AI 최신 기술은 어떻게 찾아 보고 계신가요?

모두의연구소
박은수 Research Director

3개월 집중 코스가 끝난 후의 팀프로젝트 계획들 ...

아직 생각해보고 있습니다. 스타트업 제품을 만들거나 사회에 실질적인 도움이 될만한 프로젝트를 하면 재밌을 것 같아요. 게임 이외의 강화학습 활용이 쉽지 않네요.
‘Nvidia Jetson tx2’ + RL + OpenAI = AI Robot
영상처리와 강화학습을 접목해서 무언가 함께 하고 싶습니다
1. 고기 또는 회(스시)의 종류를 판독하는 알고리즘 2. 원단(Textile)의 불량을 자동 검단하는 알고리즘 3. 그 외에도 더 재밌는 아이디어가 있다면 도전하고 싶습니다.
영상 세그멘테에션 쪽으로 하고 싶음(CNN, R-CNN, Fast R-CNN, Faster R-CNN...)
아직 잘 모르겠지만, 게임환경에서 강화학습을 진행해보고 싶습니다. 다른 분의 흥미로운 프로젝트면 적극 참여하고싶습니다.
visual q&a, 또는 오브젝트 디텍팅 프로젝트를 해보고 싶습니다
글쎄요...강화학습을 적용해서 움직임을 학습.
.

모두의연구소

3개월 집중 코스가 끝난 후의 팀프로젝트 계획들 ...

모두의연구소

모르겠음 (5)
아직 미정입니다..
해당 분야의 대표 논문을 선정해서 구현해 보는 과정으로 하면 좋을 것 같습니다.
강화학습용 "환경"을 rnn or cnn으로 학습시킨 시뮬레이터 형태로 만들어보고 싶습니다
오브젝트 인식후 텍스트로 결과 처리
Text based 케글 컴페티션 (예 https://www.kaggle.com/c/quora-question-pairs)
주식/가상화폐 같은 트레이딩에 강화학습을 적용
영상처리에 강화학습 접목
구체적인 적용 대상은 생각 중이나 모델/구현 내용은 사람의 동작을 인식하는 모델을 활용해서 프로젝트를 진행하고 싶습니다.
팀이 생기면 얘기를 더 할 수 있을듯하고.. 아직 모르겠습니다. 희망사항은 드론입니다ㅎㅎ
영상처리+강화학습. or 자율주행 (?)

아직은 모르시는게
당연한 것 같아요

저도 모름 ....

최신 트렌드 따라가 보는 방법

모두의연구소

- 테리의 딥러닝 토크
  - #0.3. SNS로 딥러닝 소식 팔로우 하는 법 (1/2)
    - https://youtu.be/Z1OdPpq9w0o
  - #0.4. SNS로 딥러닝 소식 팔로우 하는 법 (2/2)
    - https://youtu.be/w1oQQmu8NKo

제가 하는 것들...

- 메일로 소식 받기 : The Wild Week in AI

The Wild Week in AI

February 19 · Issue #79 · View online

The Wild Week in AI is a weekly AI & Deep Learning newsletter curated by @dennybritz.

매주 월요일 따끈따끈한 AI소식을 묶어서 보내줍니다

얼마나 따끈따끈한지 살펴보죠

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

모두의연구소

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

News

최신 뉴스군요

Google Cloud TPUs now available in beta

CLOUDPLATFORM.GOOGLEBLOG.COM – Share

Cloud TPUs are now available in beta on Google Cloud Platform (GCP) to train and run ML models more quickly. Built with four custom ASICs, each Cloud TPU packs up to 180 teraflops of floating-point performance and 64 GB of high-bandwidth memory onto a single board. Usage is billed by the second at the rate of $6.50 USD per Cloud TPU per hour.

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

모두의연구소

News

최신 뉴스군요

Google Cloud TPUs now available in beta

CLOUDPLATFORM.GOOGLEBLOG.COM – Share

Cloud TPUs are now available in beta on Google Cloud Platform (GCP) to train and run ML models more quickly. Built with four custom ASICs, each Cloud TPU packs up to 180 teraflops of floating-point performance and 64 GB of high-bandwidth memory onto a single board. Usage is billed by the second at the rate of $6.50 USD per Cloud TPU per hour.

Posts, Articles, Tutorials

정말 꼭 보고싶은 글들이에요

Deep Reinforcement Learning Doesn't Work Yet

WWW.ALEXIRPAN.COM – Share

Deep reinforcement learning is surrounded by mountains of hype. Unfortunately, it doesn't really work yet. There are a lot of problems in the way, many of which feel fundamentally difficult. This post goes into the technical details of what's so difficult about RL.

Shipping a Neural Network on iOS with CoreML and PyTorch

ATTARDI.ORG – Share

The author trained a simple neural network to solve a well-defined challenge in a real iOS app. He walks through every step of the development, from problem definition all the way to the App Store.

메일상자

날짜별로 정렬

- **The Wild Week in AI** 2018. 2. 19.
  The Wild Week in AI - Cl... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2018. 2. 12.
  The Wild Week in AI - Li... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2018. 2. 5.
  The Wild Week in AI - An... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2018. 1. 29.
  The Wild Week in AI - Go... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2018. 1. 22.
  The Wild Week in AI - A... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2018. 1. 15.
  The Wild Week in AI - Fa... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2017. 12. 18.
  The Wild Week in AI - An... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2017. 12. 11.
  The Wild Week in AI - Al... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2017. 12. 4.
  The Wild Week in AI - AI I... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2017. 11. 27.
  The Wild Week in AI - Ap... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2017. 11. 13.
  The Wild Week in AI - Wa... 받은 편지함 - Google
  If you enjoy the newsletter, please consider sharing it on Twitter, Facebook, etc! Really a...
- **The Wild Week in AI** 2017. 11. 6.
  The Wild Week in AI - Ac... 받은 편지함 - Google
  If you enjoy the newsletter, please consider

The Wild Week in AI

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

모두의연구소

Code, Projects & Data

PyTorch implementation of "Efficient Neural Architecture Search via Parameters Sharing"

GITHUB.COM – Share

ENAS reduces the computational requirement (GPU-hours) of Neural Architecture Search by 1000x via parameter sharing between models that are subgraphs within a large computational graph. It achieves SOTA on Penn Treebank language modeling.

놓치기 싫은 코드와 데이터 소식들

Tensor Comprehensions (Facebook)

RESEARCH.FB.COM – Share

Tensor Comprehensions (TC) is a C++ library to automatically synthesize high-performance machine learning kernels using Halide, ISL and NVRTC or LLVM. TC provides basic integration with Caffe2 and pybind11 bindings for use with python. Also see the arXiv paper.

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

모두의연구소

Highlighted Research Papers

이 주의 페이퍼 까지

[1802.05365] Deep contextualized word representations

ARXIV.ORG – Share

Deep contextualized word representations model both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Word vectors are learned functions of the internal states of a deep bidirectional language model, which is pre-trained on a large text corpus. They improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis.

[1802.04697] Learning to Search with MCTSnets

ARXIV.ORG – Share

Planning problems are most typically solved by tree search algorithms that simulate ahead into the future, evaluate future states, and back-up those evaluations to the root of a search tree. In this paper, the author learn where, what and how to search. The architecture, called an MCTSnet, incorporates simulation-based search inside a neural network, by expanding, evaluating and backing-up a vector embedding.

[1802.03268] Efficient Neural Architecture Search via Parameter Sharing

ARXIV.ORG – Share

Efficient Neural Architecture Search (ENAS) is a fast and inexpensive approach for automatic model design. A controller learns to discover neural network architectures by searching for an optimal subgraph within a large computational graph. ENAS delivers strong performance and is 1000x less expensive than standard Neural Architecture Search.

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

모두의연구소

총 4개의 카테고리

News

Posts, Articles, Tutorials

Code, Projects & Data

Highlighted Research Papers

카테고리별로 보통 4개씩 글들이 올라옵니다

너무 보고싶은데 이걸 언제 다 보나요 …

매주 보내주네요 : 우리나라 시간으로는 매주 월요일 저녁에 해당합니다

모두의연구소

모두연에 매주 이것을 함께 보는 프로그램을 만들 예정입니다

Coming Soon : The Mild Week in AI

News

Posts, Articles, Tutorials

너무 바빠요 ...

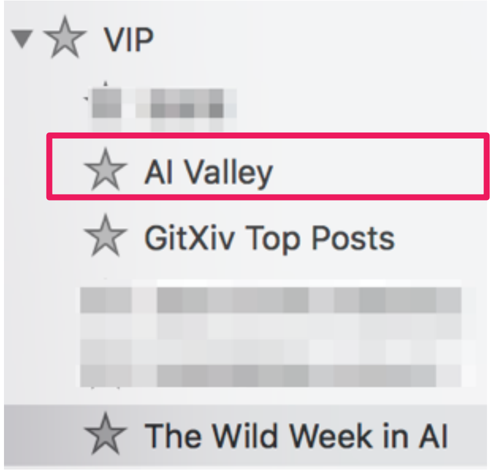
Code, Projects & Data

Highlighted Research Papers

제가 하는 것들...

모두의연구소

- 메일로 소식 받기 : AI Valley



Top Posts This Week

AI-VALLEY.COM

매주 토요일 새벽 따끈따끈한 AI소식을 묶어서 보내줍니다

얼마나 따끈따끈한지 살펴보죠

AI That Creates AI

Video - 128 shares

Next-Level Surveillance: China Embraces Facial Recognition

Video - 1224 shares

Will the Future Be Human?

Video - 2559 shares

Building a Deep Neural Net In Google Sheets

Article - 71 shares

Will Artificial Intelligence Replace Doctors?

Video - 30 shares

RedditSota/state-of-the-art-result-for-machine-learning-problems

Repo - 5921 stars

Deep Generative Models

Article - 19 shares

Machine Learning From Scratch - eriklindernoren/ML-From-Scratch

Repo - 7666 stars

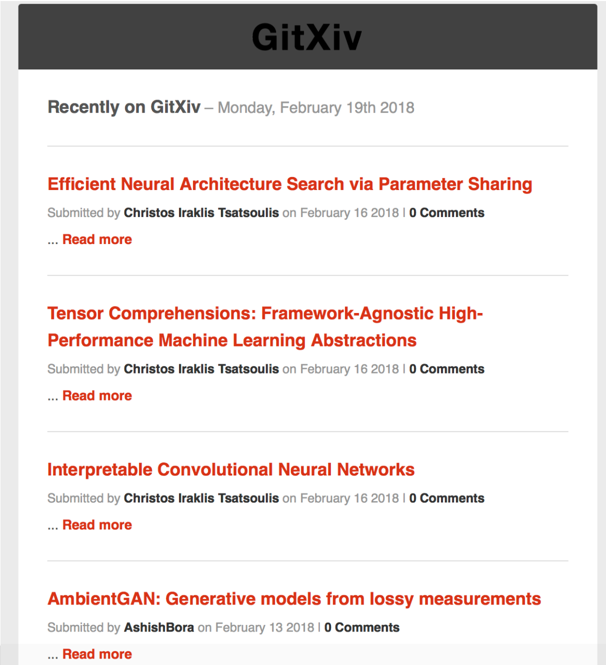
See More

AI Valley

제목만 봐도 재밌고 좋은 글들이 많은 것 같습니다

제가 하는 것들...

