Back Propagation

$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

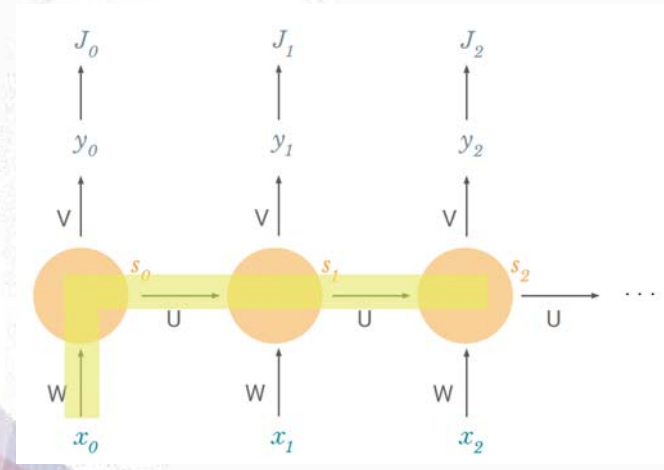


$$\frac{\partial J_2}{\partial W} =$$

$$s_2 = \tanh(Us_1 + Wx_2)$$

s_1 also depends on W !!

Dependency of s_2 on W

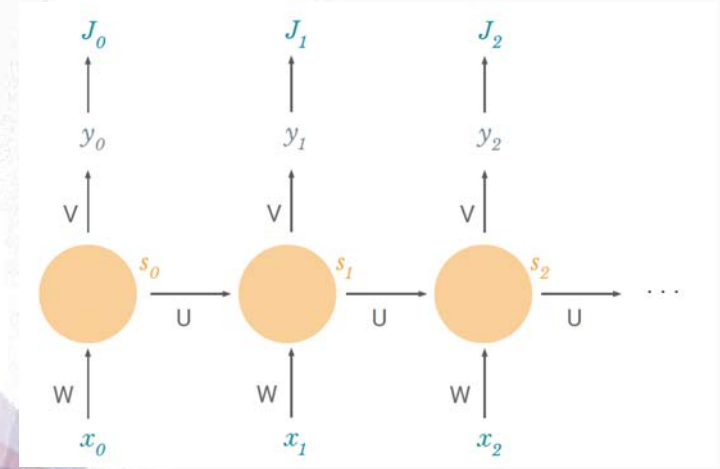


$$\frac{\partial s_2}{\partial W} + \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W} + \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial s_0}\frac{\partial s_0}{\partial W}$$

BPTT

$$\frac{\partial J_2}{\partial W} = \sum_{k=0}^{2} \frac{\partial J_2}{\partial y_2}\frac{\partial y_2}{\partial s_2}\frac{\partial s_2}{\partial s_k}\frac{\partial s_k}{\partial W}$$

$$\frac{\partial s_2}{\partial s_1} \cdots \frac{\partial s_{k+1}}{\partial s_k}$$



Problem: Vanishing Gradient

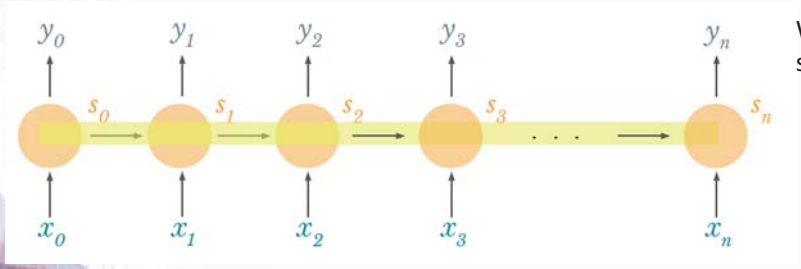
$$\frac{\partial J_n}{\partial W} = \sum_{k=0}^{n} \frac{\partial J_n}{\partial y_n} \frac{\partial y_n}{\partial s_n} \frac{\partial s_n}{\partial s_k} \frac{\partial s_k}{\partial W}$$

$$\frac{\partial s_n}{\partial s_{n-1}} \frac{\partial s_{n-1}}{\partial s_{n-2}} \cdots \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial s_0}$$

$$\frac{\partial s_n}{\partial s_{n-1}} \leftarrow W^T \text{diag}[\phi'(W x_j + U s_{j-1})]$$

W: sampled from *N(0,1)*

$\phi' < 1$



We're multiplying lots of small numbers

Problem: Vanishing Gradient

We're multiplying lots of **small numbers**

→ Errors due to further back timesteps have increasingly **smaller gradients**

→ Parameters become biased to capture **short-term dependencies**

Solutions:

- Use special units instead of hidden nodes
- E.g. LSTM (Long Short-Term Memory)

Image Captioning

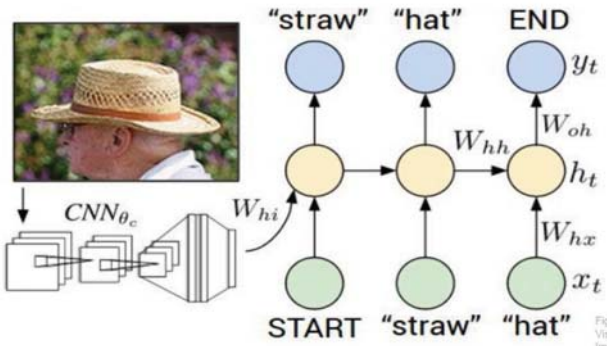


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015. Reproduced for educational purposes.

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Image Captioning

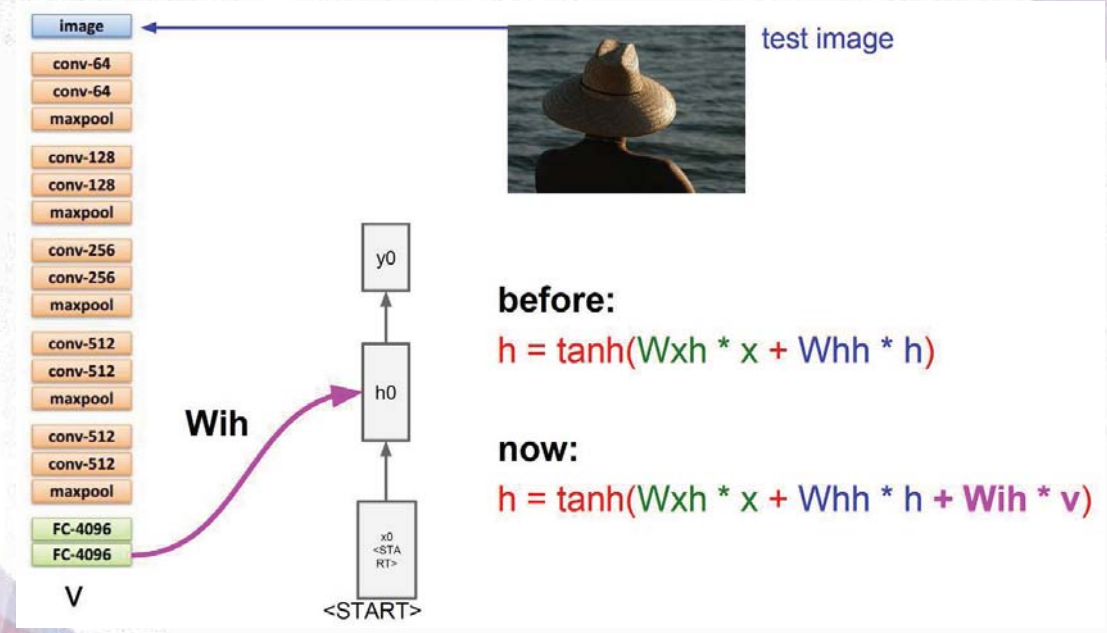


Image Captioning

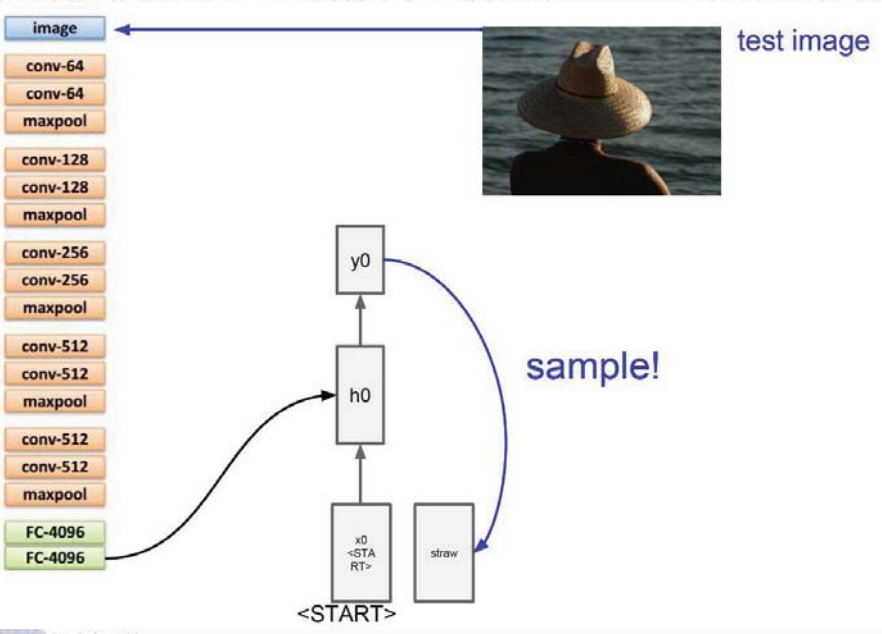


Image Captioning

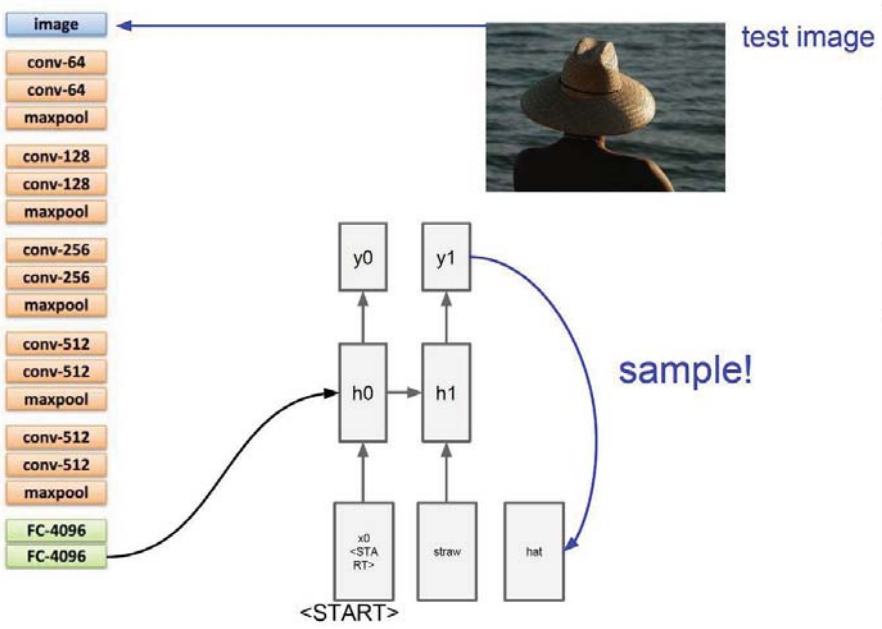


Image Captioning

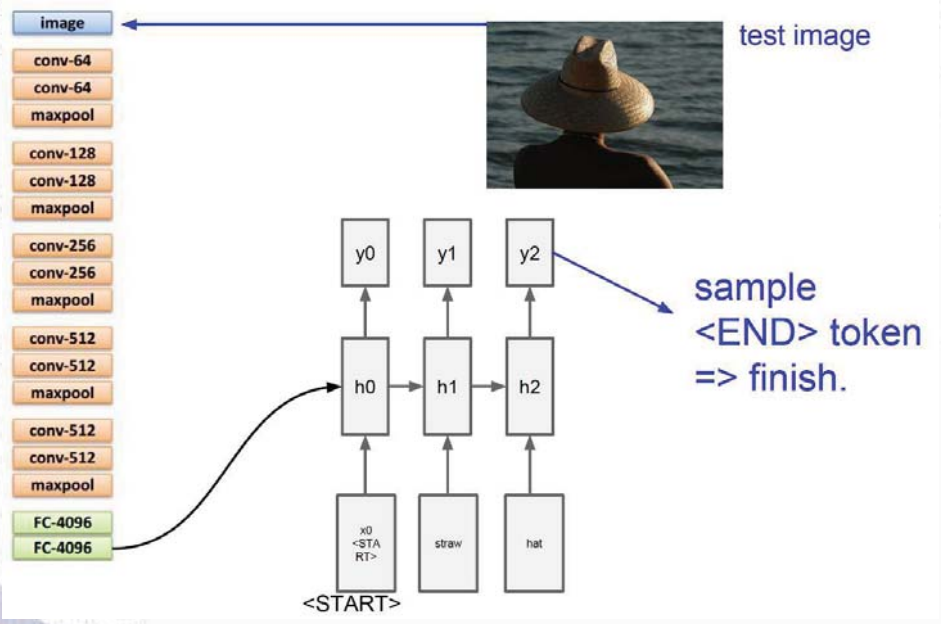


Image Captioning: Examples

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

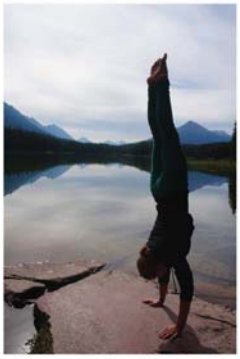
Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

Image Captioning: Failures

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Visual Question Answering

Q: What endangered animal is featured on the truck?

A: A bald eagle.
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?

A: Onto 24 ¾ Rd.
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: When was the picture taken?