- 메일로 소식 받기 : GitXiv Top Posts

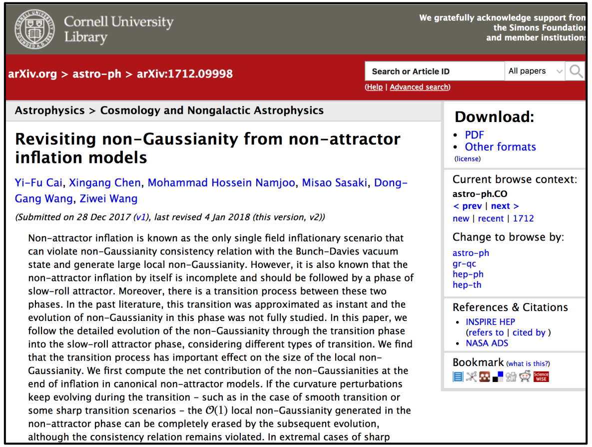


매주 월요일 GitXiv의
소식을 전해 줍니다

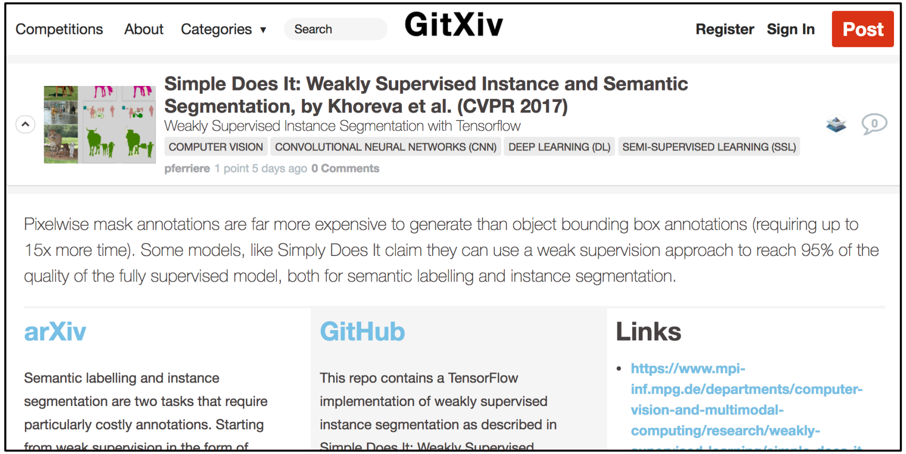
제가 하는 것들...

모두의연구소

- 메일로 소식 받기 : GitXiv Top Posts



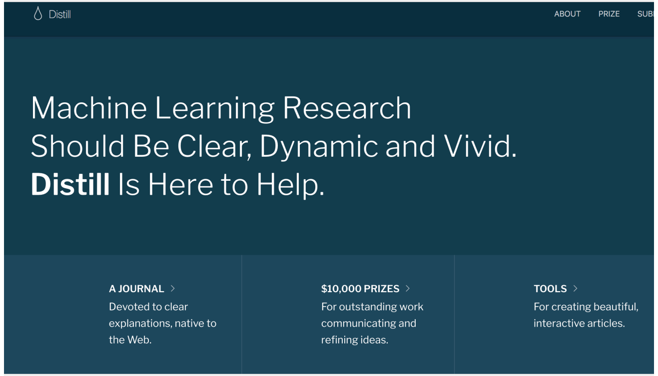
논문을 볼 수 있음



1. 논문을 볼 수 있음
2. 코드를 볼 수 있음
3. 기타 관련 링크를 볼 수 있음

제가 하는 것들...

- 새로운 형태의 논문 : Distill

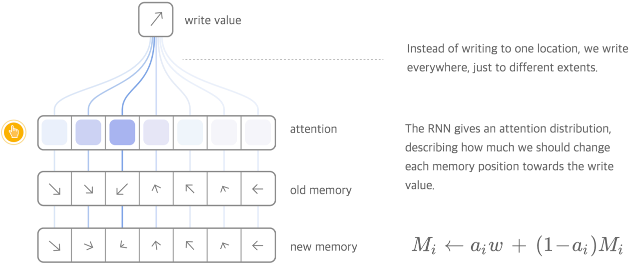


BibTeX citation

```
@article{olah2016attention,
  author = {Olah, Chris and Carter, Shan},
  title = {Attention and Augmented Recurrent Neural Networks},
  journal = {Distill},
  year = {2016},
  url = {http://distill.pub/2016/augmented-rnns},
  doi = {10.23915/distill.00001}
}
```

Citation

내용을 이해하기 좋게 상호작용 할 수 있는 요즘 시대에
맞는 형태의 출간 형태로 보시면 됩니다

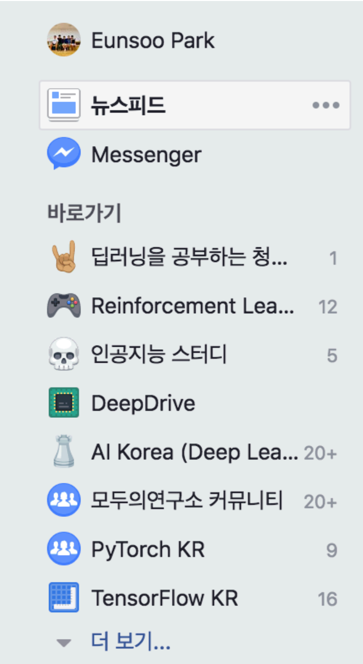


직접 조작가능

But how do NTMs decide which positions in memory to focus their attention on? They actually use a combination of two different methods: content-based attention and location-based attention. Content-based attention allows NTMs to search through their memory and focus on places that match what they're looking for, while location-based attention allows relative movement in memory, enabling the NTM to loop.

제가 하는 것들...

- SNS로 소식받기 : 페이스북

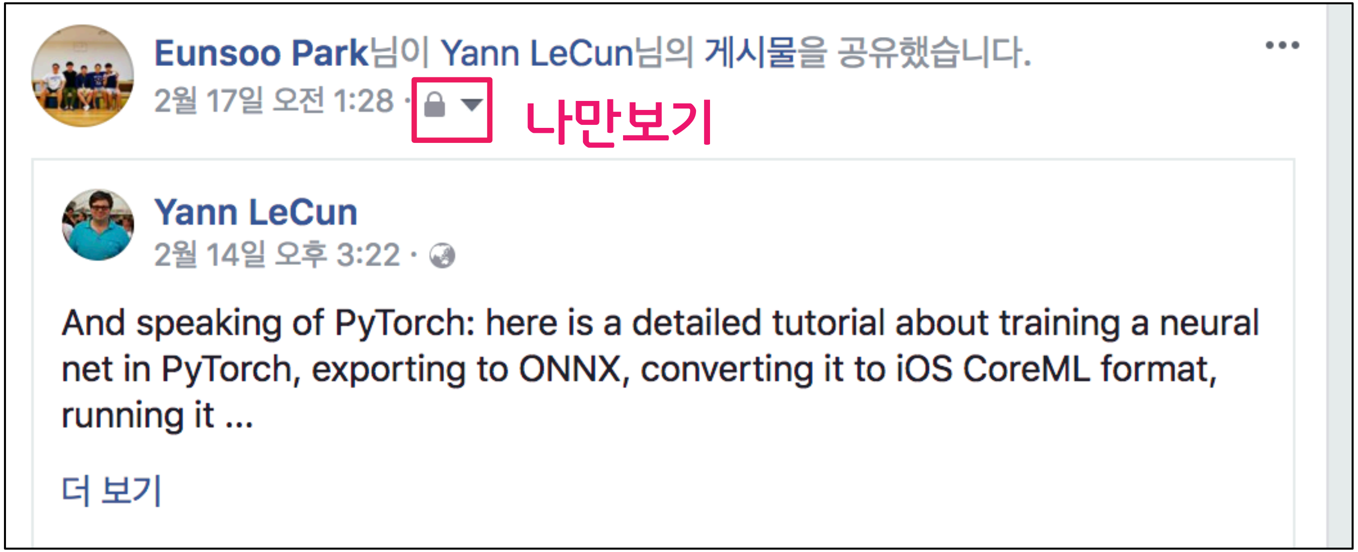


아마 이미 여러분은 저보다 더 많이 가입하셨을것 같습니다

제가 하는 것들...

- SNS로 소식받기 : 페이스북

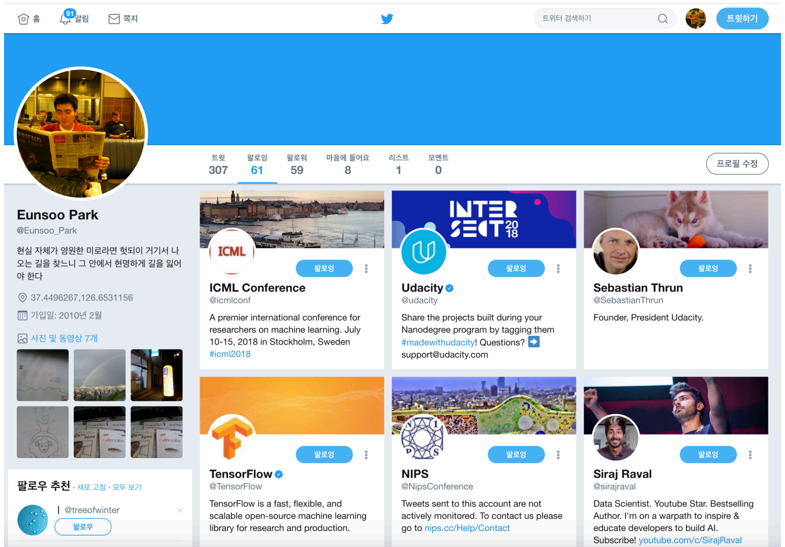
전 소심해서 나만보기로 공유해서 스크랩 해 두고 거의 보지 않는 수백(천?)개의 글이 쌓여있습니다



제가 하는 것들...

모두의연구소

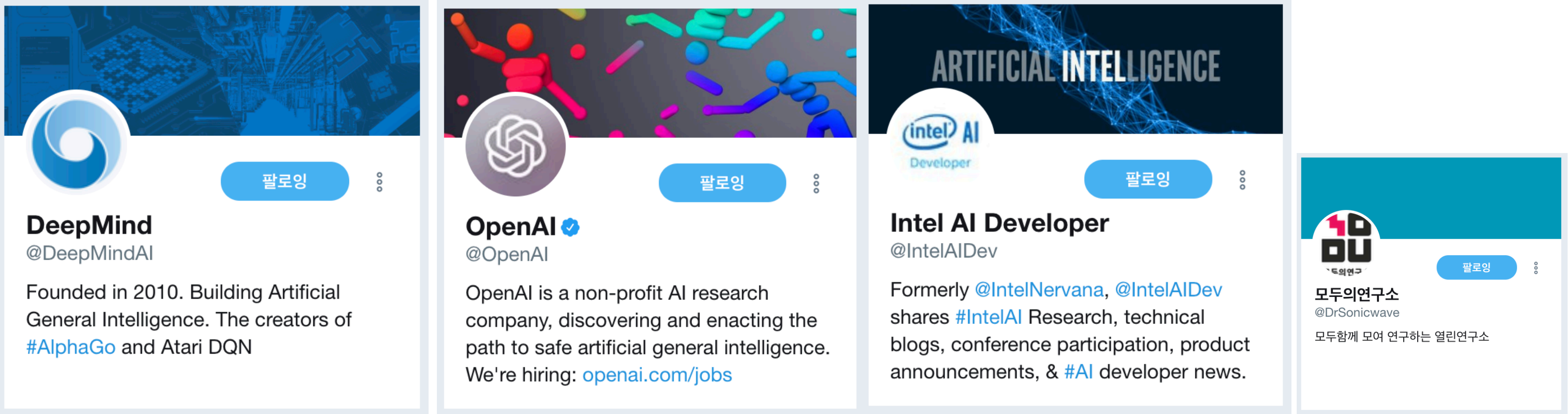
- SNS로 소식받기 : 트위터



제가 하는 것들...

모두의연구소

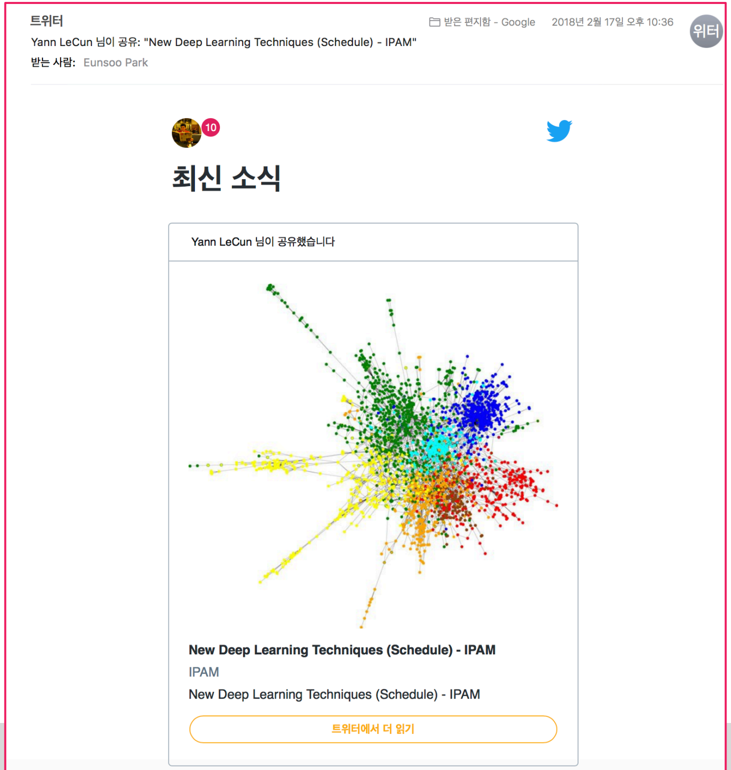
- SNS로 소식받기 : 트위터



제가 하는 것들...

모두의연구소

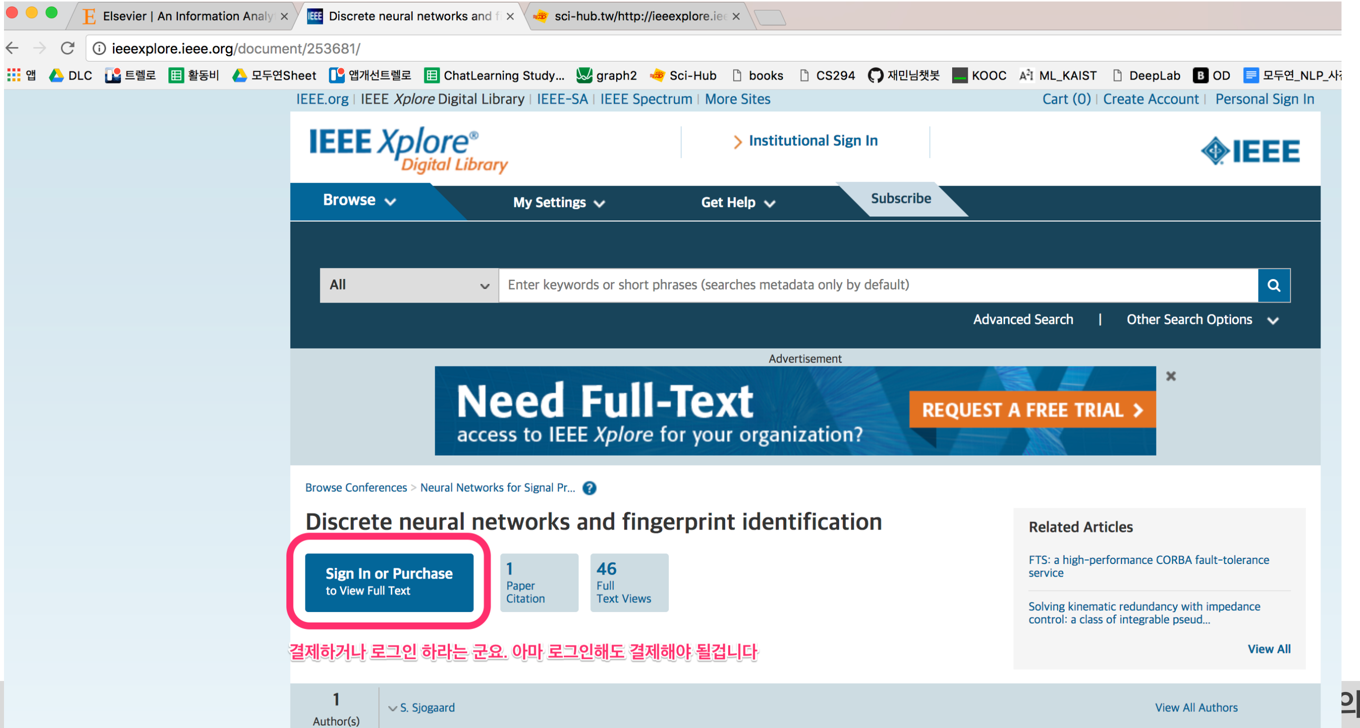
- SNS로 소식받기 : 트위터는 메일도 보내줘요



제가 하는 것들...

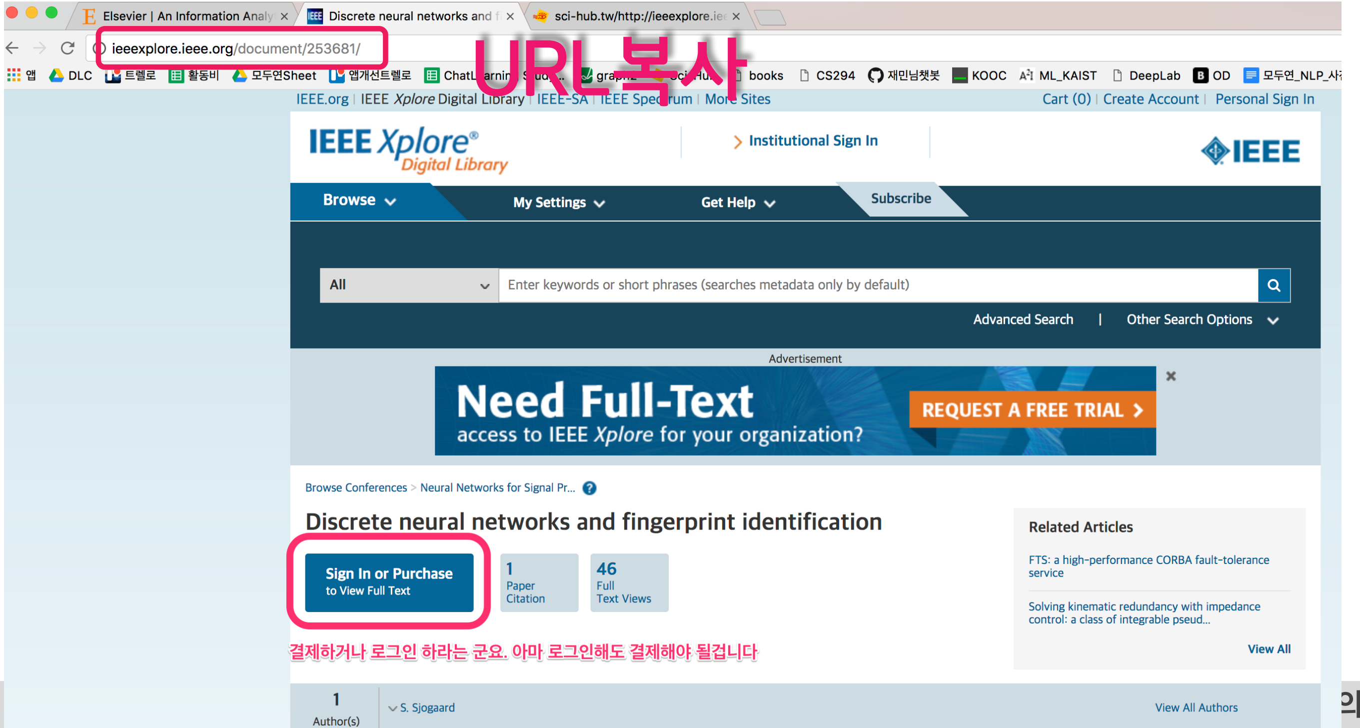
모두의연구소

- 좋은 논문이 있는데 결재하라고 해요~



제가 하는 것들...

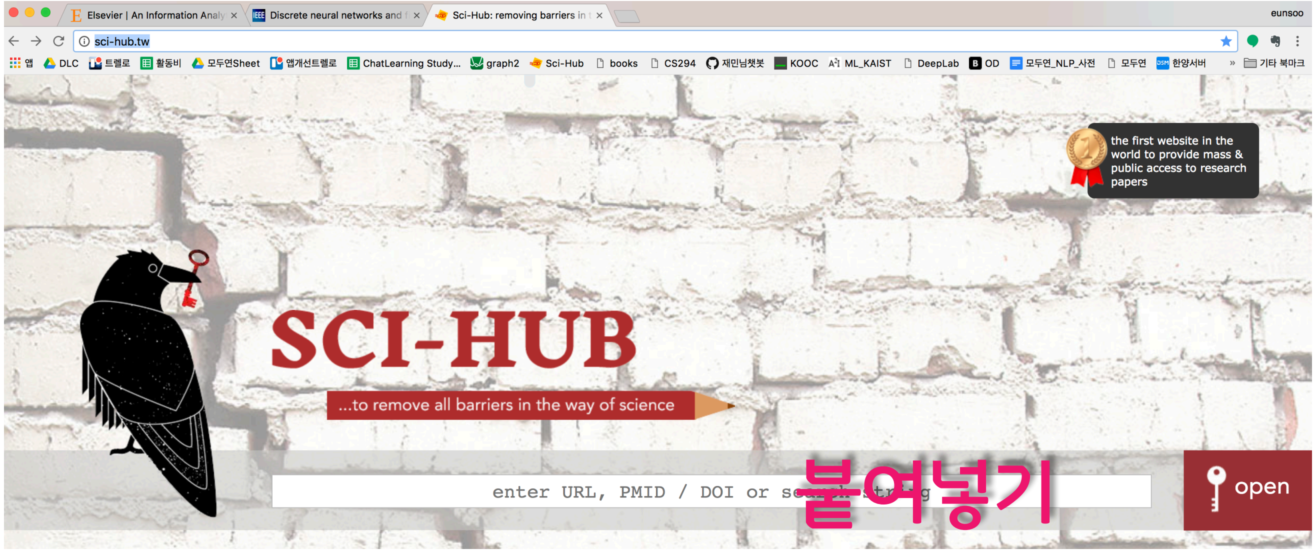
- 좋은 논문이 있는데 결재하라고 해요~



모두의연구소

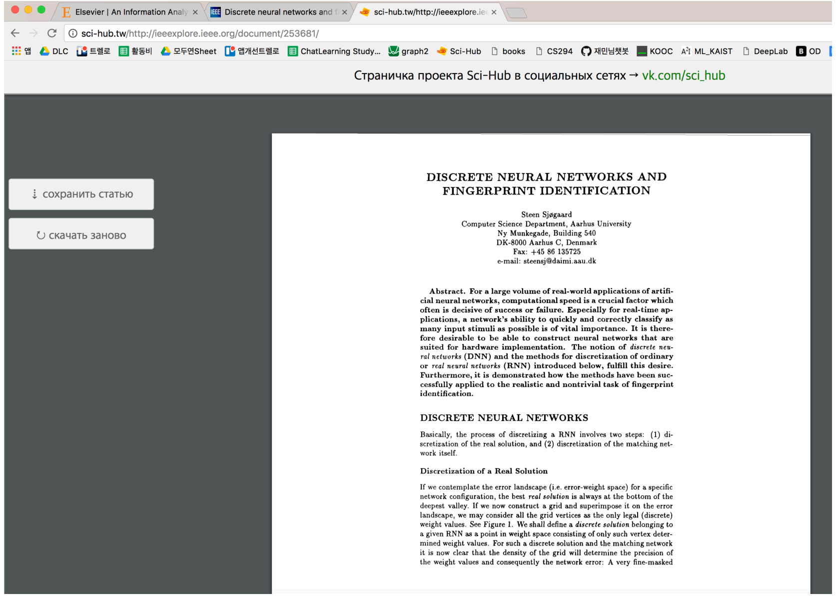
제가 하는 것들...

- 좋은 논문이 있는데 결재하라고 해요~
  – SCI-Hub : http://sci-hub.tw/



제가 하는 것들...

- 좋은 논문이 있는데 결재하라고 해요~
  - SCI-Hub : http://sci-hub.tw/



성공

제가 하는 것들...

모두의연구소

- 추천하는 연구주제 : OpenAI Research 2.0

Requests for Research 2.0

JANUARY 31, 2018

We're releasing a new batch of **seven unsolved problems** which have come up in the course of our research at OpenAI. Like our original Requests for Research (which resulted in several papers), we expect these problems to be a fun and meaningful way for new people to enter the field, as well as for practitioners to hone their skills (it's also a great way to get a job at OpenAI). Many will require inventing new ideas. Please email us with questions or solutions you'd like us to publicize!

(Also, if you don't have deep learning background but want to learn to solve problems like these, please apply for our Fellowship program!)