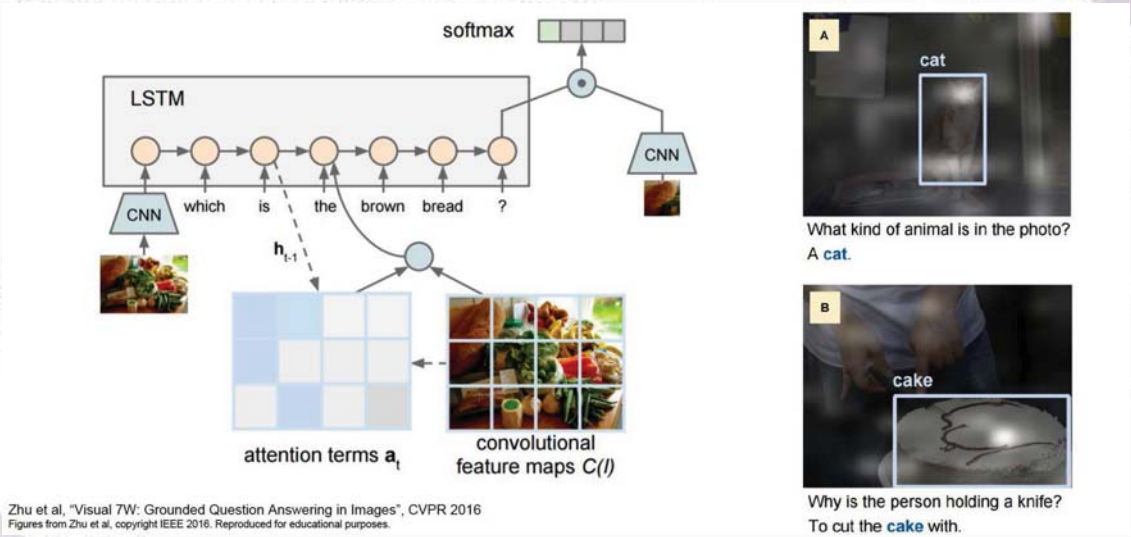
A: During a wedding.
A: During a bar mitzvah.
A: During a funeral.
A: During a Sunday church service

Q: Who is under the umbrella?

A: Two women.
A: A child.
A: An old man.
A: A husband and a wife.

Agrawal et al, "VQA: Visual Question Answering", ICCV 2015
Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016
Figure from Zhu et al, copyright IEEE 2016. Reproduced for educational purposes.

Visual QA: RNN + Attention



Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016
Figures from Zhu et al, copyright IEEE 2016. Reproduced for educational purposes.

Long Short Term Memory

Multilayer RNNs

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$h \in \mathbb{R}^n$ $\quad W^l \ [n \times 2n]$

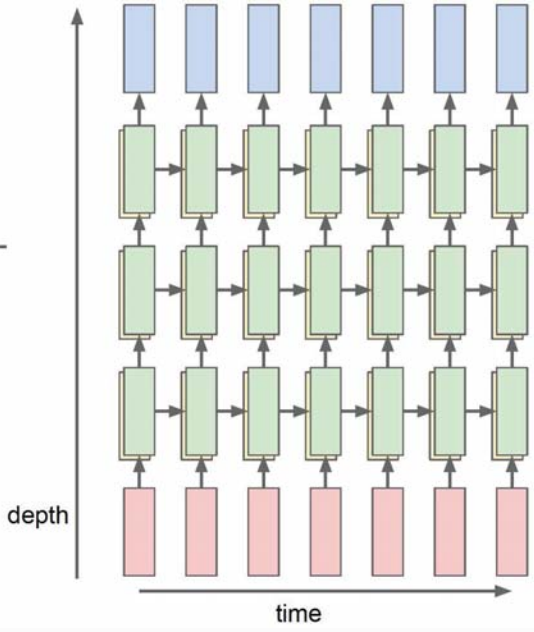
LSTM:

$W^l \ [4n \times 2n]$

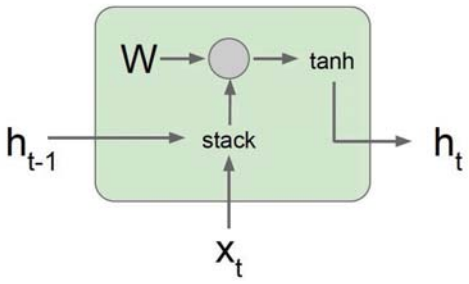
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



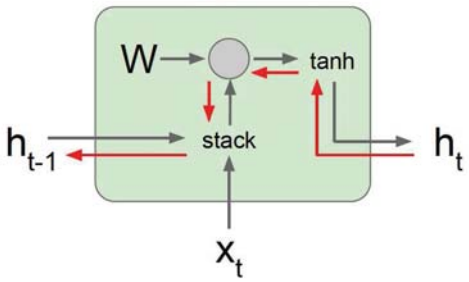
Vanilla RNN Gradient Flow



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

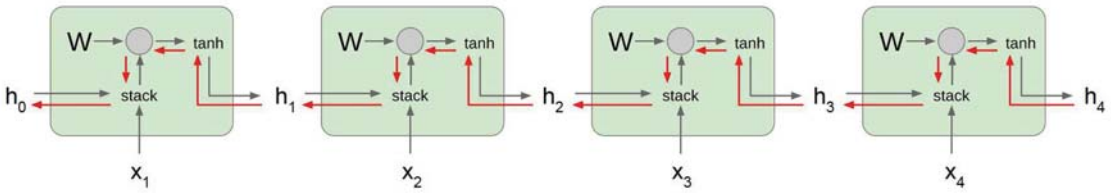
Vanilla RNN Gradient Flow

Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Vanilla RNN Gradient Flow



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

LSTM [Hochreiter et al., 1997]

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

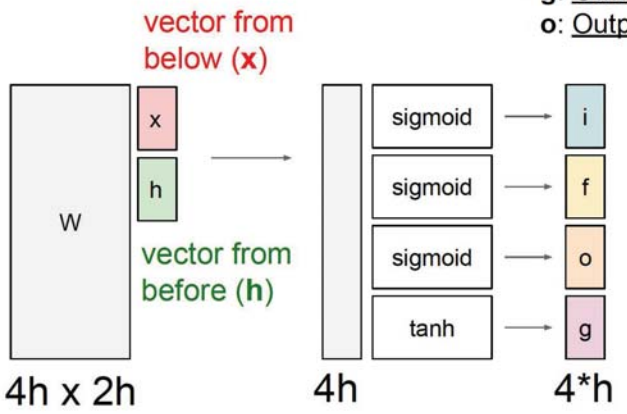
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM [Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell

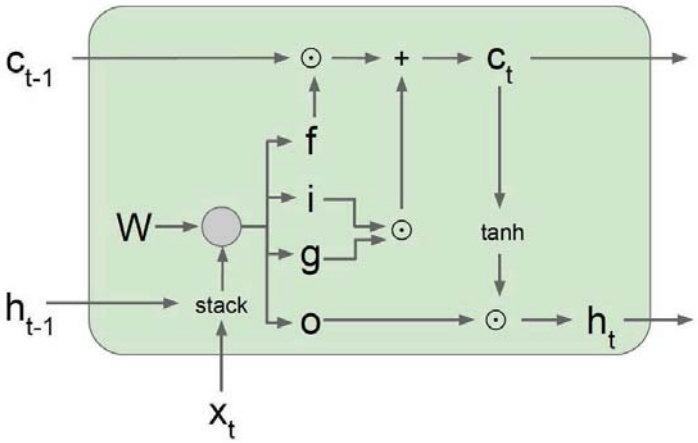


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM: Gradient Flow

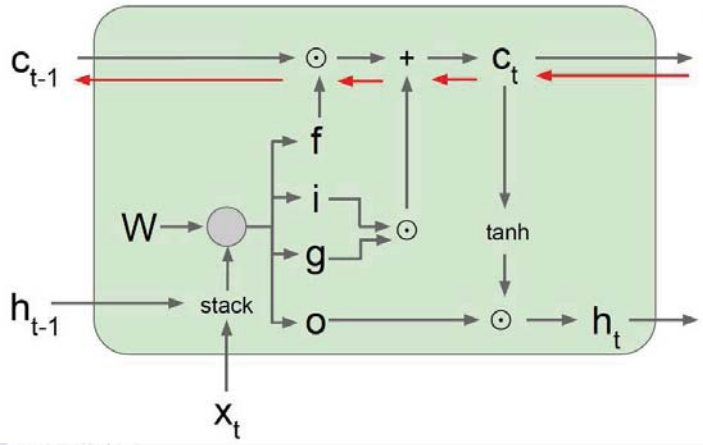


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM: Gradient Flow



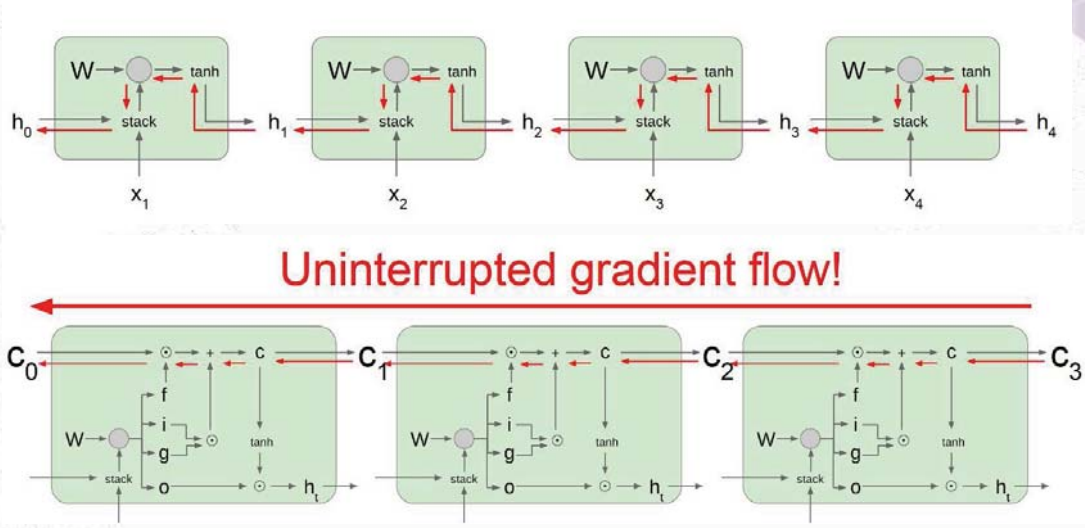
Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Vanilla vs. LSTM: Gradient Flow

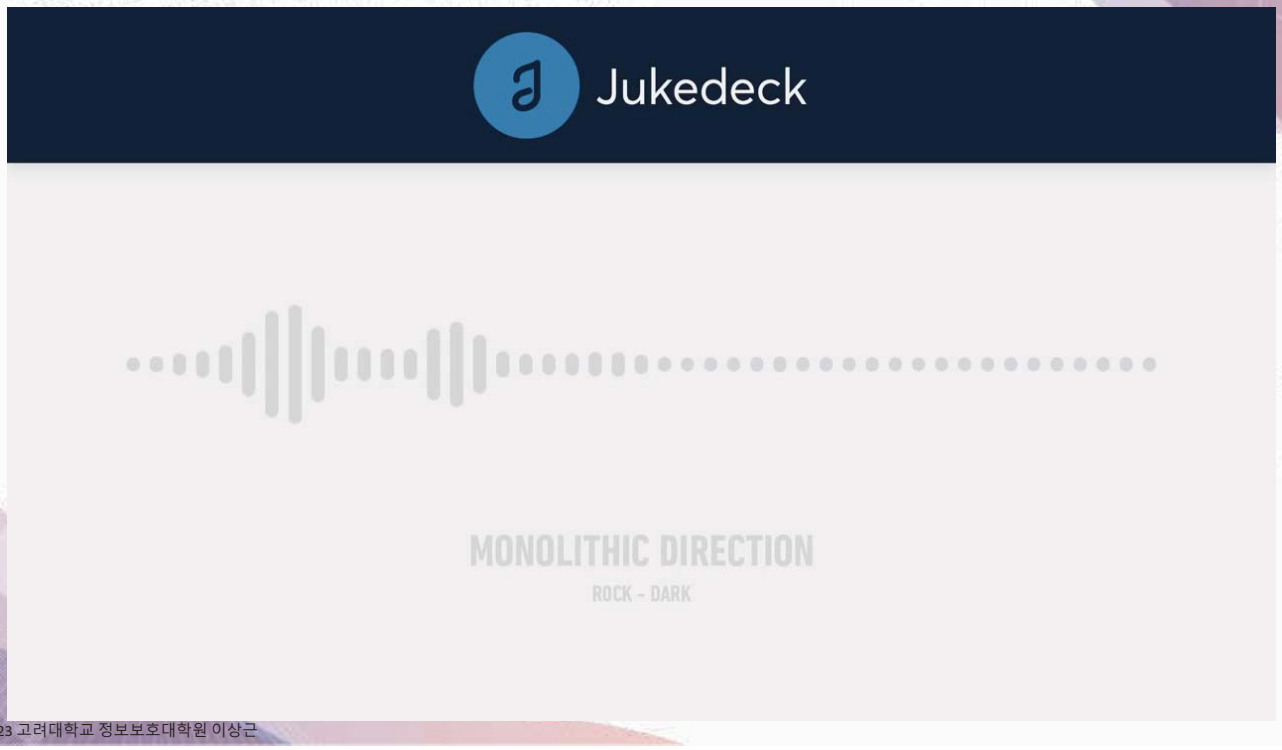


Gated Recurrent Unit

GRU [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
\tilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{aligned}
$$

Jukedeck



Thank You

Introduction to Deep Learning

Generative Adversarial Net

고려대학교 정보보호대학원 인공지능연구실 이상근

KSBi-BIML 2023

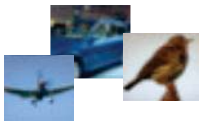
Machine Learning Models

Discriminative

e.g. Supervised learning

- Data: (x: data, y: label)
- Goal: find a function x → y

Generative



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

- Goal: find $p_{model}(x)$ similar to $p_{data}(x)$

Why Generative Models?

- Realistic samples for artworks, colorization, etc.