⭐⭐⭐ **Parameter Averaging in Distributed RL.** Explore the effect of parameter averaging schemes on sample complexity and amount of communication in RL algorithms. While the simplest solution is to average the gradients from every worker on every update, you can save on communication bandwidth by independently updating workers and then infrequently averaging parameters. In RL, this may have another benefit: at any given time we'll have agents with different parameters, which could lead to better exploration behavior. Another possibility is use algorithms like EASGD that bring parameters partly together each update.

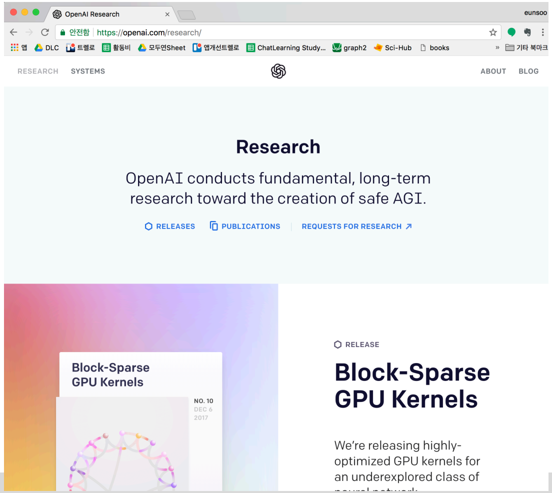
⭐⭐⭐ **Transfer Learning Between Different Games via Generative Models.** Proceed as follows:

- Train 11 good policies for 11 Atari games. Generate 10,000 trajectories of 1,000 steps each from the policy for each game.
- Fit a generative model (such as the Transformer) to the trajectories produced by 10 of the games.
- Then fine-tune that model on the 11th game.
- Your goal is to quantify the benefit from pre-training on the 10 games. How large does the model need to be for the pre-training to be useful? How does the size of the effect change when the amount of data from the 11th game is reduced by 10x? By 100x?

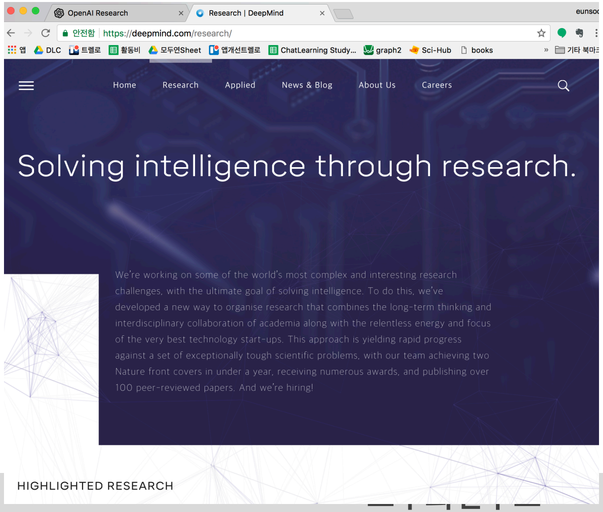
제가 하는 것들...

- OpenAI research나 Deep Mind 의 research 도 좋습니다

https://openai.com/research/



https://deepmind.com/research/

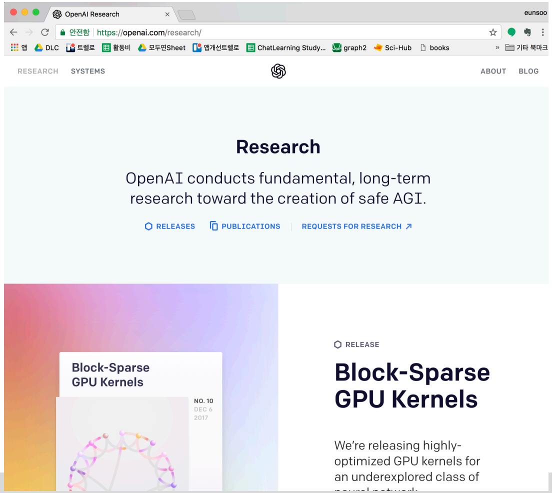


제가 하는 것들...

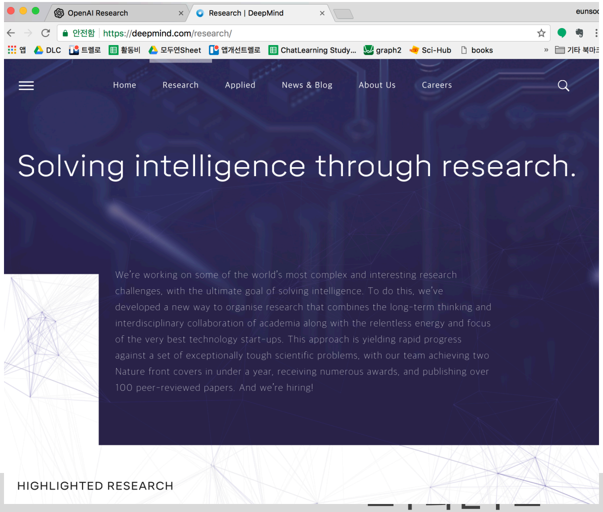
모두의연구소

- 이들의 연구주제는 대부분 Artificial General Intelligence 입니다 (AGI)

https://openai.com/research/



https://deepmind.com/research/



제가 하는 것들... 정리

- 메일로 소식 받기
  - The Wild Week in AI
  - AI valley
  - Gitxiv
- 페이스북 커뮤니티 활동하기
- 트위터로 유명인, 유명회사, 유명 컨퍼런스 팔로윙 하기
- 기타
  - 새로운 형태의 인터렉티브 논문 : Distill
  - 논문결재 돈 요구 : SCI-hub (http://sci-hub.tw/)
  - OpenAI, DeepMind 등의 연구 따라가 보기
  - Research 2.0 주제 확인해 보기

다른 분들은 어떻게 하시나요?

모두의연구소

공유해 주세요~~!!!!

지난시간 Review

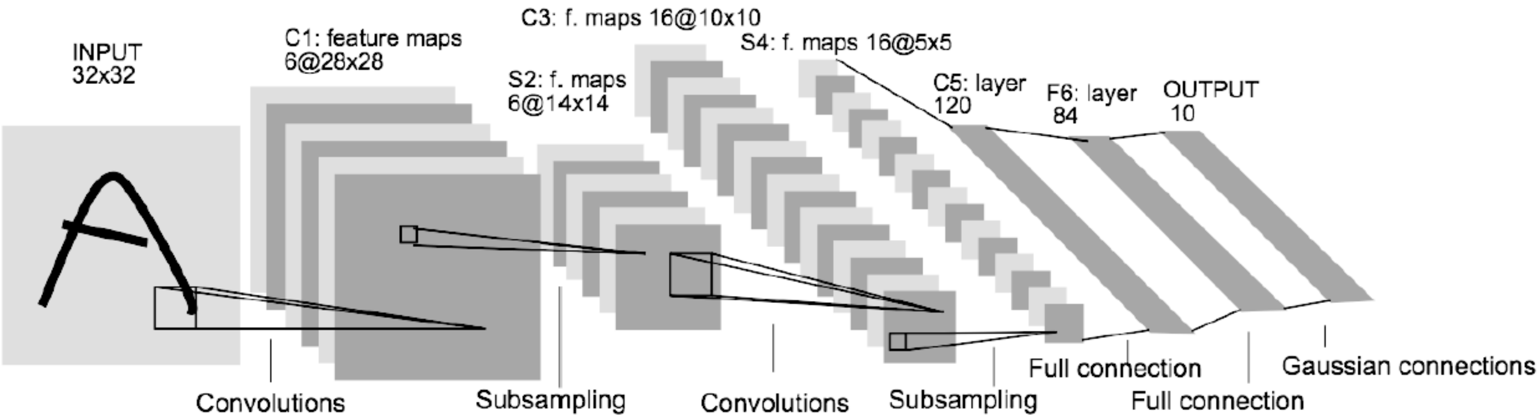
모두의연구소
박은수 Research Director

지난시간 돌아보기 …

- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

지난시간 돌아보기 ...

- Convolutional Neural Networks



지난시간 돌아보기 ...

모두의연구소

- **Momentum**

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta\frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

Gradient 이동누적 스러운 방법

업데이트 1) $v_1 \leftarrow \alpha * 0 - K_o : -K_o$
업데이트 2) $v_2 \leftarrow \alpha v_1 - K_1 : -\alpha K_o - K_1$
업데이트 3) $v_3 \leftarrow \alpha v_2 - K_2 : -\alpha^2 K_o - \alpha K_1 - K_2$
업데이트 4) $v_4 \leftarrow \alpha v_3 - K_3 : -\alpha^3 K_o - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1) $W \leftarrow W + v_1$
- 2) $W \leftarrow W + v_2$
- 3) $W \leftarrow W + v_3$
- 4) $W \leftarrow W + v_4$

- **Adagrad**

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\frac{1}{\sqrt{\boxed{K_0^2}}}K_o$
업데이트 2) $\frac{1}{\sqrt{\boxed{K_1^2+K_0^2}}}K_1$
업데이트 3) $\frac{1}{\sqrt{\boxed{K_3^2+K_1^2+K_0^2}}}K_3$
업데이트 4) $\frac{1}{\sqrt{\boxed{K_4^2+K_3^2+K_1^2+K_0^2}}}K_4$

Gradient Normalization 스러운 방법

- **RMSprop**

$$\mathbf{h} \leftarrow \alpha\mathbf{h} + (1-\alpha)(\frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}})$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\mathbf{h}_1 = (1-a)\mathbf{K}_1^2$
업데이트 2) $\mathbf{h}_2 = a(1-a)\mathbf{K}_1^2 + (1-a)\mathbf{K}_2^2$
업데이트 3) $\mathbf{h}_3 = a^2(1-a)\mathbf{K}_1^2 + a(1-a)\mathbf{K}_2^2 + (1-a)\mathbf{K}_3^2$
업데이트 4) $\mathbf{h}_4 = a^3(1-a)\mathbf{K}_1^2 + a^2(1-a)\mathbf{K}_2^2 + a(1-a)\mathbf{K}_3^2 + (1-a)\mathbf{K}_4^2$

지난시간 돌아보기 ...

모두의연구소

- Momentum

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta\frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

업데이트 1) $v_1 \leftarrow \alpha * 0 - K_o : -K_o$
업데이트 2) $v_2 \leftarrow \alpha v_1 - K_1 : -\alpha K_o - K_1$
업데이트 3) $v_3 \leftarrow \alpha v_2 - K_2 : -\alpha^2 K_o - \alpha K_1 - K_2$
업데이트 4) $v_4 \leftarrow \alpha v_3 - K_3 : -\alpha^3 K_o - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1) W ← W + v_1
- 2) W ← W + v_2
- 3) W ← W + v_3
- 4) W ← W + v_4

- Adagrad

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\frac{1}{\sqrt{\boxed{K_0^2}}}K_o$
업데이트 2) $\frac{1}{\sqrt{\boxed{K_1^2+K_0^2}}}K_1$
업데이트 3) $\frac{1}{\sqrt{\boxed{K_3^2+K_1^2+K_0^2}}}K_3$
업데이트 4) $\frac{1}{\sqrt{\boxed{K_4^2+K_3^2+K_1^2+K_0^2}}}K_4$

두 방법의 같이쓰자
Adam

- RMSprop

$$\mathbf{h} \leftarrow \alpha\mathbf{h} + (1-\alpha)(\frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}})$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\mathbf{h}_1 = (1-a)\mathbf{K}_1^2$
업데이트 2) $\mathbf{h}_2 = a(1-a)\mathbf{K}_1^2 + (1-a)\mathbf{K}_2^2$
업데이트 3) $\mathbf{h}_3 = a^2(1-a)\mathbf{K}_1^2 + a(1-a)\mathbf{K}_2^2 + (1-a)\mathbf{K}_3^2$
업데이트 4) $\mathbf{h}_4 = a^3(1-a)\mathbf{K}_1^2 + a^2(1-a)\mathbf{K}_2^2 + a(1-a)\mathbf{K}_3^2 + (1-a)\mathbf{K}_4^2$

지난시간 돌아보기 ...

모두의연구소

- Adam
  - RMSProp + 모멘텀

Adam update

[Kingma and Ba, 2014]

(incomplete, but close)

```
# Adam
m = beta1*m + (1-beta1)*dx # update first moment
v = beta2*v + (1-beta2)*(dx**2) # update second moment
x += - learning_rate * m / (np.sqrt(v) + 1e-7)
```

momentum

RMSProp-like

Looks a bit like RMSProp with momentum

```
# RMSProp
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

지난시간 돌아보기 ...

- Adam
  - RMSProp + 모멘텀

Adam update

[Kingma and Ba, 2014]

(incomplete, but close)

$$c_N = \alpha c_{N-1} + (1-\alpha)x_N$$

```
# Adam
m = beta1*m + (1-beta1)*dx # update first moment
v = beta2*v + (1-beta2)*(dx**2) # update second moment
x += - learning_rate * m / (np.sqrt(v) + 1e-7)
```

momentum

RMSProp-like

많이 등장하는 패턴이군요

Looks a bit like RMSProp with momentum

```
# RMSProp
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

평균을 구해보자

N개의 sample에 대한
평균 c_N 을 구해보자.

$$c_N = \frac{1}{N}(x_1 + x_2 + x_3 + \cdots + x_N)$$
$$= \frac{1}{N}\sum_{i=1}^{N} x_i$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$
$$= \frac{1}{N}\left(\boxed{\sum_{i=1}^{N-1} x_i} + \boxed{x_N}\right)$$
$$= \boxed{\frac{N-1}{N}}\underbrace{\boxed{\frac{1}{N-1}}\boxed{\sum_{i=1}^{N-1} x_i}}_{c_{N-1}} + \frac{1}{N}x_N$$
$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

$$c_{N-1}$$

$$= \frac{N-1}{N}\boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i} + \frac{1}{N}x_N$$

$$= \alpha \boxed{c_{N-1}} + (1-\alpha)x_N \quad 0 \langle \alpha \langle 1$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

c_{N-1}

$$= \boxed{\frac{N-1}{N}}\,\boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i} + \frac{1}{N}x_N$$

$$= \boxed{\alpha}\,\boxed{c_{N-1}} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N}$$

$$1 - \alpha = 1 - \frac{N-1}{N}$$

모두의연구소

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

$$= \boxed{\frac{N-1}{N}}\ \boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i} + \boxed{\frac{1}{N}}\, x_N$$

c_{N-1}

$$= \boxed{\alpha}\,\boxed{c_{N-1}} + \boxed{(1-\alpha)}\, x_N \qquad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N}$$

$$1 - \alpha = 1 - \frac{N-1}{N}$$

$$1 - \alpha = \frac{N}{N} - \frac{N-1}{N} = \frac{1}{N}$$

모두의연구소

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\boxed{\sum_{i=1}^{N-1} x_i} + \boxed{x_N}\right)$$

$$= \boxed{\frac{N-1}{N}}\underbrace{\boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i}}_{c_{N-1}} + \frac{1}{N}x_N$$

$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N} = 1 - \frac{1}{N}$$

α 가 크다는 것은?

N이 ?

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\boxed{\sum_{i=1}^{N-1} x_i} + \boxed{x_N}\right)$$

$$= \boxed{\frac{N-1}{N}}\,\boxed{\frac{1}{N-1}}\underbrace{\boxed{\sum_{i=1}^{N-1} x_i}}_{c_{N-1}} + \frac{1}{N}x_N$$

$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N} = 1 - \frac{1}{N}$$

α 가 크다는 것은?

N이 ? 커야 겠군요

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

$$= \frac{N-1}{N}\underbrace{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i}_{c_{N-1}} + \frac{1}{N}x_N$$

$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N} = 1 - \frac{1}{N}$$

α 가 크다는 것은?

N이 ? 커야 겠군요

1. 더 많은 이동평균을 고려한 관점이라고 해석할 수 있을 것 같아요

제 생각이에요 ...

2. 이전의 평균값에 더 많은 가중치를 준 합이라고 볼 수 있네요 (수식 그대로 해석)

지난시간 돌아보기 ...

모두의연구소

Batch Normalization

[Ioffe and Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Note: at test time BatchNorm layer functions differently:

The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used.

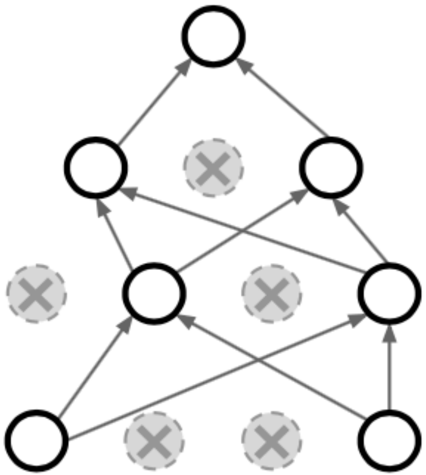
(e.g. can be estimated during training with running averages)

지난시간 돌아보기 ...

모두의연구소

Regularization: Dropout

How can this possibly be a good idea?



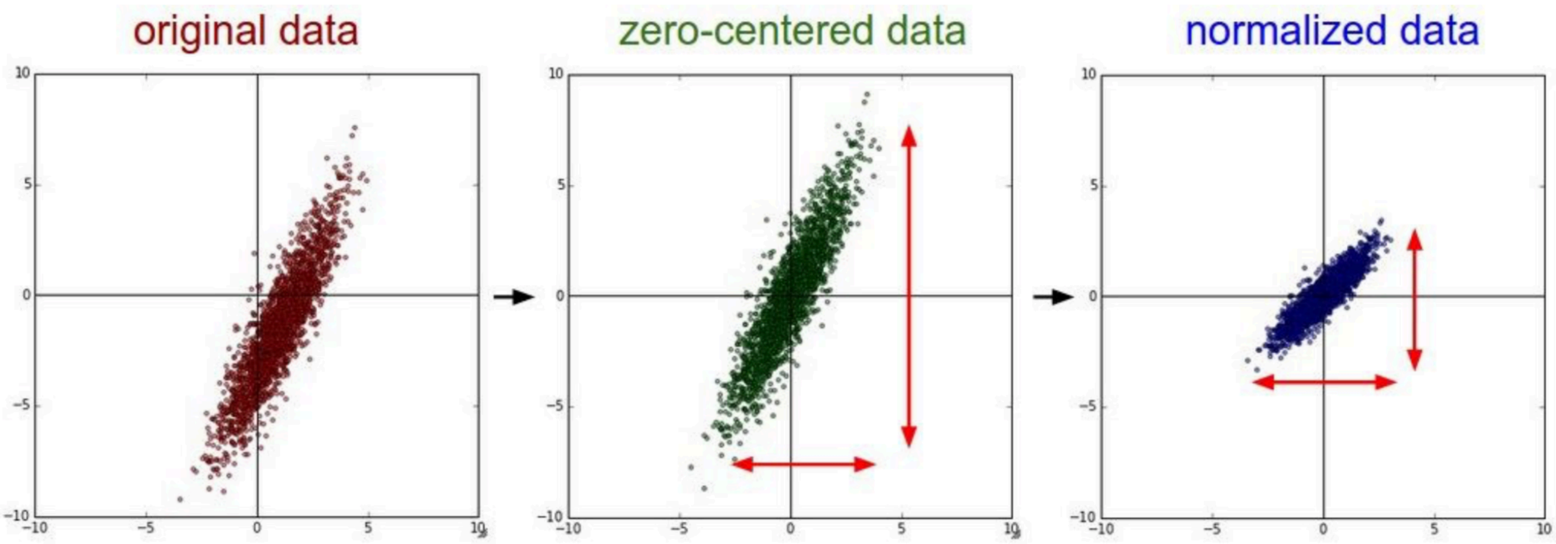
Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

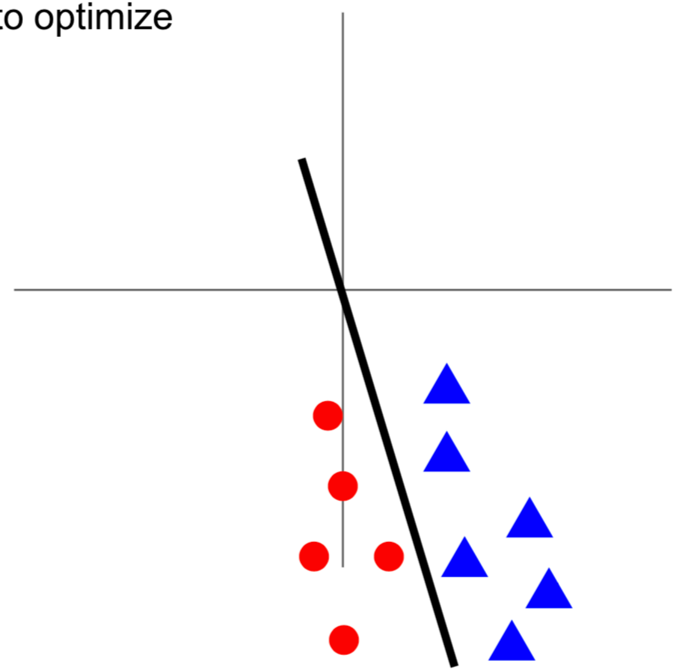
An FC layer with 4096 units has $2^{4096} \sim 10^{1233}$ possible masks!
Only $\sim 10^{82}$ atoms in the universe...

Last time: Data Preprocessing

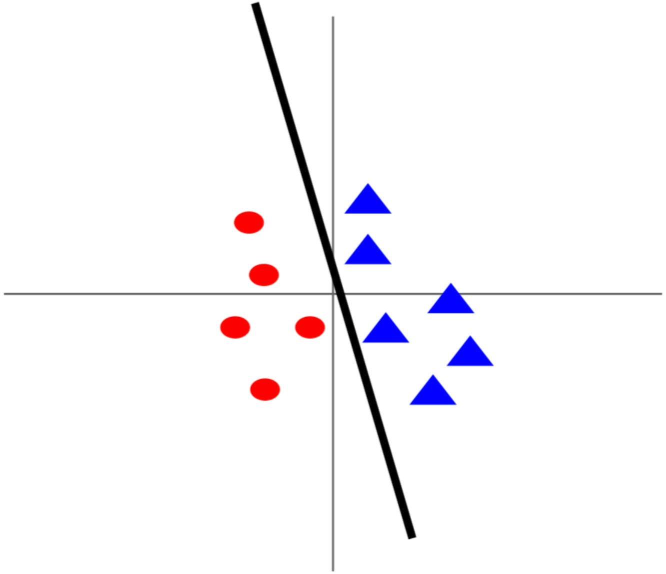


Last time: Data Preprocessing

Before normalization: classification loss very sensitive to changes in weight matrix; hard to optimize



After normalization: less sensitive to small changes in weights; easier to optimize



지난시간 돌아보기 ...