- Learn latent representation that can be used as general features
- To understand the structure of data
- Generation of time-series data can be used for simulation & planning

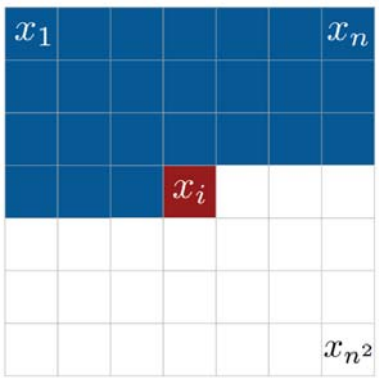
PixelRNN [van der Oord et al., 2016]

- Fully visible belief network to explicitly model the density function
- Chain rule to decompose the likelihood of an image:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

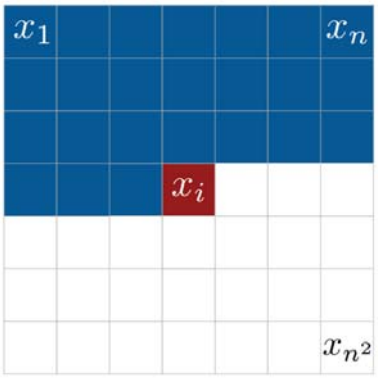
Likelihood of image x

Probability of i'th pixel value given all previous pixels



PixelRNN [van der Oord et al., 2016]

- Maximize the likelihood function w.r.t. training data
  - Need to define ordering of pixels
  - Complex distribution over pixel values → represented as an NN
- Color dependence: $p(x_{i,R}|\mathbf{x}_{<i})p(x_{i,G}|\mathbf{x}_{<i}, x_{i,R})p(x_{i,B}|\mathbf{x}_{<i}, x_{i,R}, x_{i,G})$



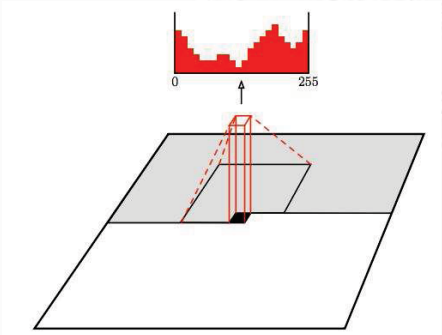
- Use LSTM-RNN to model dependency on previous pixels
- Image generation is sequential (slow)

PixelCNN [van der Oord et al., 2016]

- Dependency on previous pixels: modeled using a CNN over context region
- Training: MLE

$$p(x) = \prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1})$$

- Training is faster than PixelRNN
  - Convolutions can be parallelized
- Generation is still sequential (slow)



PixelCNN Samples

32x32 CIFAR-10

32x32 ImageNet

PixelRNN / PixelCNN

Pros:

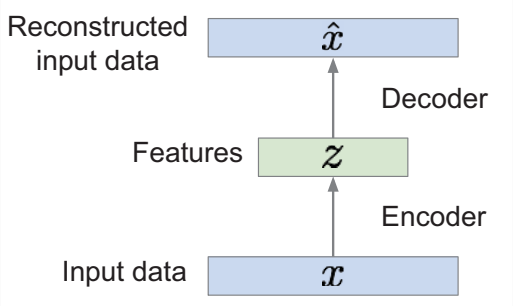
- Explicit probability model
  - Can compute $p_{model}(x)$
  - Explicit likelihood function: good evaluation metric
  - Good samples

Cons:

- Sequential generation can be very slow

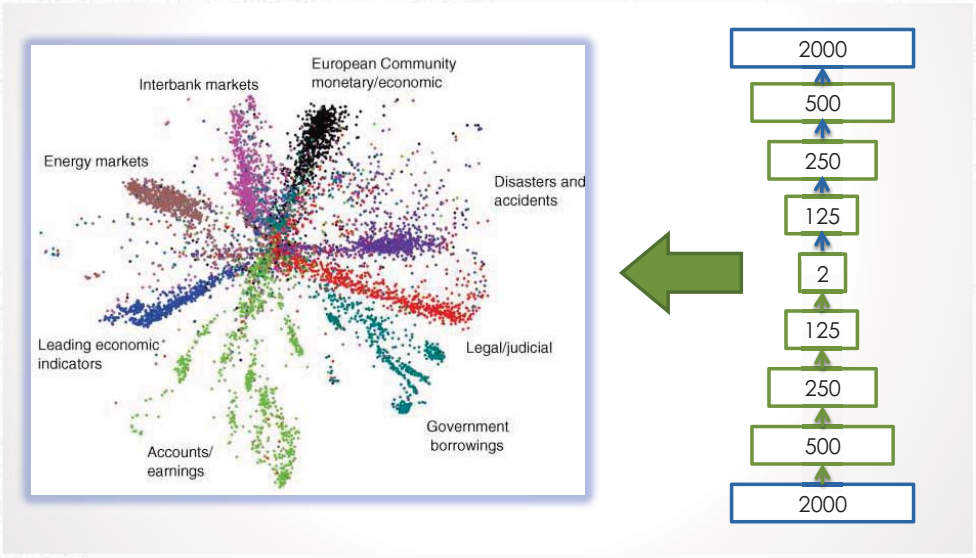
Autoencoder

- Find a lower-dimensional feature representation of the data
- Which can be used to reconstruct the original data (unsupervised)



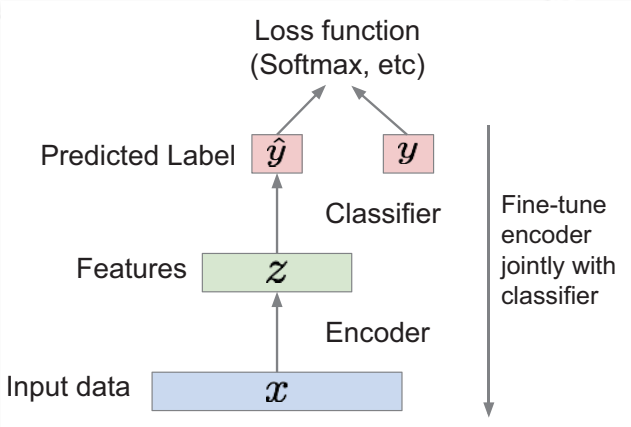
AE: Semantic Hashing

Salakhutdinov & Hinton, 2009



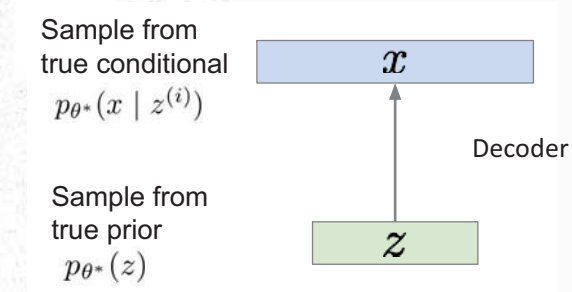
AE: Initialization of a Classifier

- Learn AE with large, unlabeled data (unsupervised)
- Adjust weights with labeled data for classification (supervised)



Variational Autoencoder (VAE)

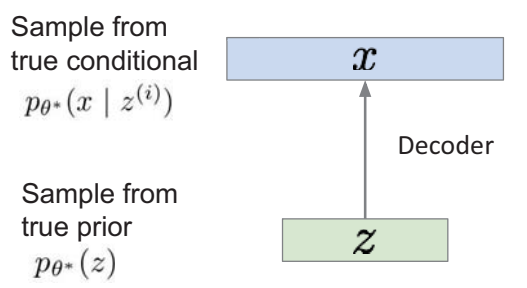
- Can we use AEs for generating new samples?
- VAE: probabilistic variant of AE [Kingma & Welling, 2014]



- Assumption:
  - Training data $\{x^{(i)}\}_{i=1}^{N}$ are generated from latent representations **z**
  - E.g. x = faces, z = different facial expressions

Variational Autoencoder (VAE)

- Can we use AEs for generating new samples?
- VAE: probabilistic variant of AE [Kingma & Welling, 2014]



- Prior p(z)
  - Choose a simple model
  - E.g. Gaussian
- Conditional p(x|z)
  - Complex model
  - Represented with a decoder NN

VAE

- PixelCNN: uses explicit, tractable density function (can optimize directly)

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

- VAE: uses explicit, intractable density function (cannot optimize directly)

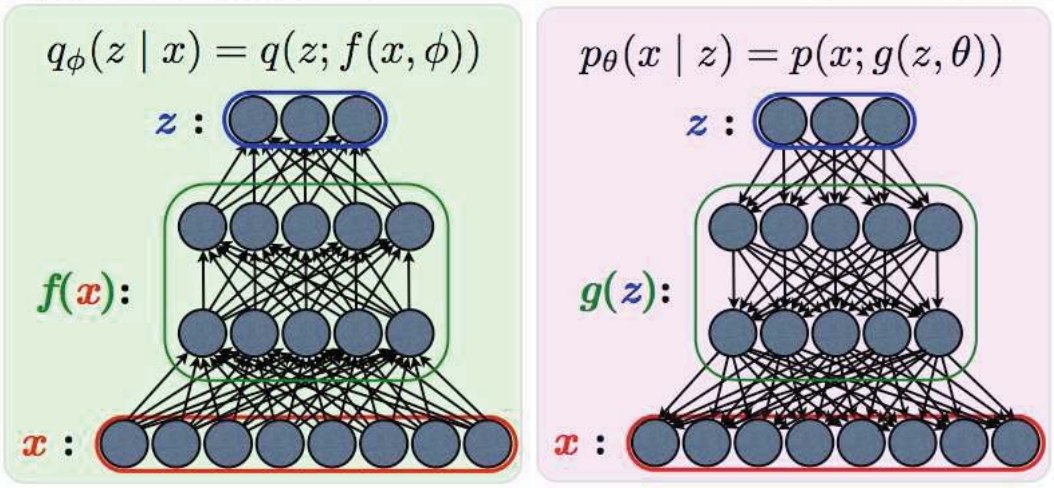
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

→ Derive & optimize a lower bound (ELBO) on the likelihood

VAE: Encoder & Decoder

Encoder

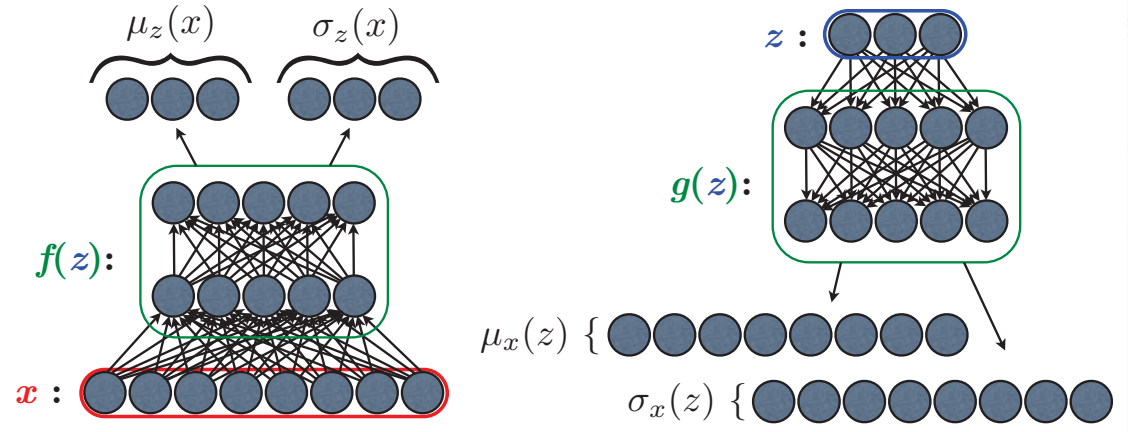
Decoder



VAE: Reparametrization Trick

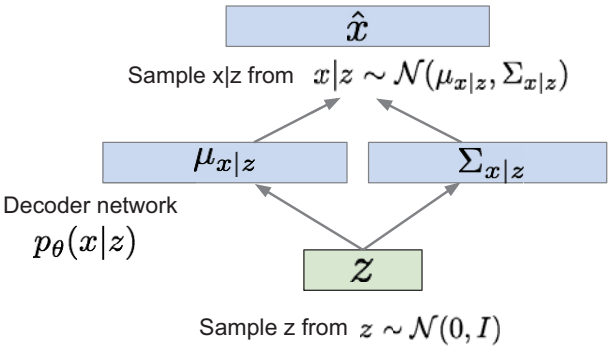
Parametrize z as $z = \mu_z(x) + \sigma_z(x)\epsilon_z$ where $\epsilon_z = \mathcal{N}(0, 1)$

(optional) Parametrize x a $x = \mu_x(z) + \sigma_x(z)\epsilon_x$ where $\epsilon_x = \mathcal{N}(0, 1)$

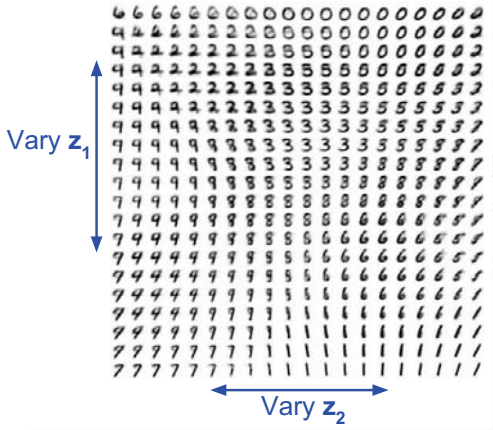


VAE: Generation

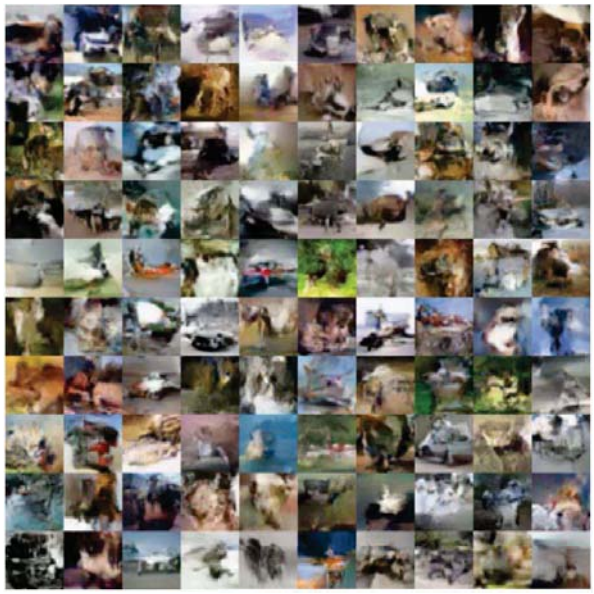
Sample z from a prior, and use the decoder NN



Data manifold for 2-d **z**



VAE: Samples



32x32 CIFAR-10

Labeled Faces in the Wild

VAE

Pros:

- Bayesian learning / inference framework
- Can use q(z|x) for feature representation

Cons:

- Optimize a lower bound of the likelihood
- Samples are rather blurry and in low quality

GAN: Motivation

- PixelCNN: uses explicit, tractable ~~density function~~ (can optimize directly)

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, ..., x_{i-1})$$

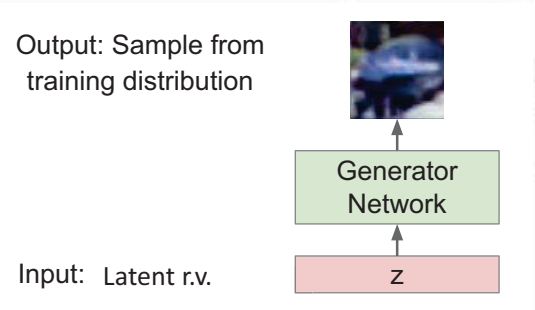
- VAE: uses explicit, intractable ~~density function~~ (cannot optimize directly)

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

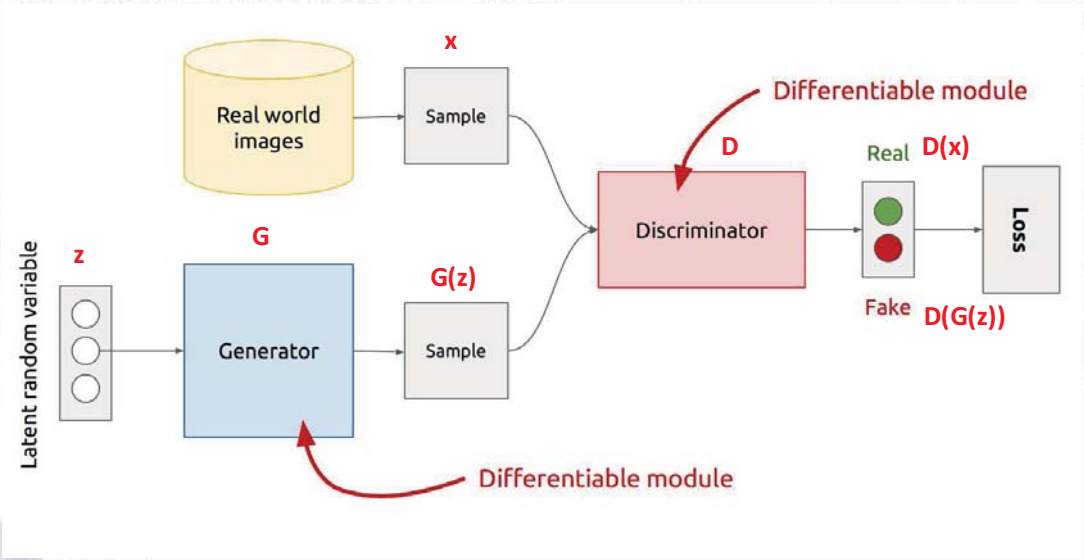
- GAN: can we just sample, without explicit modeling of density?

GAN

- GAN: can we just sample, without explicit modeling of density?
  - Need to sample from a complex, high-dim data distribution. How?
- Work-around:
  - Sample from a sample, latent distribution
  - Learn a transformation:
    - latent → data distribution
    - Represented with an NN



GAN Architecture



GAN: Training

$$D_{\theta_d}(x) \in (0, 1)$$

Formulated as a two-player minimax game:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data x

Discriminator output for generated fake data G(z)

- Discriminator player: maximize the obj w.r.t. θ_d s.t.

$$D_{\theta_d}(x) \approx 1 \qquad D_{\theta_d}(G_{\theta_g}(z)) \approx 0$$

- Generator player: minimize the obj w.r.t. θ_g s.t.

$$D_{\theta_d}(G_{\theta_g}(z)) \approx 1$$

GAN: Training

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Optimize via block-coordinate descent:

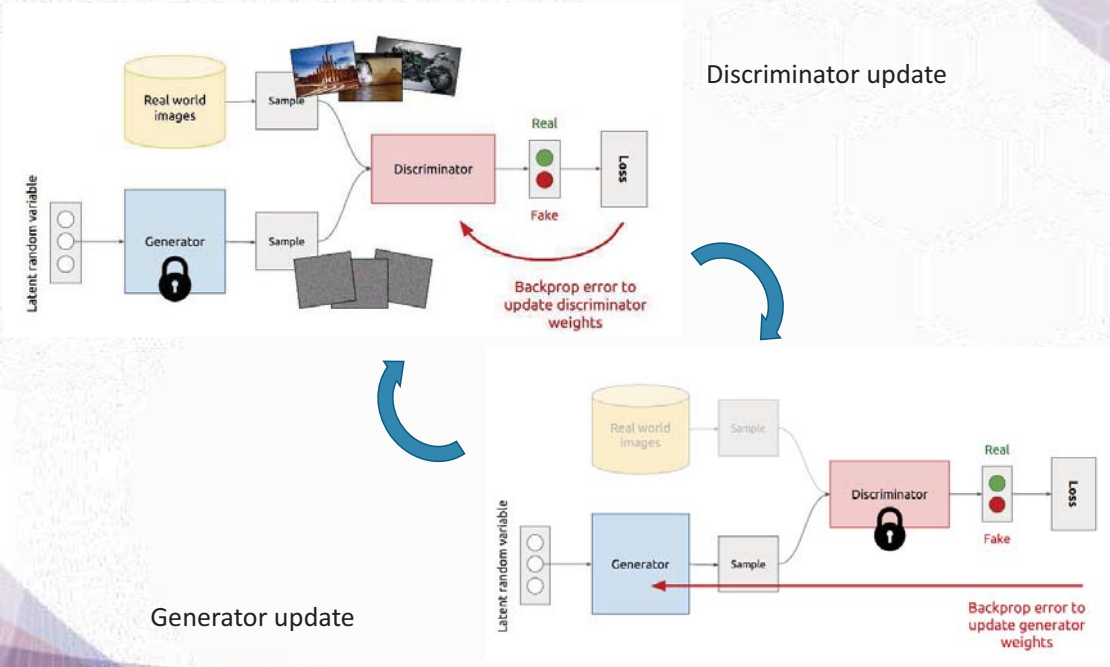
1. Gradient ascent on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient descent on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

GAN: Alternating Training



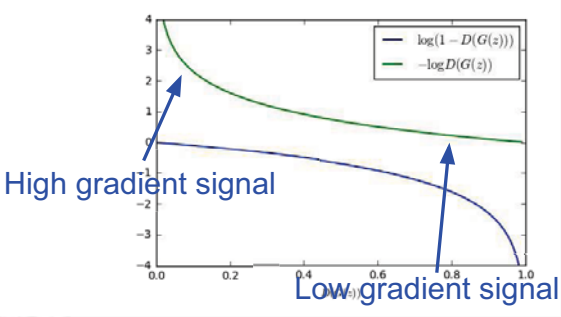
GAN: Vanishing Gradient

Generator update:

$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

$$\nabla_a \log\big(1-\sigma(a)\big) = \frac{-\nabla_a\,\sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\,\big(1-\sigma(a)\big)}{1-\sigma(a)} = -\sigma(a) = -D\big(G(z)\big)$$

Gradient goes to 0 if D is confident, i.e., D(G(z)) ~ 0

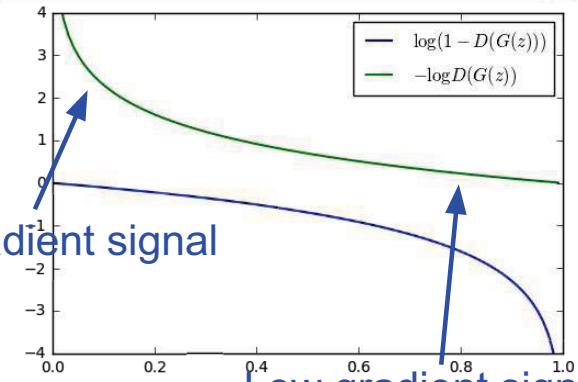


GAN: Generator Update (Modified)

Generator update:

~~$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$~~

$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



High gradient signal

Low gradient signal

GAN Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator updates

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

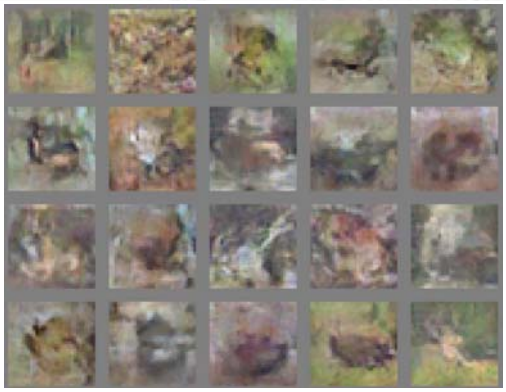
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

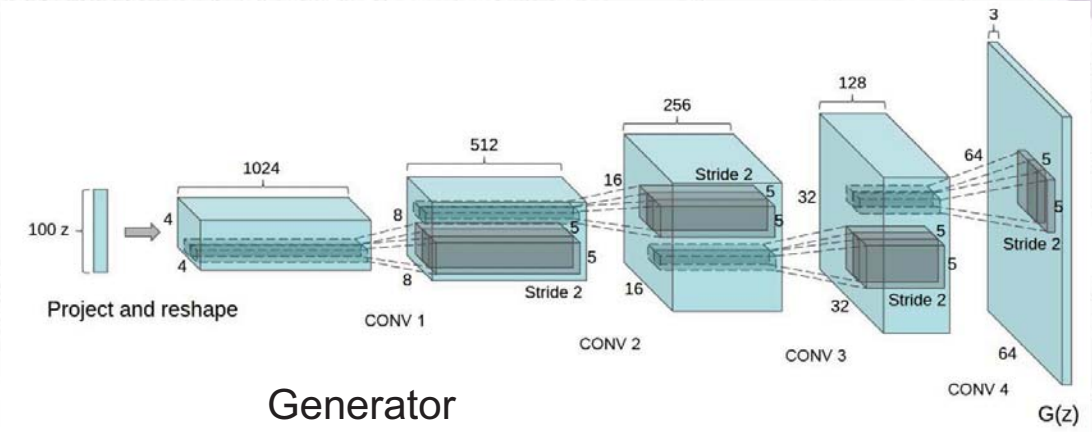
GAN Generated Samples

Faces



CIFAR-10

Deep Convolutional GAN (DCGAN)



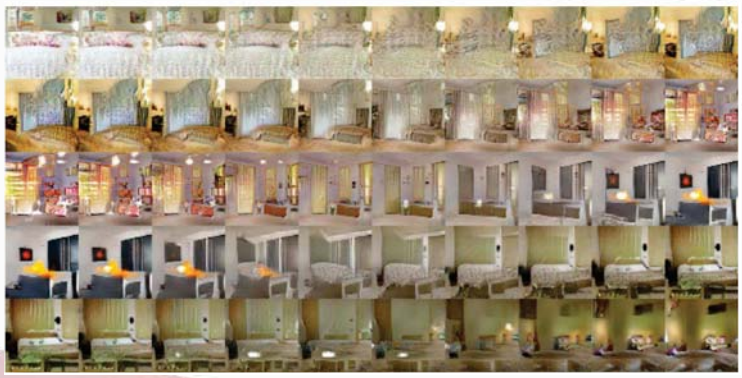
Generator

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

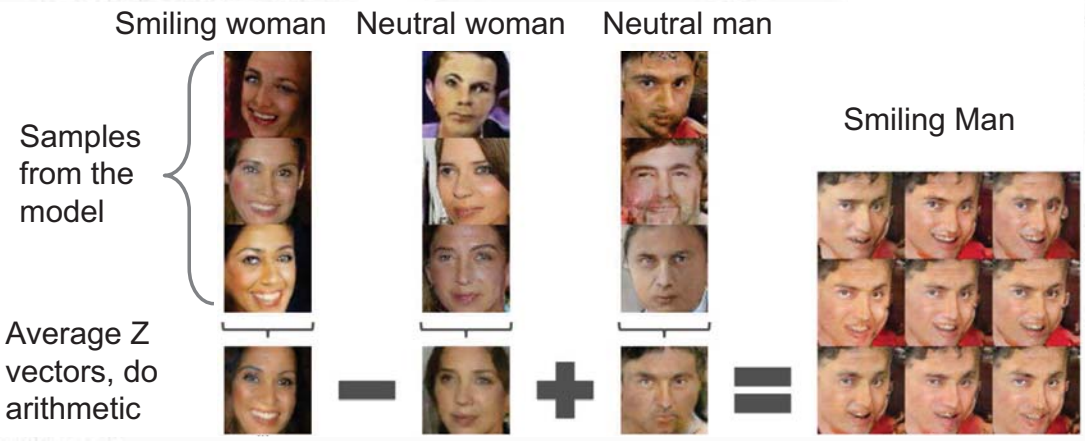
DCGAN Generated Samples

Bedroom images

Interpolation in z space



GAN: Vector-Space Embedding



GAN: Vector-Space Embedding

GAN

Pros

- State-of-the-art sample generation

Cons

- Convergence issues in training optimization
- Cannot handle inference regarding p(x), p(z|x)