Summary

TLDRs

We looked in detail at:

- Activation Functions (use ReLU)
- Data Preprocessing (images: subtract mean)
- Weight Initialization (use Xavier init)
- Batch Normalization (use)
- Babysitting the Learning process
- Hyperparameter Optimization
  (random sample hyperparams, in log space when appropriate)

현재 우리는

6주 동안의 진행계획

- 1주 : Loss Function and Optimization
- 2주 : Introduction to Neural Networks
- 3주 : Convolutional Neural Networks
- 4주 : Training Neural Networks 1
- 5주 : Training Neural Networks 2
- 6주 : Recurrent Neural Networks

이미 기초를 저보다도 많이 알고 계신분들도 있습니다

그렇지만 이 강의는 지금 딥러닝을
시작하는 분들을 위한 강의입니다

Training Neural Networks - 2

모두의연구소
박은수 Research Director

오늘의 계획

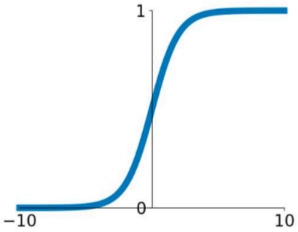
모두의연구소

- 좀 더 자세히 (from : cs231n)
  - Activation Functions
  - Fancier Optimization
  - Regularization : Data augmentation
- Transfer Learning

Activation Functions

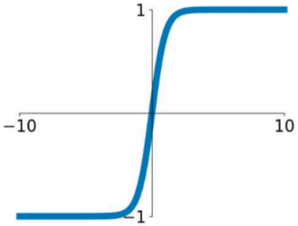
Sigmoid

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



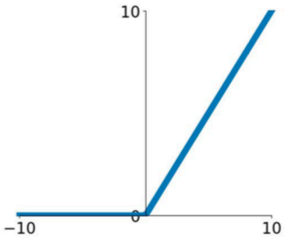
tanh

$\tanh(x)$



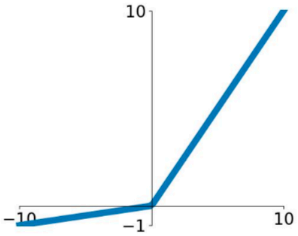
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

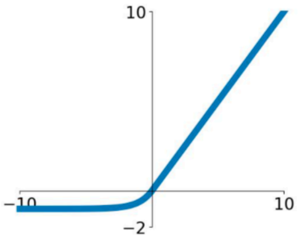


Maxout

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

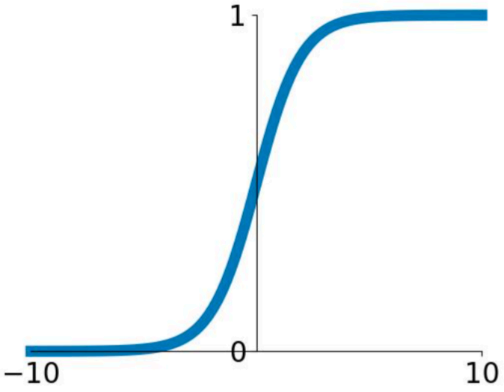
ELU

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



Activation Functions

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

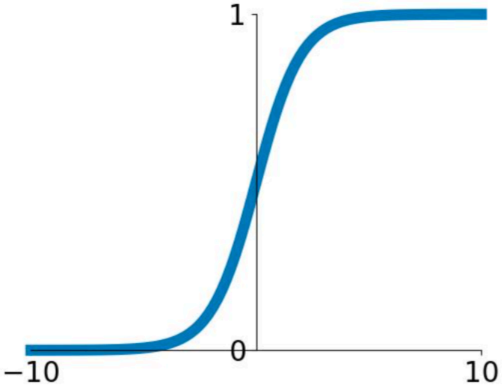


Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

Activation Functions

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

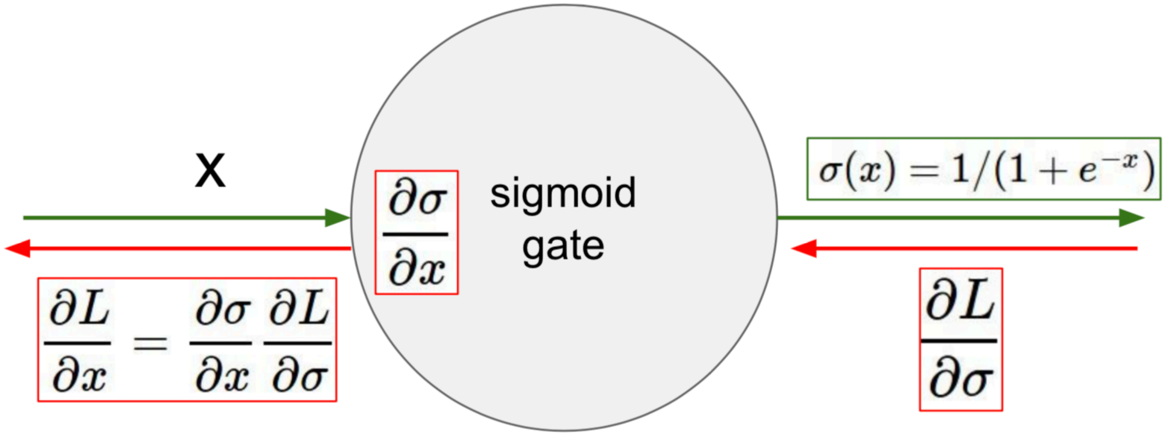


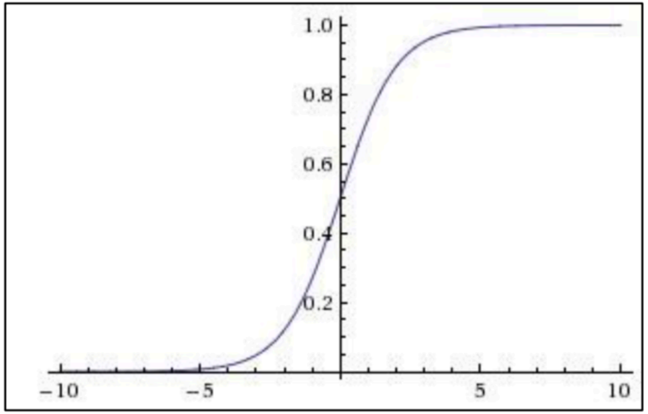
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients

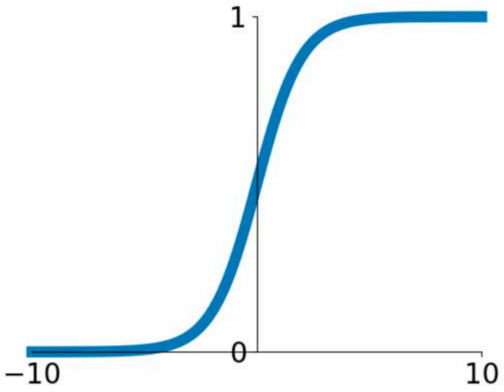




What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

Activation Functions

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



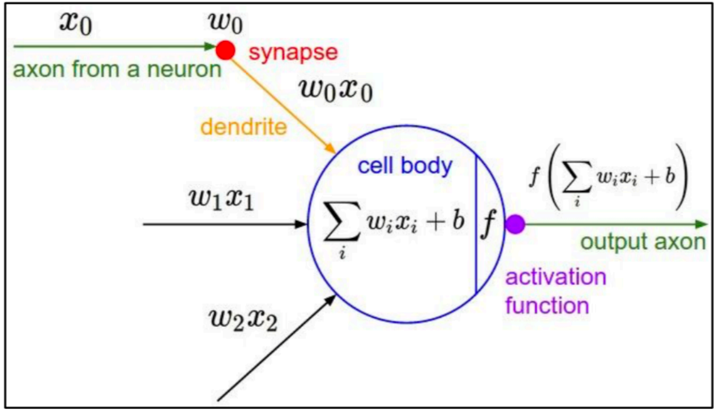
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered

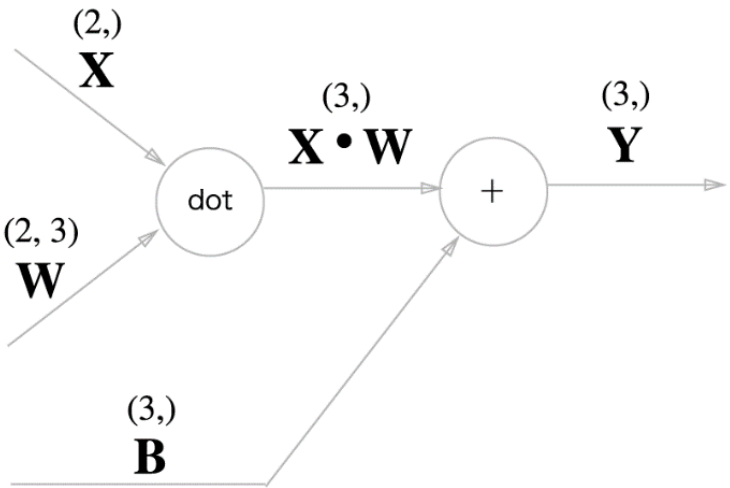
Consider what happens when the input to a neuron (x) is always positive:



$$f\left(\sum_i w_i x_i + b\right)$$

What can we say about the gradients on **w**?

역전파



$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\boxed{\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}}$$

복습 : Computational Graph

- **내적연산**

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \ \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

Gradient **w**

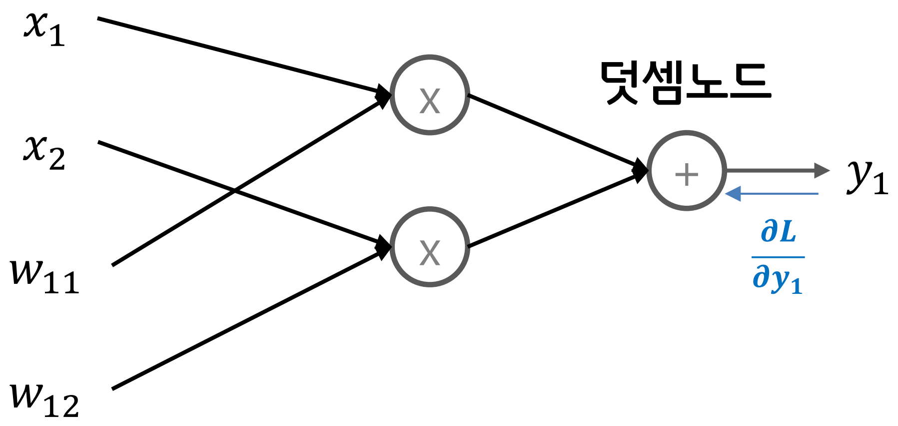
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



같은 벡터단위로 살펴봅시다

복습 : Computational Graph

모두의연구소

- Affine 계층

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$[x_1 \ \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

Gradient **w**

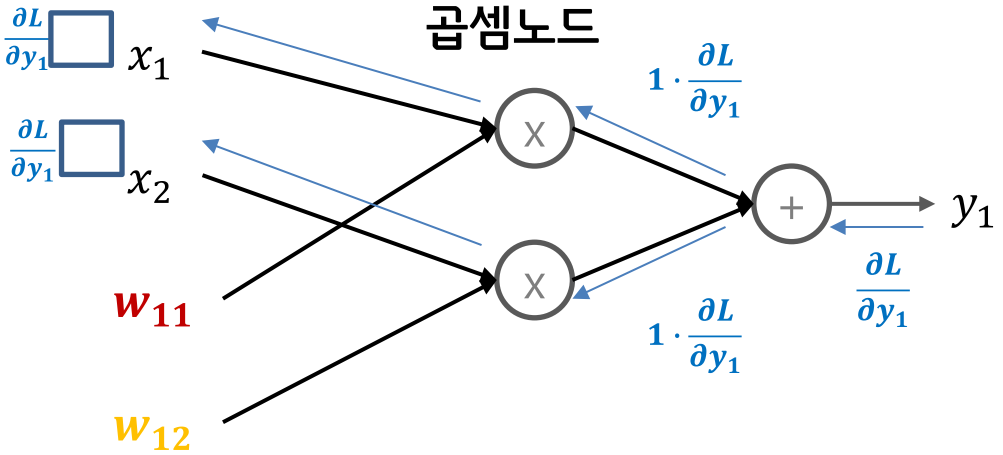
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \ \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{x}} = \left[\frac{\partial L}{\partial y_1} w_{11} \quad \frac{\partial L}{\partial y_1} w_{12}\right] = \frac{\partial L}{\partial y_1} \cdot [w_{11} \quad w_{12}] = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

Gradient **w**

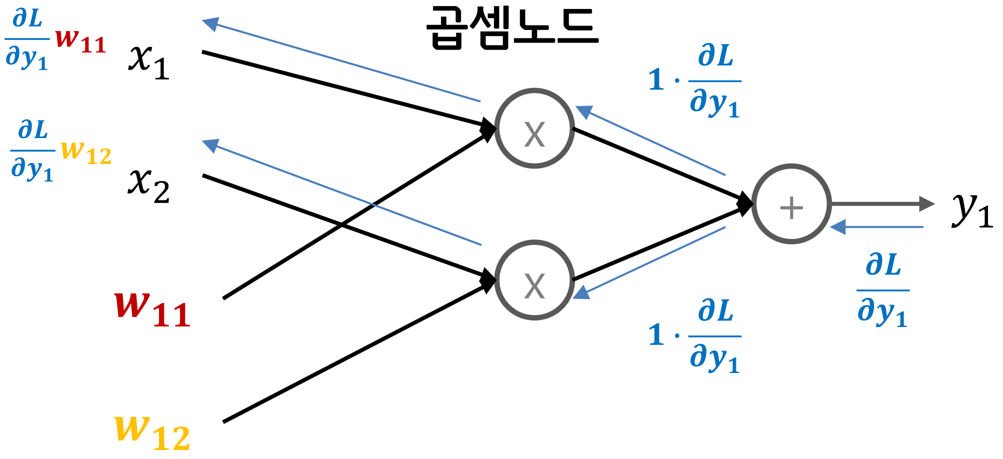
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$