Thank You

Introduction to Deep Learning

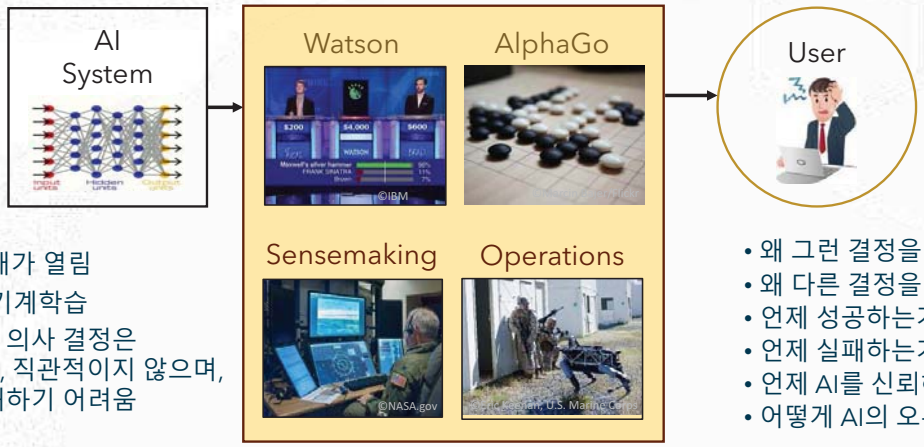
XAI (eXplainable AI)

고려대학교 정보보호대학원 인공지능연구실 이상근

KSBi-BIML 2023

XAI (eXplainable AI)

- 미국 방위고등연구계획국(DARPA)의 연구 프로그램 (2016~2021)
  - https://www.darpa.mil/program/explainable-artificial-intelligence

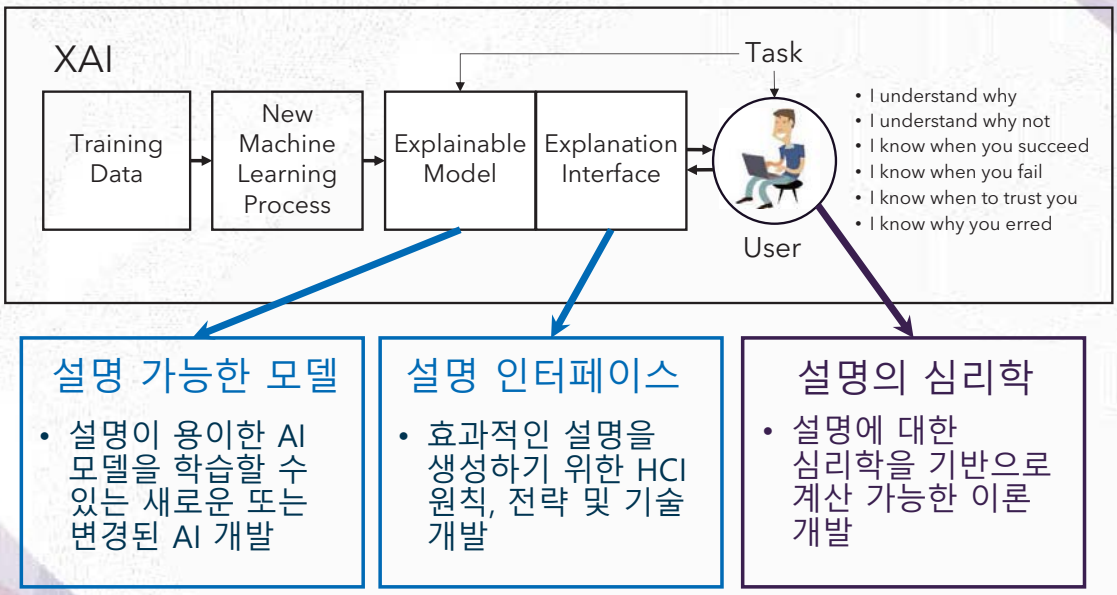


- AI 응용 시대가 열림
- 핵심기술: 기계학습
- 기계학습의 의사 결정은 불투명하고, 직관적이지 않으며, 사람이 이해하기 어려움

- 왜 그런 결정을 내렸는가?
- 왜 다른 결정을 내리지 않았는가?
- 언제 성공하는가?
- 언제 실패하는가?
- 언제 AI를 신뢰해도 괜찮은가?
- 어떻게 AI의 오류를 보정할 수 있는가?

- AI의 의사 결정과 행동의 이유를 설명하지 못하면 AI의 효과적 적용이 제한될 수밖에 없음
- 설명가능한 AI는 AI를 이해하고, 적정 수준까지 신뢰하며, 효과적으로 운영하기 위해 필수적임

XAI의 구성 요소

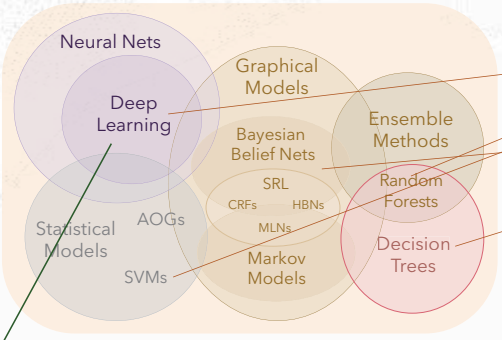


XAI 모델 (1): 설명가능한 새로운 딥러닝 모델

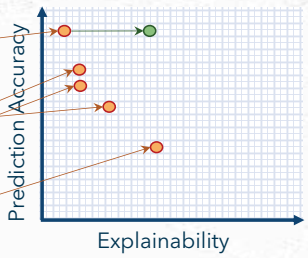
새로운 방법론

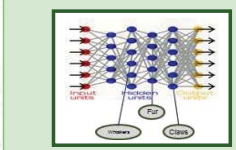
더 나은 설명력과 예측 성능을 갖는 모델의 개발

현재 AI 방법론



설명가능성





Deep Explanation

Modified deep learning techniques to learn explainable features

Example Explanations

This is a Kentucky warbler because this is a yellow bird with a black cheek patch and a black crown.

This is a pied billed grebe because this is a brown bird with a long neck and a large beak.

Hendricks et al., Generating Visual Explanations, arXiv, 2016 (UC Berkeley)

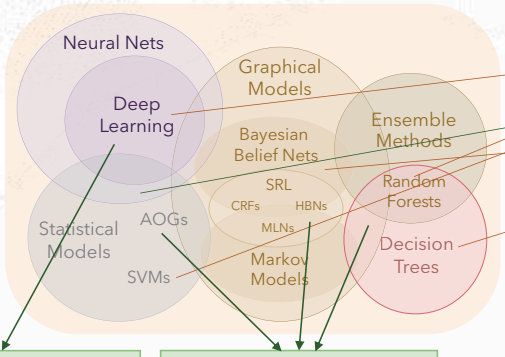
- 새의 종류를 85%의 정확도로 예측
- 이미지의 설명과 종의 정의와 부합하는 부분을 설명

XAI 모델 (2): 설명가능한 향상된 기계학습 모델

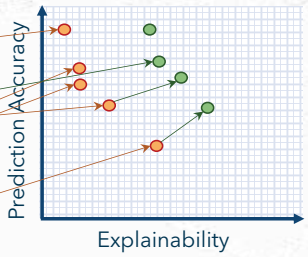
새로운 방법론

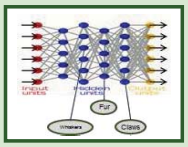
더 나은 설명력과
예측 성능을 갖는
모델의 개발

현재 AI 방법론

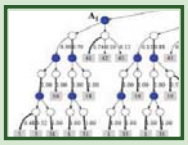


설명가능성





Deep Explanation
Modified deep learning techniques to learn explainable features



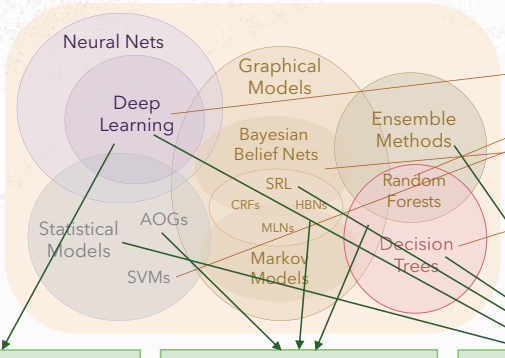
Interpretable Models
Techniques to learn more structured, interpretable, causal models

XAI 모델 (3): 설명 추출 기법

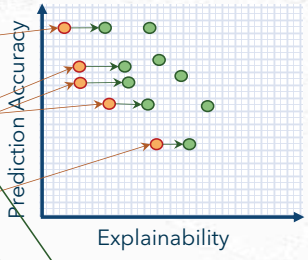
새로운 방법론

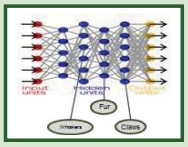
더 나은 설명력과
예측 성능을 갖는
모델의 개발

현재 AI 방법론

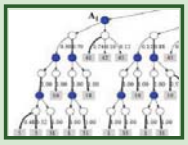


설명가능성

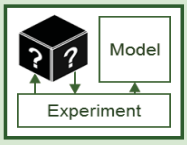




Deep Explanation
Modified deep learning techniques to learn explainable features



Interpretable Models
Techniques to learn more structured, interpretable, causal models



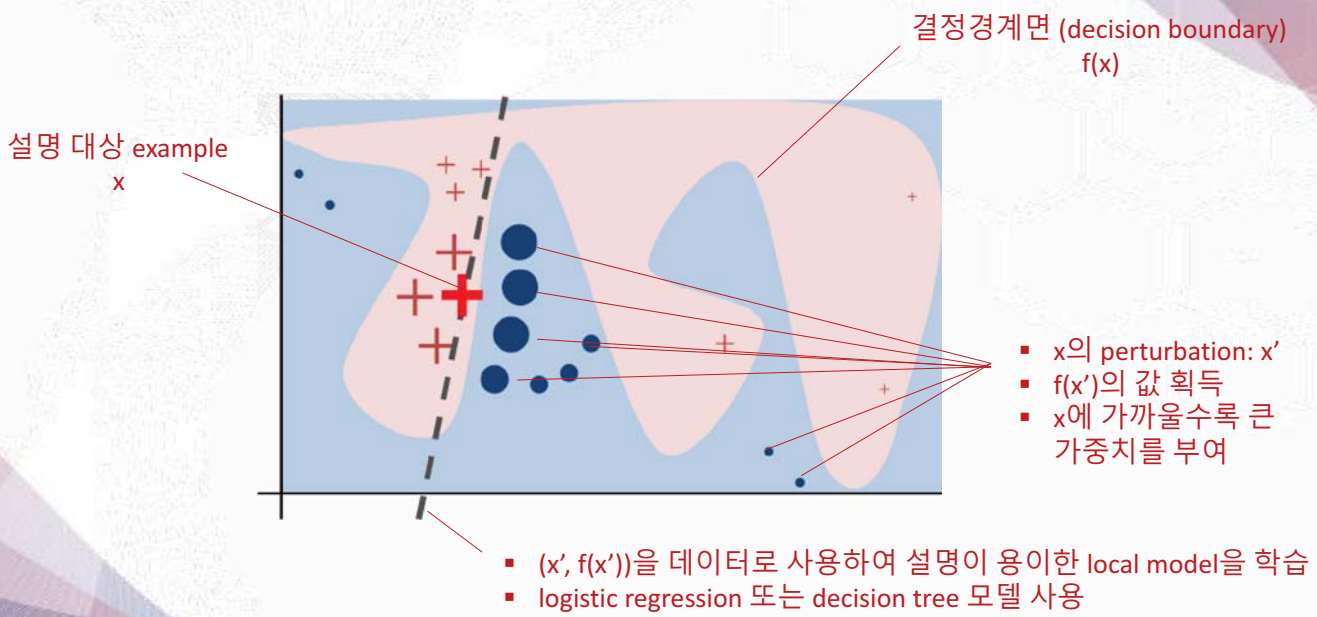
Model Induction
Techniques to infer an explainable model from any model as a black box

Model Induction Methods

- Perturbation-Based Methods
  - 입력의 변화에 따른 예측값의 변화로 특정 인자의 중요도를 산출
  - LIME, SHAP, EMP, RISE, XRAI…
- Input Gradient-Based Methods
  - 입력에 대한 출력의 미분치로 입력의 중요도를 산출
  - Guided Backprobagation, SmoothGrad, VarGrad, Integrated Gradients, Guided Integrated Gradients, DeepLIFT, …
- Decomposition Methods
  - 출력에서 보이는 중요도를 입력으로 전달하는 일종의 역전파 알고리즘을 구성
  - LRP, Contrastive LRP, RAP, …
- Activation-Based Methods
  - CNN의 마지막 activation의 민감도를 인자 중요도 산출에 사용
  - CAM, Grad-CAM, Grad-CAM++, Score-CAM, Ablation-CAM, Layer-CAM, …

Perturbation-based

LIME (Local Interpretable Model-agnostic Explanations)



M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1135–1144, 2016.

LIME

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

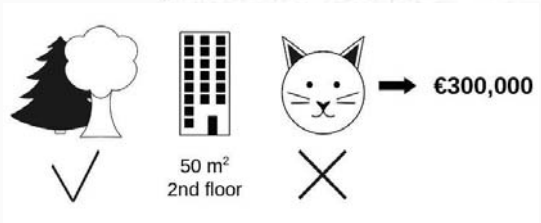
(a) Husky classified as wolf (b) Explanation

허스키를 늑대로 오분류한 예시를 학생들에게 보여줬을 때, 설명을 보기 전/후의 학생들의 1) 모델의 신뢰도와 2) snow가 늑대로 분류되기 위해 사용된 feature일 것이라는 반응에 대한 조사

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

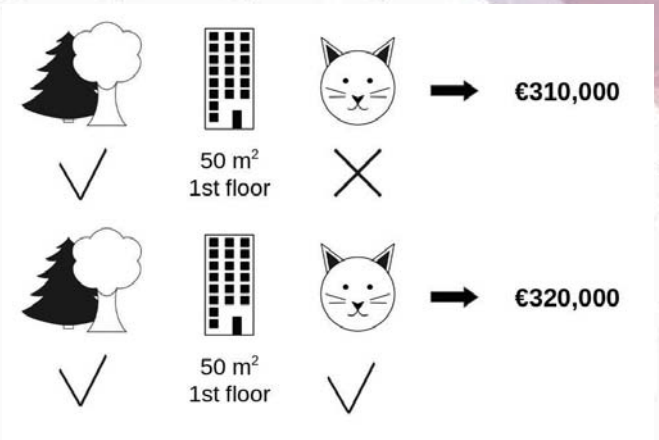
SHAP (SHapley Additive exPlanation)

- Sharpley value (Shapley, 1953)



50m², 2층, 공원 옆, 고양이 금지 아파트
: 가격 예측치가 300,000 유로

➔ 각 인자가 예측치에 미치는 영향력을 평가?

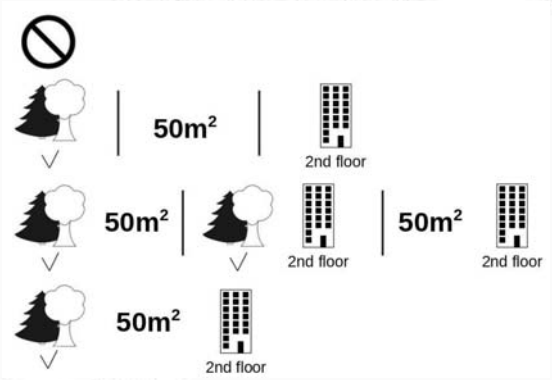


- 크기, 층, 공원 옆 여부가 같을 때,
고양이 허용 여부의 차이에 따른
아파트 가격 예측치의 변화 (-10,000)로 "고양이 금지"의 영향력을 추정

- 이 과정을 여러 설정 샘플에 대해 반복 & 평균

SHAP (SHapley Additive exPlanation)

- 모든 가능한 설정(coalition)의 종류 예시



(공원 옆, 크기:50, 층:2층) =

(0, 0, 0)

(1, 0, 0), (0, 1, 0), (0, 0, 1)

(1, 1, 0), (1, 0, 1), (0, 1, 1)

(1, 1, 1)

f(조합 & 고양이 허용) – f(조합 & 고양이 비허용) 값의 가중 평균값으로 "고양이 허용"의 영향력 평가

$$\phi_j(val) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} (val\,(S \cup \{j\}) - val(S))$$

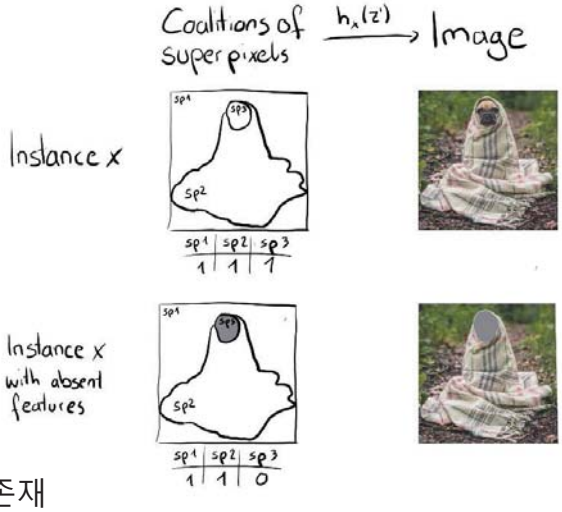
SHAP (SHapley Additive exPlanation)

SHAP: an additive feature attribution method:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

$$z' \in \{0,1\}^M$$

M: no of simplified input features

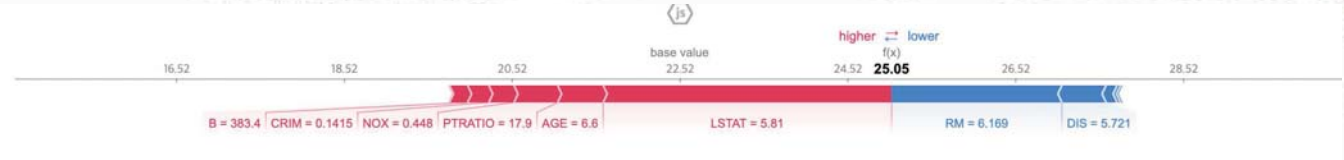


분류기의 종류에 따라 Kernel SHAP, Tree SHAP 등이 존재

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

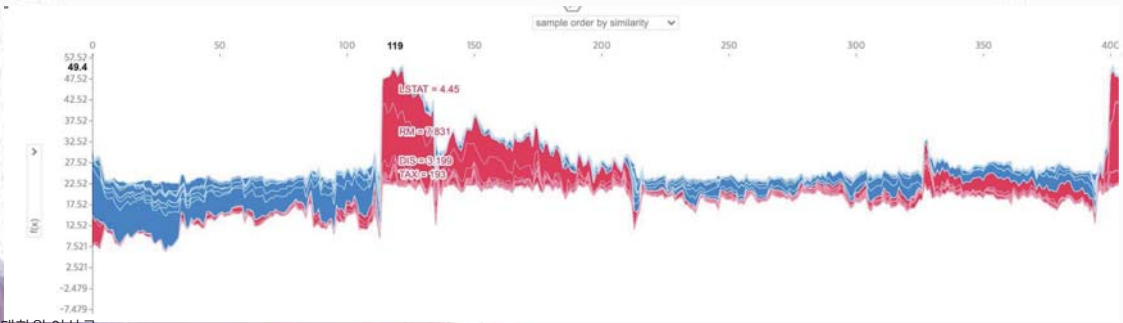
SHAP (SHapley Additive exPlanation)

보스턴 주택 가격 데이터 분석 예시 (특정 집):

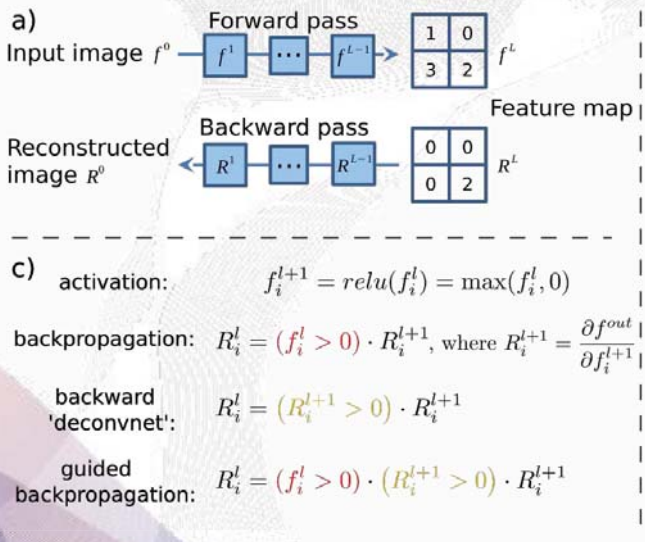


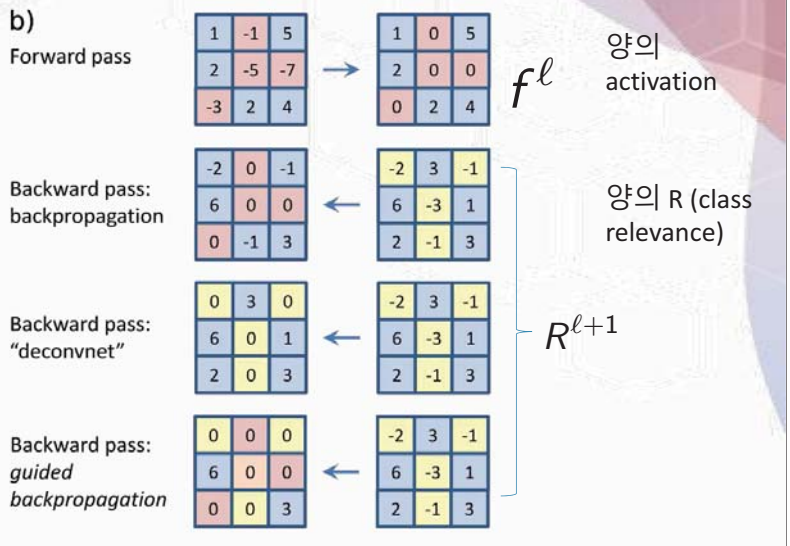
- 집값 상승에 긍정적 영향: LSTAT (지역의 하위 계층 비율)
- 집값 상승에 부정적 영향: RM (방의 수)

모든 집에 대한 분석 시각화



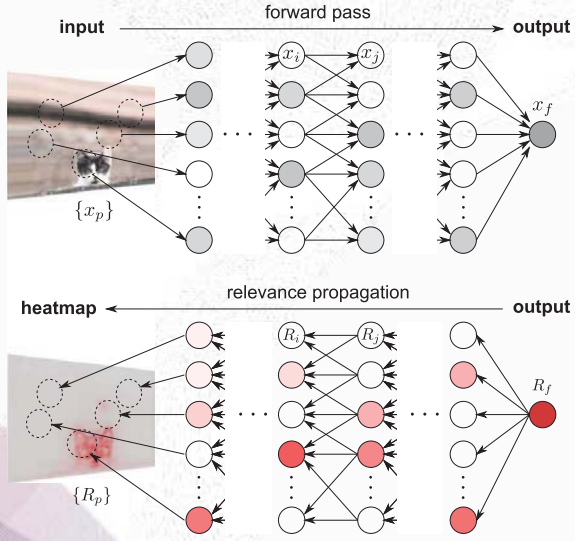
Guided Backprobagation





J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In ICLR (workshop track), 2015.

LRP



함수 f에 대한 1차 Taylor 전개:

$$f(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) + \left(\frac{\partial f}{\partial \boldsymbol{x}}|_{x=\tilde{x}}\right)^{\top} \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) + \varepsilon = 0 + \sum_p \underbrace{\frac{\partial f}{\partial x_p}|_{x=\tilde{x}} \cdot (x_p - \tilde{x}_p)}_{R_p(\boldsymbol{x})} + \varepsilon,$$

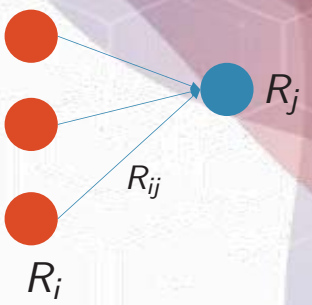
$f(\tilde{\boldsymbol{x}}) = 0$

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Mu¨ller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE, 10(7):1–46, 07 2015.
G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Mu¨ller. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition, 65:211–222, 2017

LRP

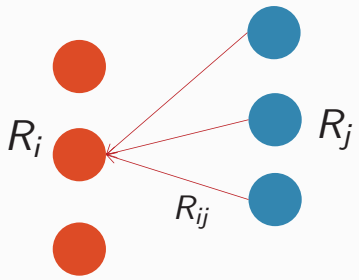
Relevance의 1차 Taylor 전개:

$$R_j = \left(\frac{\partial R_j}{\partial \{x_i\}}|_{\{\tilde{x}_i\}^{(j)}}\right)^{\top} \cdot (\{x_i\} - \{\tilde{x}_i\}^{(j)}) + \varepsilon_j = \sum_i \underbrace{\frac{\partial R_j}{\partial x_i}|_{\{\tilde{x}_i\}^{(j)}} \cdot (x_i - \tilde{x}_i^{(j)})}_{R_{ij}} + \varepsilon_j,$$



Relevance의 역전파 (backpropagation) 정의:

$$R_i = \sum_j R_{ij}.$$

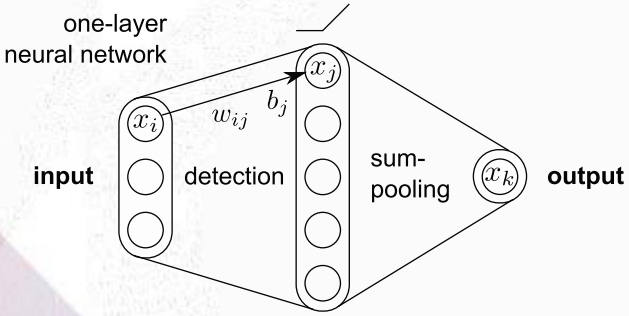


LRP

Relevance 역전파를 위해 원래의 DNN의 레이어 구조가 아닌, detection-pooling 구조를 사용

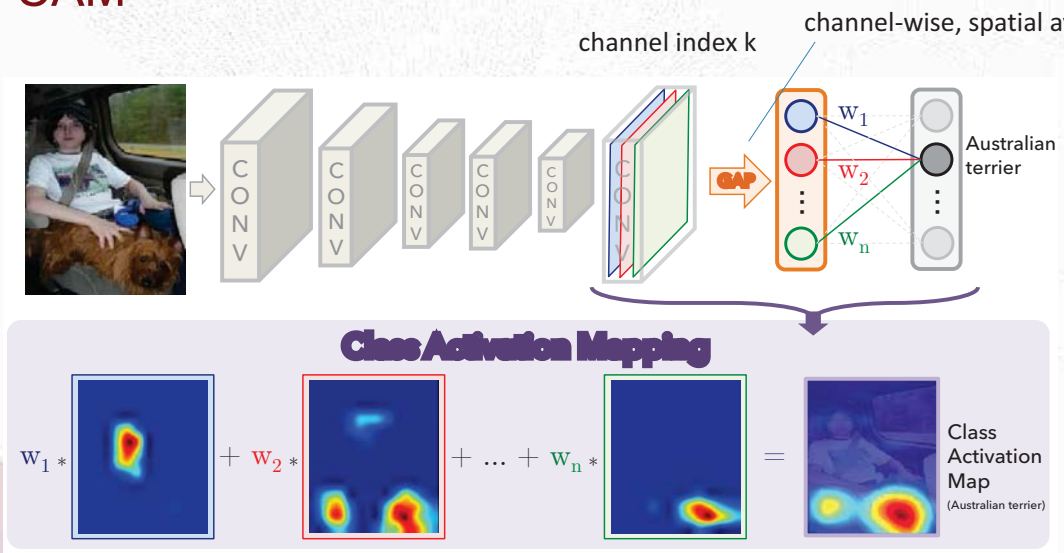
$$f(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) + \left(\frac{\partial f}{\partial \boldsymbol{x}}|_{x=\tilde{x}}\right)^{\top} \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) + \varepsilon = 0 + \sum_p \underbrace{\frac{\partial f}{\partial x_p}|_{x=\tilde{x}} \cdot (x_p - \tilde{x}_p)}_{R_p(\boldsymbol{x})} + \varepsilon,$$

$f(\tilde{\boldsymbol{x}}) = 0$



이는 Taylor 전개 시 root finding 절차를 간소화 시키기 위함

CAM



classification score:

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y)$$

attribution map:

$$M_c(x,y) = \sum_k w_k^c f_k(x,y).$$

CNN의 마지막 레이어가 GAP (global average pooling)구조를 가지도록 신경망을 변형하여 학습

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discrimina- tive localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

CAM 예시

- 입력의 최대 확률 class 특징을 highlight

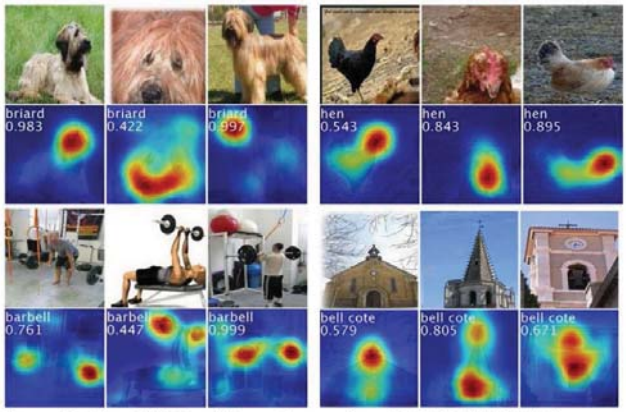


Figure 3. The CAMs of four classes from ILSVRC [20]. The maps highlight the discriminative image regions used for image classification e.g., the head of the animal for *briard* and *hen*, the plates in *barbell*, and the bell in *bell cote*.