같은형태로 나오게 $\boxed{[x_1 \;\; x_2]} \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$

$w_{11}x_1 + w_{12}x_2 = y_1$

$$\frac{\partial L}{\partial \mathrm{x}} = \boxed{\left[\frac{\partial L}{\partial y_1} w_{11} \quad \frac{\partial L}{\partial y_1} w_{12}\right]} = \frac{\partial L}{\partial y_1} \cdot [w_{11} \quad w_{12}] = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

Gradient **w**

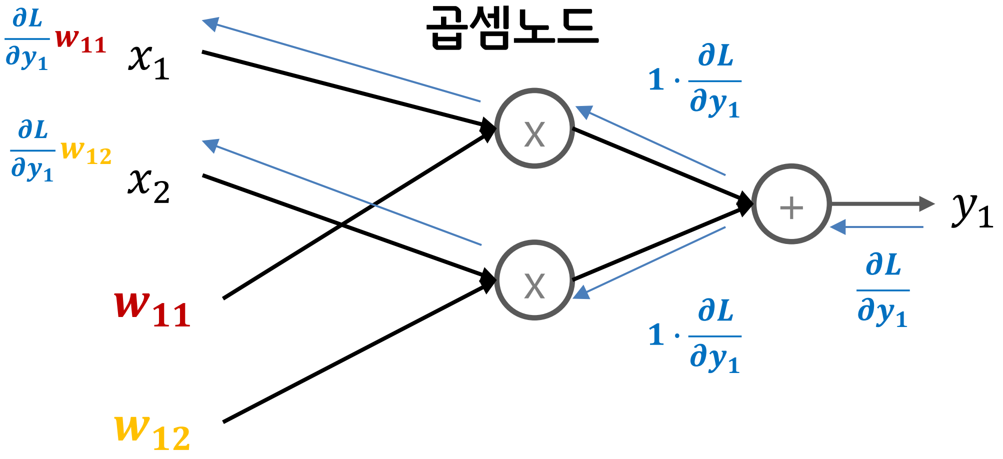
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

모두의연구소

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \;\; x_2] \times \boxed{\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix}} = y_1$$

같은형태로 나오게

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{x}} = \left[\frac{\partial L}{\partial y_1} w_{11} \quad \frac{\partial L}{\partial y_1} w_{12}\right] = \frac{\partial L}{\partial y_1} \cdot \boxed{[w_{11} \quad w_{12}]} = \frac{\partial L}{\partial y_1} \cdot \boxed{\mathbf{w}^{\mathrm{T}}}$$

Gradient **w**

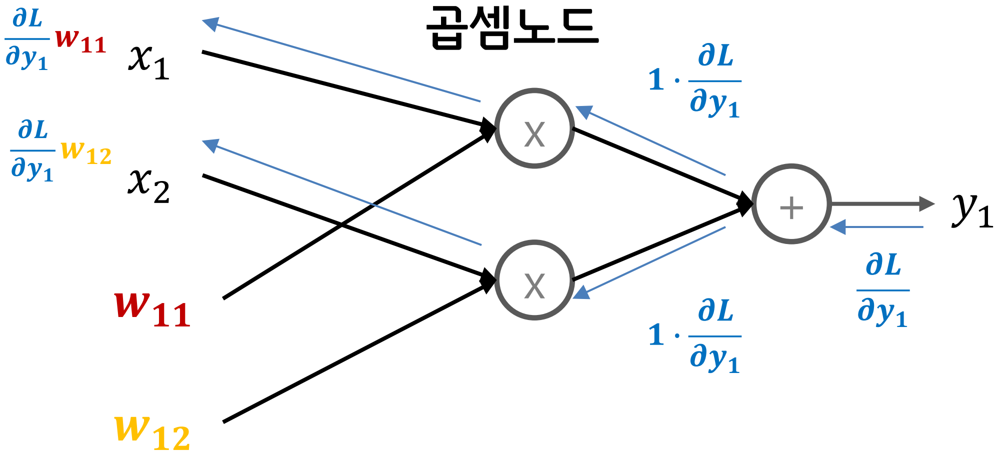
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \ \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\boxed{\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}}$$

Gradient **w**

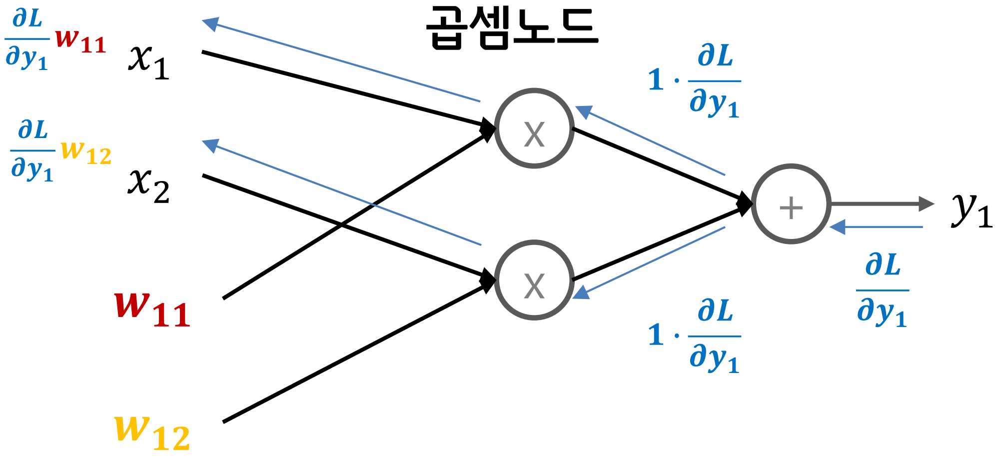
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



- Affine 계층

$$x^{T}\mathbf{w} = \mathbf{y}$$

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{T}$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

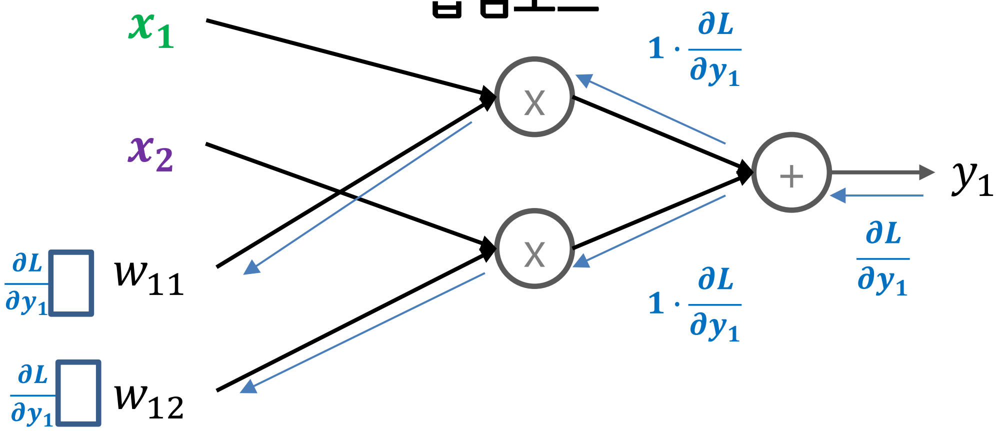
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{w}} = \begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 \\ \frac{\partial L}{\partial y_1} x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \frac{\partial L}{\partial y_1} = \mathrm{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

Gradient w

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

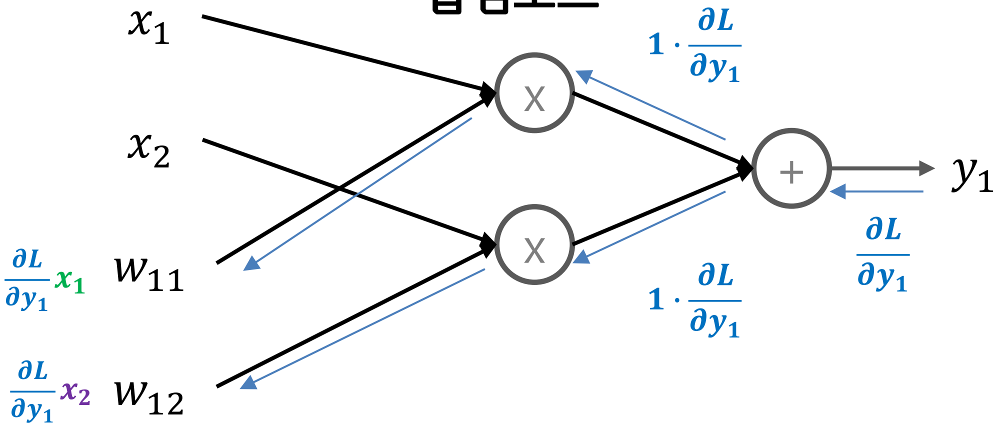
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient x

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

같은형태로 나오게

$$[x_1 \;\; x_2] \times \boxed{\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix}} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{w}} = \boxed{\begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 \\ \frac{\partial L}{\partial y_1} x_2 \end{bmatrix}} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \frac{\partial L}{\partial y_1} = \mathbf{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

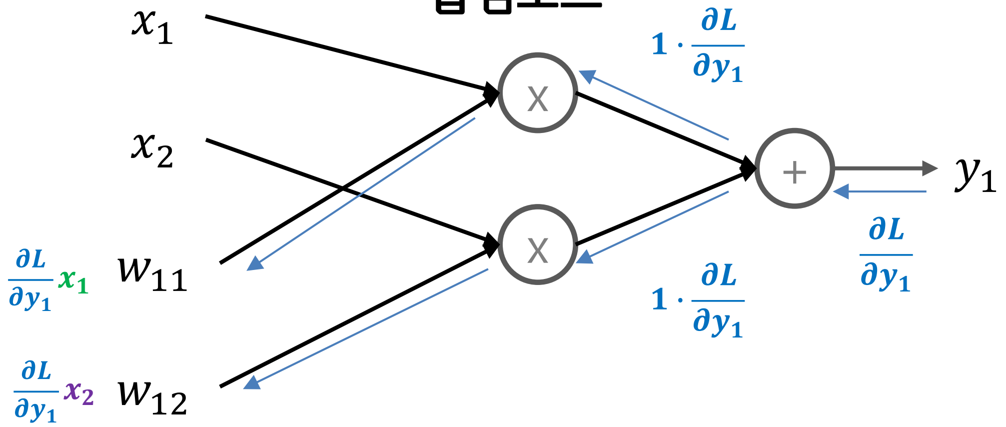
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$\boxed{[x_1\ \ x_2]} \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

같은 표현으로

$$\frac{\partial L}{\partial \mathrm{w}} = \begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 \\ \frac{\partial L}{\partial y_1} x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \frac{\partial L}{\partial y_1} = \boxed{\mathrm{x}^{\mathrm{T}}} \cdot \frac{\partial L}{\partial y_1}$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

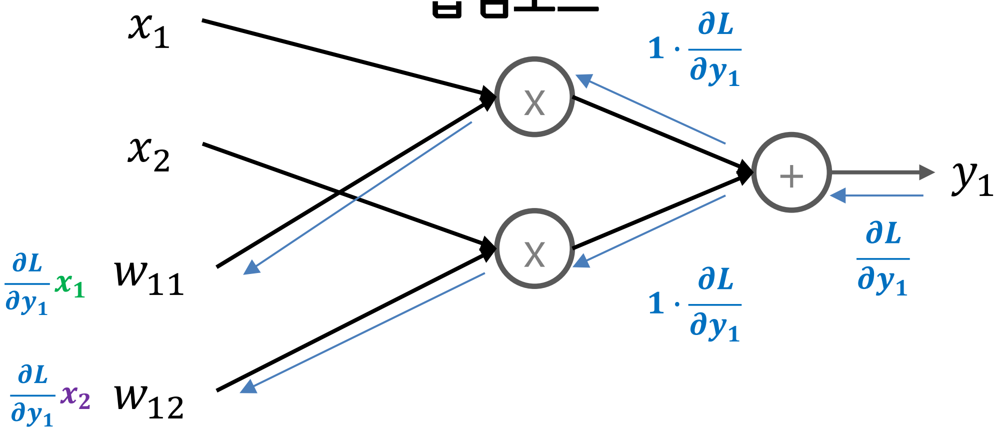
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathrm{w}} = \mathbf{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