- 사용자가 지정한 class의 특징을 highlight

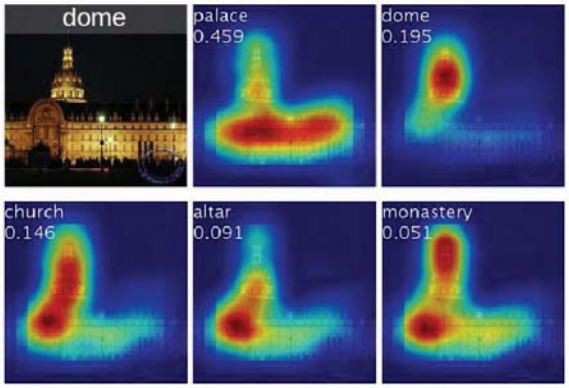


Figure 4. Examples of the CAMs generated from the top 5 predicted categories for the given image with ground-truth as dome. The predicted class and its score are shown above each class activation map. We observe that the highlighted regions vary across predicted classes e.g., *dome* activates the upper round part while *palace* activates the lower flat part of the compound.

Grad-CAM

- CAM의 문제점
  - 설명 생성을 위해 원래 신경망의 마지막에 반드시 GAP 레이어가 필요
  - 구조 변경 시 원래 신경망과 비교하여 예측 성능의 열화 발생 문제

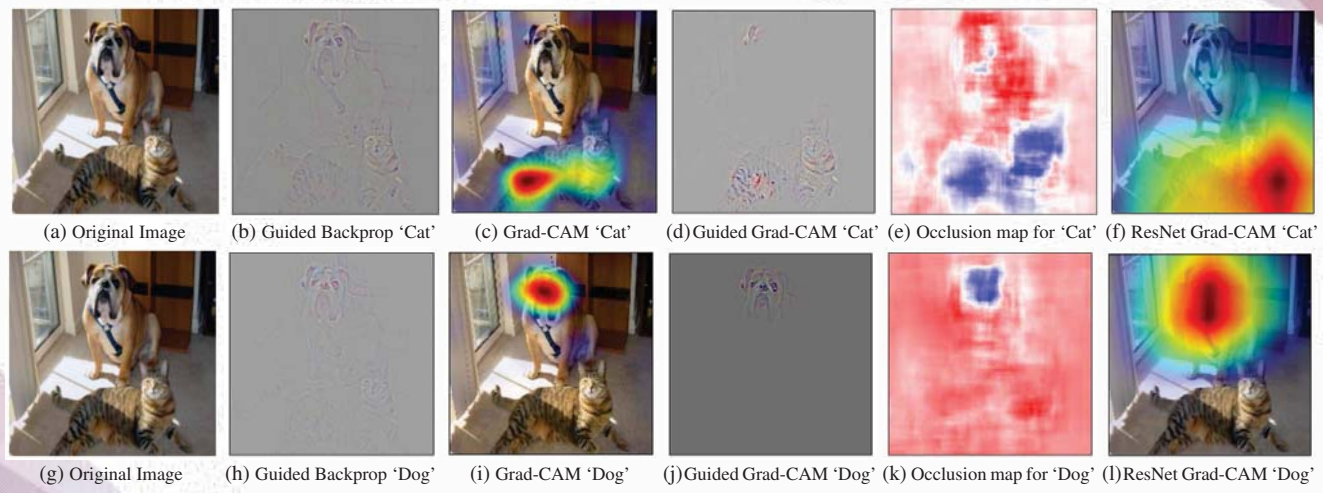
- Grad-CAM
  - CAM의 식을 기반으로, explicit한 GAP 구조 없이 미분을 통해 CAM을 생성할 수 있음을 보임

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L^c_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.

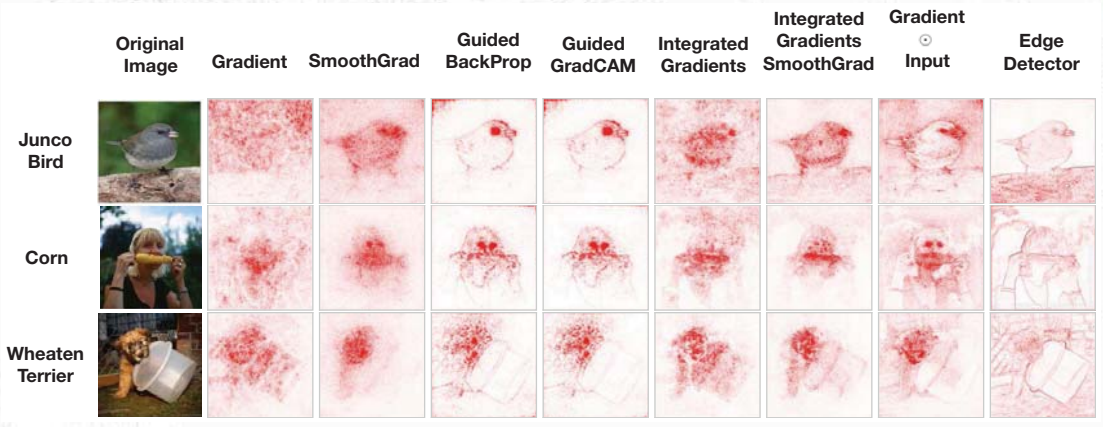
Grad-CAM 예시



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d) Guided Grad-CAM 'Cat' (e) Occlusion map for 'Cat' (f) ResNet Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j) Guided Grad-CAM 'Dog' (k) Occlusion map for 'Dog' (l) ResNet Grad-CAM 'Dog'

Sanity Check



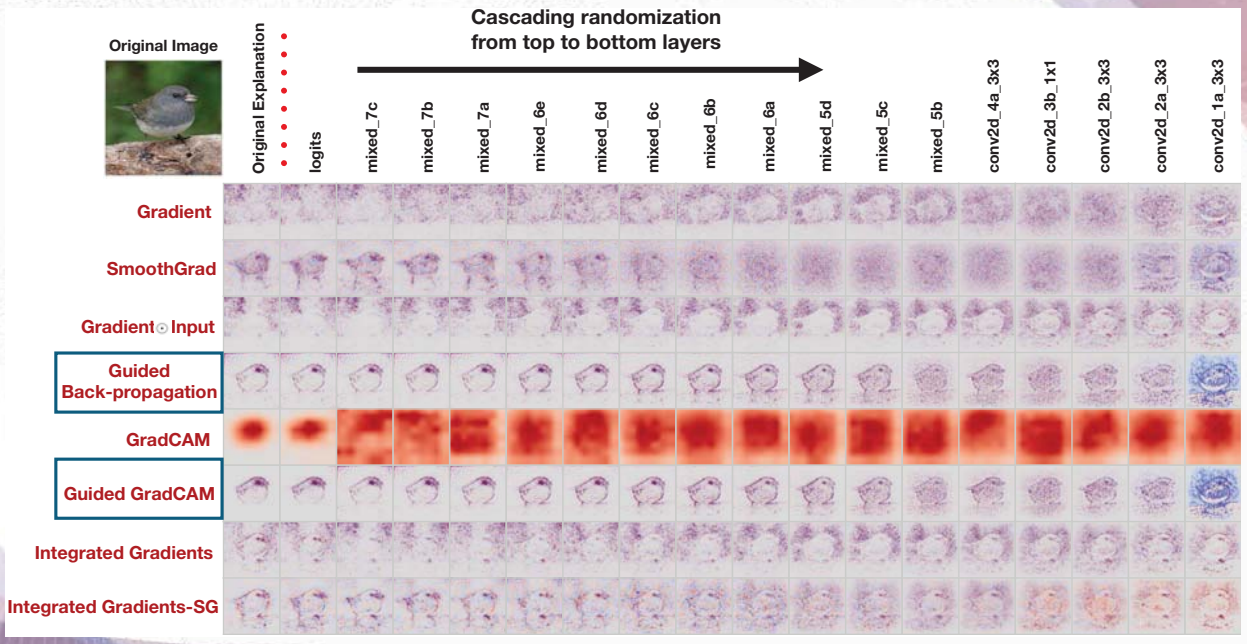
- 일부 attribution map은 edge detector의 결과와 매우 흡사해 보임.
- 좋은 설명(attribution)은, 입력과 모델 모두를 반영해야 함

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018.

Sanity Check

일부 기법은 모델 파라미터에 대한 낮은 민감도를 보임

➔ sanity check 실패



KOREA UNIVERSITY

IJCAI ECAI VIENNA 22

Libra-CAM: An Activation-Based Attribution Based on the Linear Approximation of Deep Neural Nets and Threshold Calibration

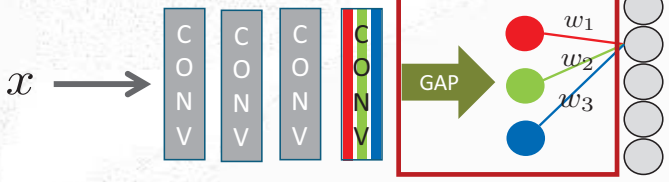
Sangkyun Lee* & Sungmin Han

School of Cybersecurity
Korea University, South Korea

IJCAI-22

CAM and Grad-CAM

❖ **CAM [CVPR`16]**

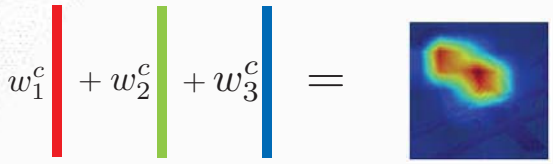


Explicit modification

Trained, Channel-wise

$$s^c(x) = \sum_k \boxed{w_k^c} \sum_{i,j} A_{ij}^k$$

Class Activation Map (CAM)



$$I_{\text{CAM}}^c(x) := \sum_k \boxed{w_k^c} A_{ij}^k$$

❖ **Grad-CAM [ICCV`17]**

$$I_{\text{Grad-CAM}}^c(x) := \text{ReLU}\Big(\sum_k \boxed{w_k^c} A^k(x) \Big)$$

Computable, Channel-wise

$$\boxed{w_k^c = \frac{1}{Z} \sum_{i,j} \frac{\partial f_c(x)}{\partial A_{ij}^k}}$$

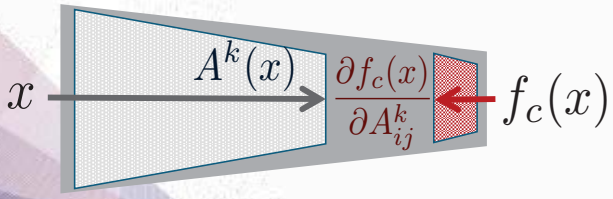
Grad-CAM [Selvaraju et al., ICCV`17] and Others

Q2 **Q1**

$$I_{\text{Grad-CAM}}^c(x) := \boxed{\text{ReLU}}\Big(\sum_k \boxed{\alpha_k^c} A^k(x) \Big)$$

Q1

$$\boxed{\alpha_k^c := \frac{1}{Z} \sum_{i,j} \frac{\partial f_c(x)}{\partial A_{ij}^k}}, \quad Z := \sum_{i,j} 1$$



Q1: Why channel-wise importance weights?

❖ **They are reminiscence of the GAP layer**

Q2: What is the role of ReLU?

❖ Positive values = positive influence?

❖ **It's a naïve form of attribution thresholding**

Q3: Is the gradient information unreliable?

❖ Score-CAM [Wang et al., CVPR`20]: replaces gradient information with prediction scores of channel-masked images

❖ Relevance-CAM [Lee et al., CVPR`20]: uses Contrastive LRP

❖ **In fact, it is quite reliable, since the backprop path is rather short**

Linear Approximation of f_c

- Approximation:

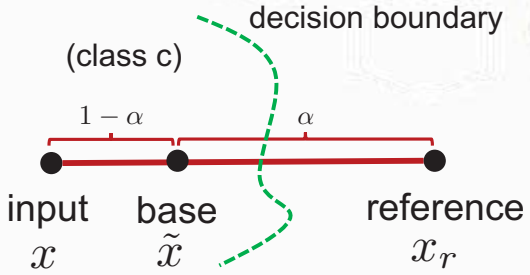
x : a given input → activation $A := A(x)$

$$f_c(x) - f_c(\tilde{x}) \approx \sum_{i,j} \sum_{k} \frac{\partial f_c}{\partial A_{ij}^k}\Big|_x (A_{ij}^k - \tilde{A}_{ij}^k)$$

$\tilde{x}$: a base point → activation $\tilde{A} := A(\tilde{x})$

- A base point is defined with a **contrastive** reference point:

$$\tilde{A} = A + \alpha(A_r - A), \text{ for some } \alpha \in (0, 1).$$



$f_c(x_r) < \tau$

- Approximation error is asymptotically zero as $\alpha \to 0$

Libra-CAM: a CAM based on LInear approximation and threshold caliBRAtion

- A preliminary version:

$$I_r(x) := \rho\left(\alpha \sum_k \frac{\partial f_c}{\partial A^k}\Big|_x \otimes \boxed{(A^k - A_r^k)}\right)$$

Scaling to [0,1] range

Can be arbitrarily small > 0
→ minimize approximation error

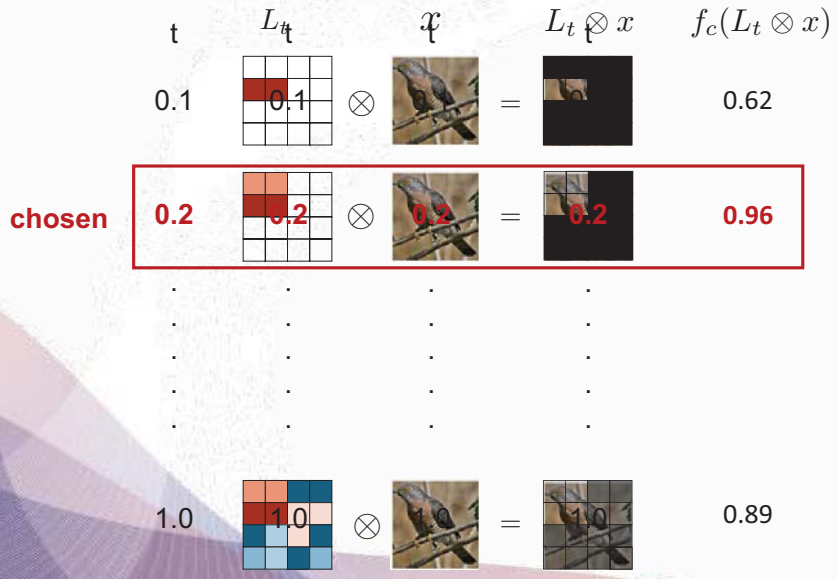
❖ **Multiple contrastive reference points**
- ❖ We can choose any reference without sacrificing the approximation error
- ❖ Use references contrastive to the target class c:

$$I_{\text{Libra-CAM}}(x) := \frac{1}{R} \sum_{r=1}^{R} I_r$$

- ❖ A pre-built reference library is used with ref filtering: $f_c(I_r \otimes x) > \gamma$

Threshold Calibration

❖ **Determine the best threshold to mute relevance values**



for $t \leftarrow 0.1$ to 1.0 with the increment of 0.1 **do**
  Parallel(i, j): $[L_t]_{ij} \leftarrow L_{ij}$ if $L_{ij} \geq t$, else 0.
end for
Parallel(t): $C_t \leftarrow f_c(L_t \otimes x)$.
$t^* \leftarrow \arg\max_t C_t$.
return L_{t^*}

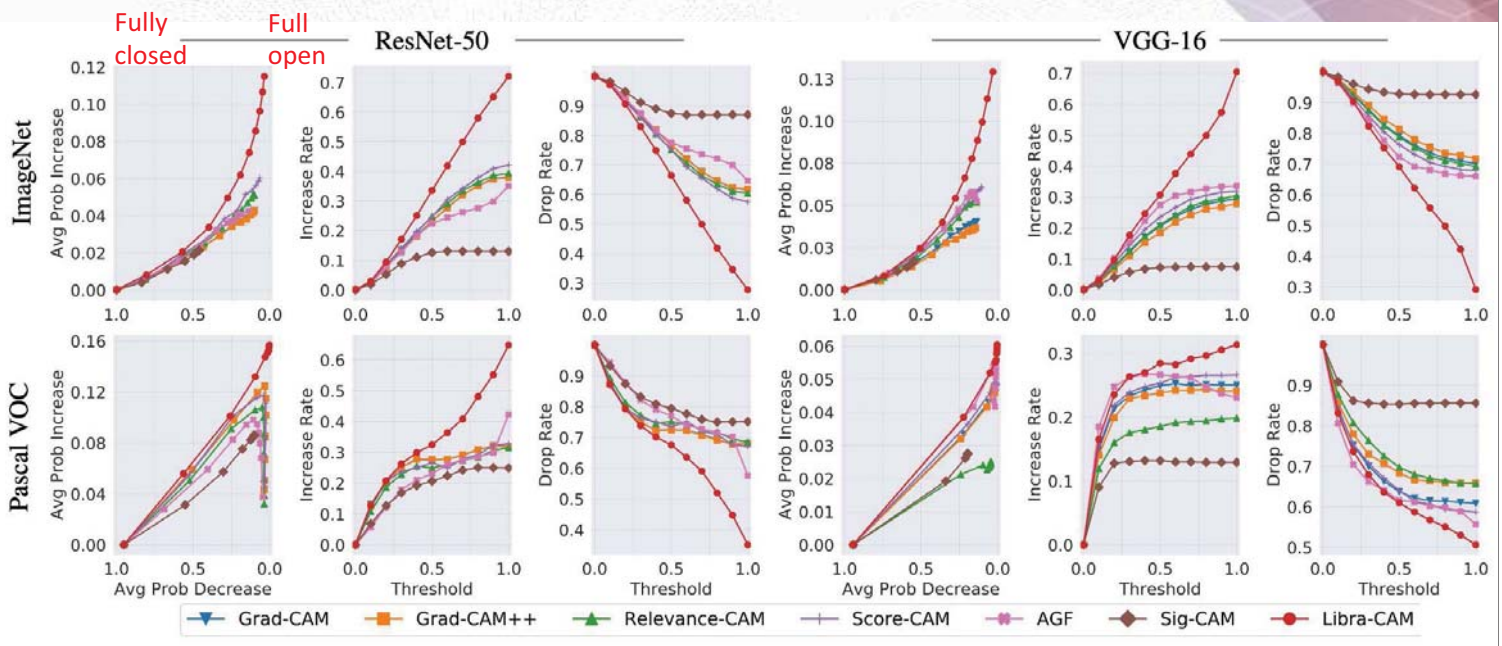
Quality Measures

- Avg Prob Inc (API): $\frac{1}{n}\sum_{i=1}^{n}\frac{(o_i^c - y_i^c)^+}{y_i^c}$
- Avg Prob Drop (APD): $\frac{1}{n}\sum_{i=1}^{n}\frac{(y_i^c - o_i^c)^+}{y_i^c}$
- Inc Rate (IR): $\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(y_i^c < o_i^c)$
- Drop Rate (DR): $\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(y_i^c > o_i^c)$

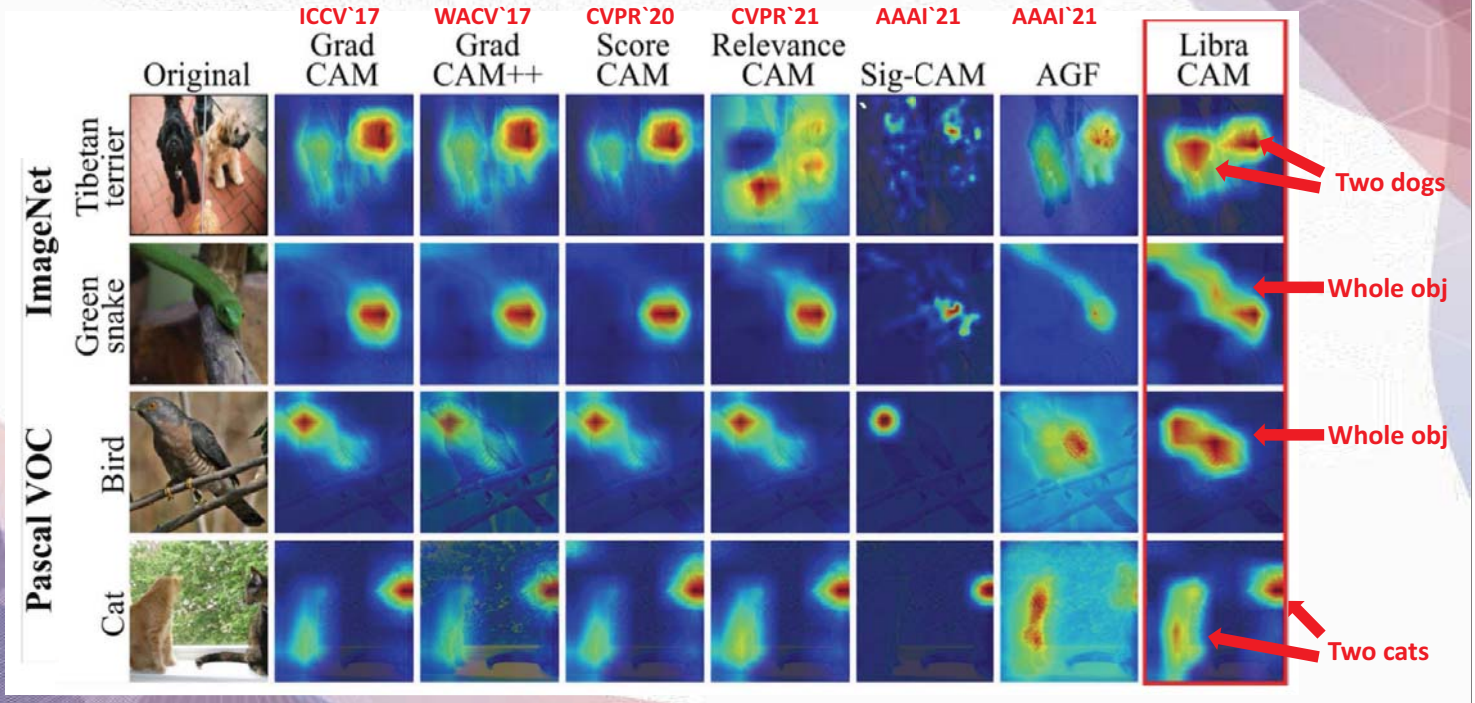
- $y_i^c = f_c(x)$
- $o_i^c = f_c(I^c(x) \otimes x)$

Attribution quality

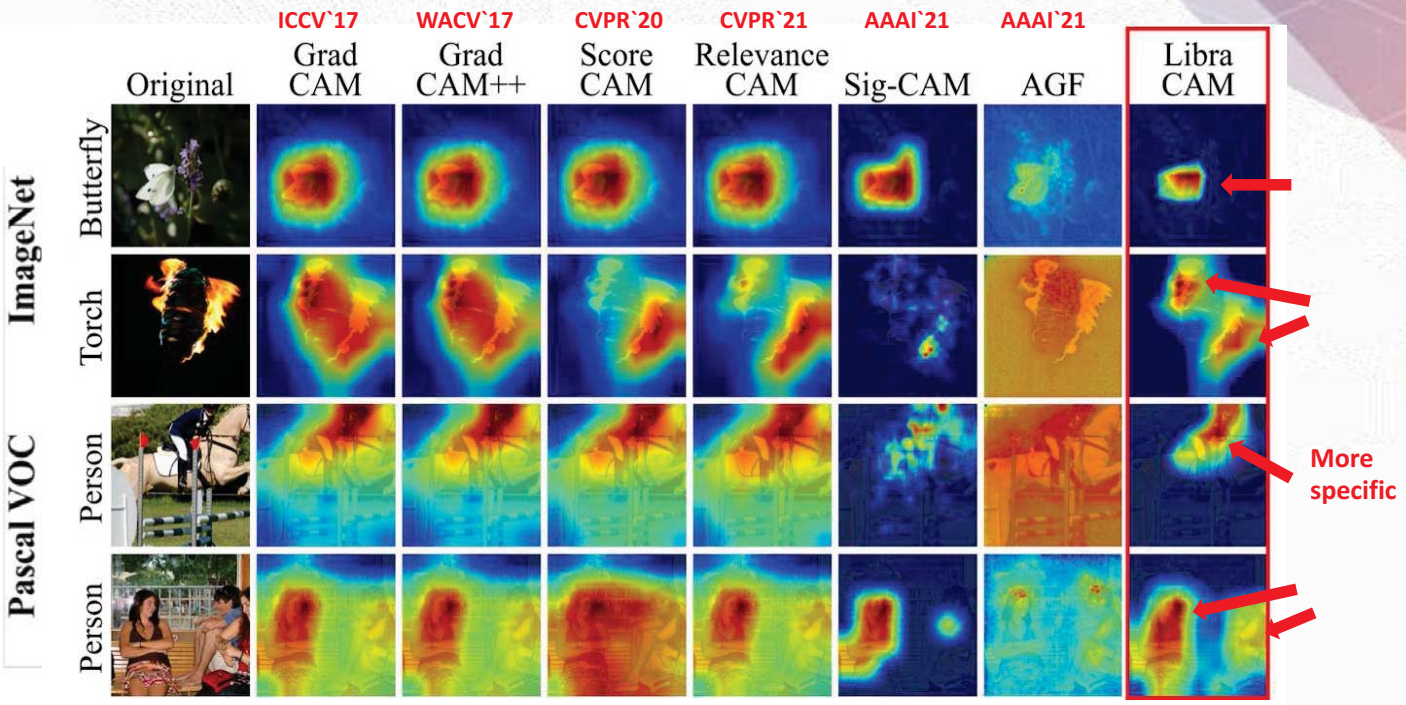
at threshold levels t ∈ [0, 1] with the increment of 0.1 from left to right in all plots.



Qualitative Result (VGG-16)



Qualitative Result (ResNet-50)



Conclusion

- XAI는 AI의 이해를 증진
  - 보다 향상된 AI의 개발
  - AI의 취약성 발견
  - AI의 해석을 기반으로 한 과학적 사실/가설의 검증, 새로운 가설의 발견
- XAI의 발전 방향
  - 해석 가능한 새로운 모델의 개발
  - 기존 해석 가능한 모델 (decision tree)의 예측력 향상
  - 기존 모델을 해석할 수 있는 model induction 기법
- 고려대 AIRLAB이 개발한 Libra-CAM은 가장 좋은 XAI 성능을 보유하고 있음 (https://air.korea.ac.kr/)

Thank You