Gradient w

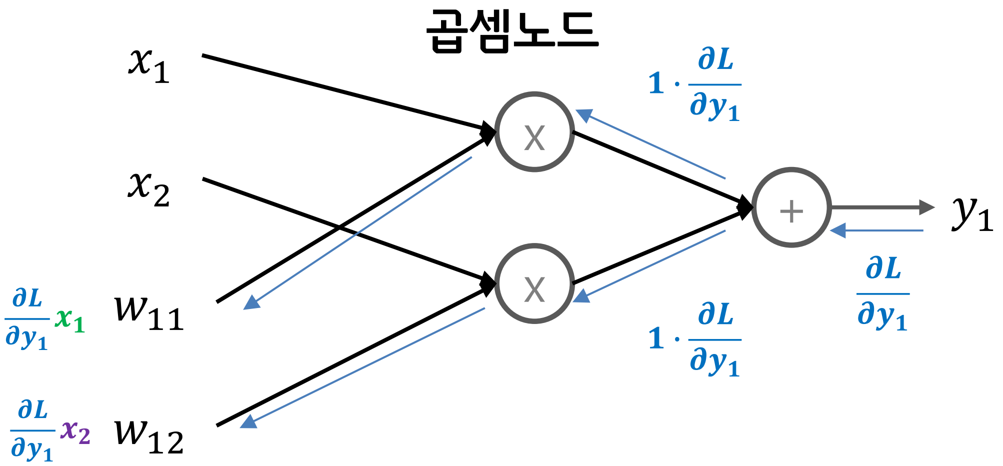
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient x

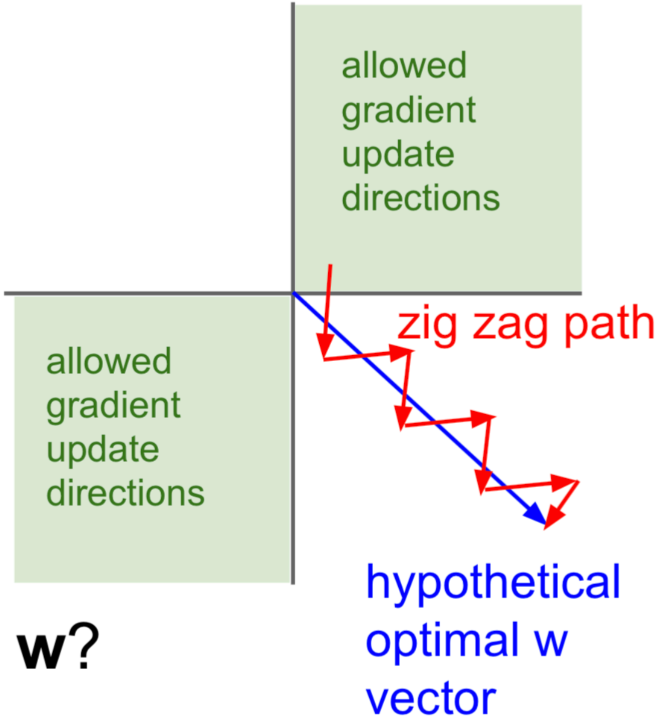
$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



Consider what happens when the input to a neuron is always positive...

$$f\left(\sum_i w_i x_i + b\right)$$



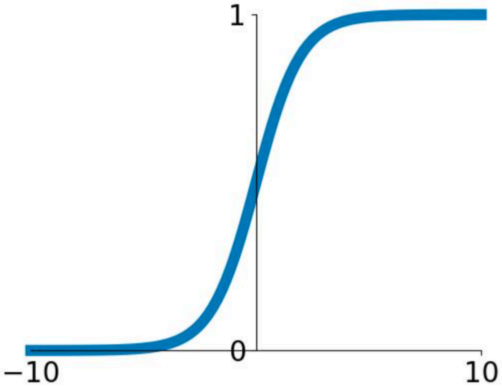
What can we say about the gradients on **w**?

Always all positive or all negative :(

(this is also why you want zero-mean data!)

Activation Functions

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



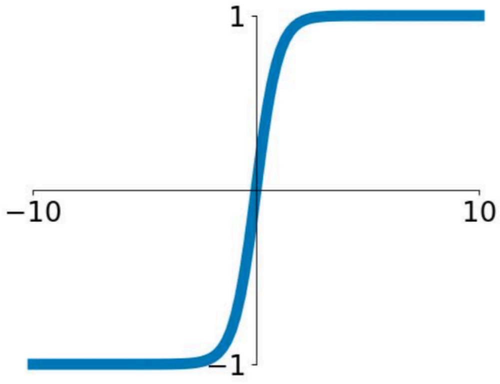
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

Activation Functions

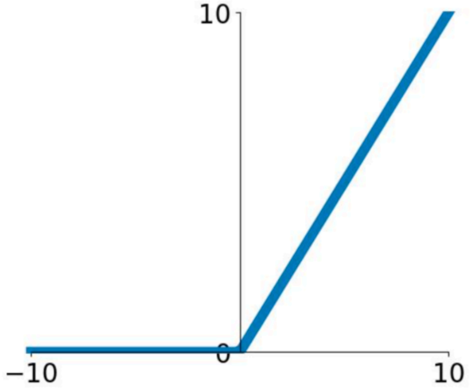


tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

[LeCun et al., 1991]

Activation Functions



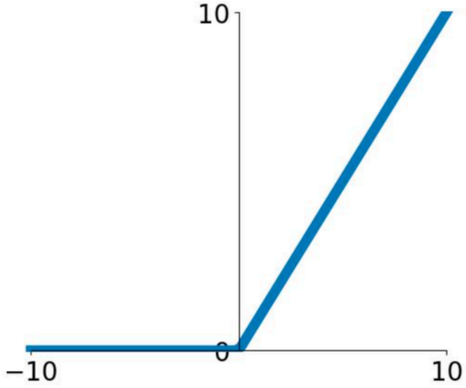
ReLU
(Rectified Linear Unit)

- Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

[Krizhevsky et al., 2012]

Activation Functions



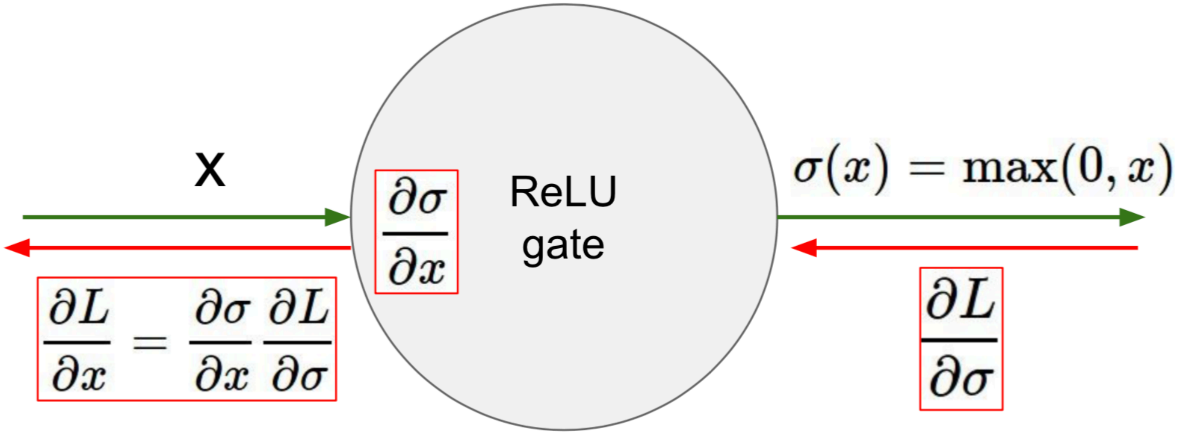
ReLU
(Rectified Linear Unit)

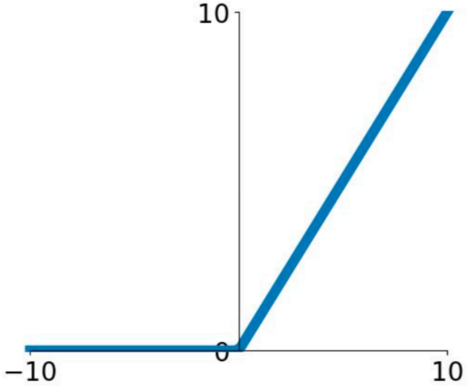
- Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

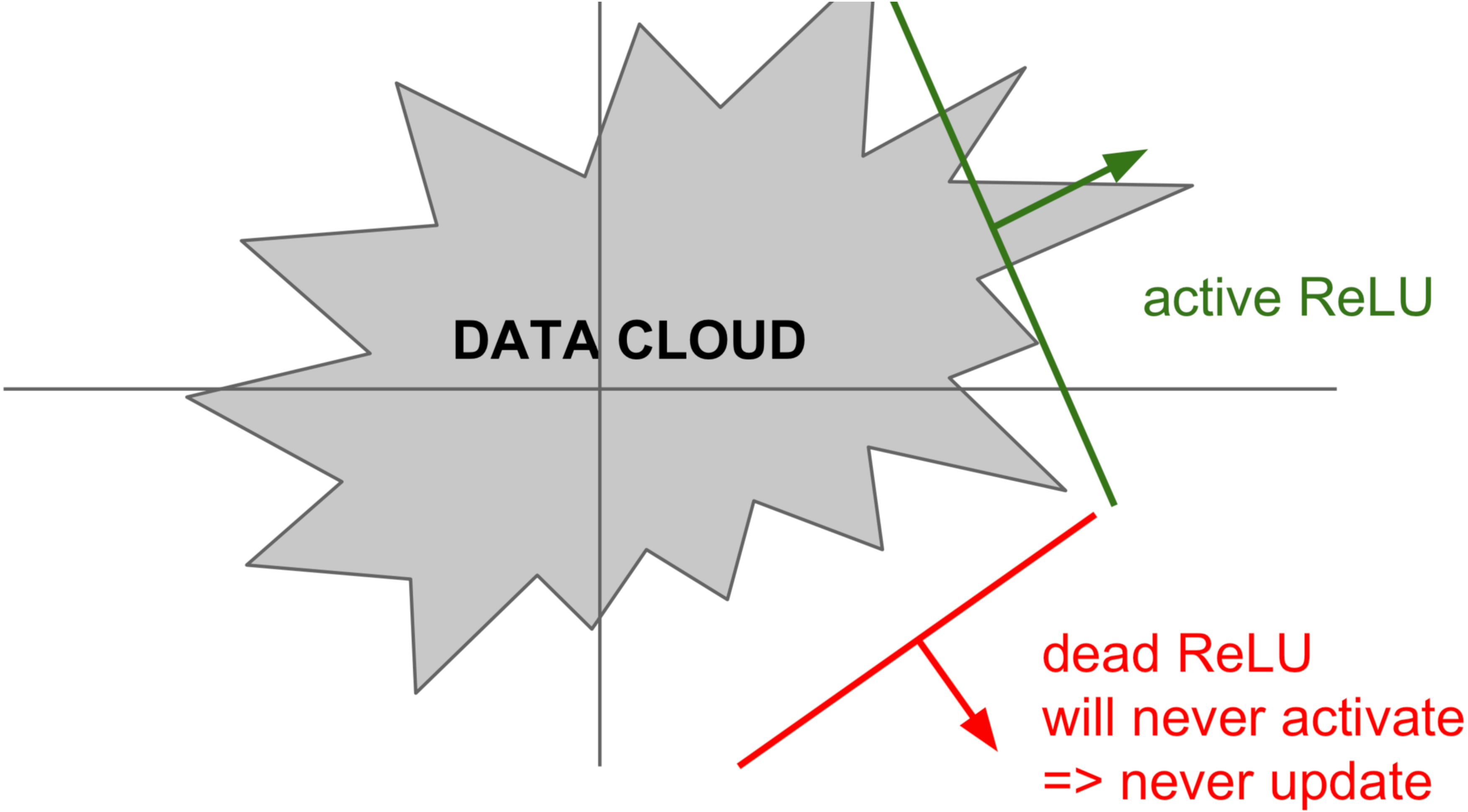
- Not zero-centered output
- An annoyance:

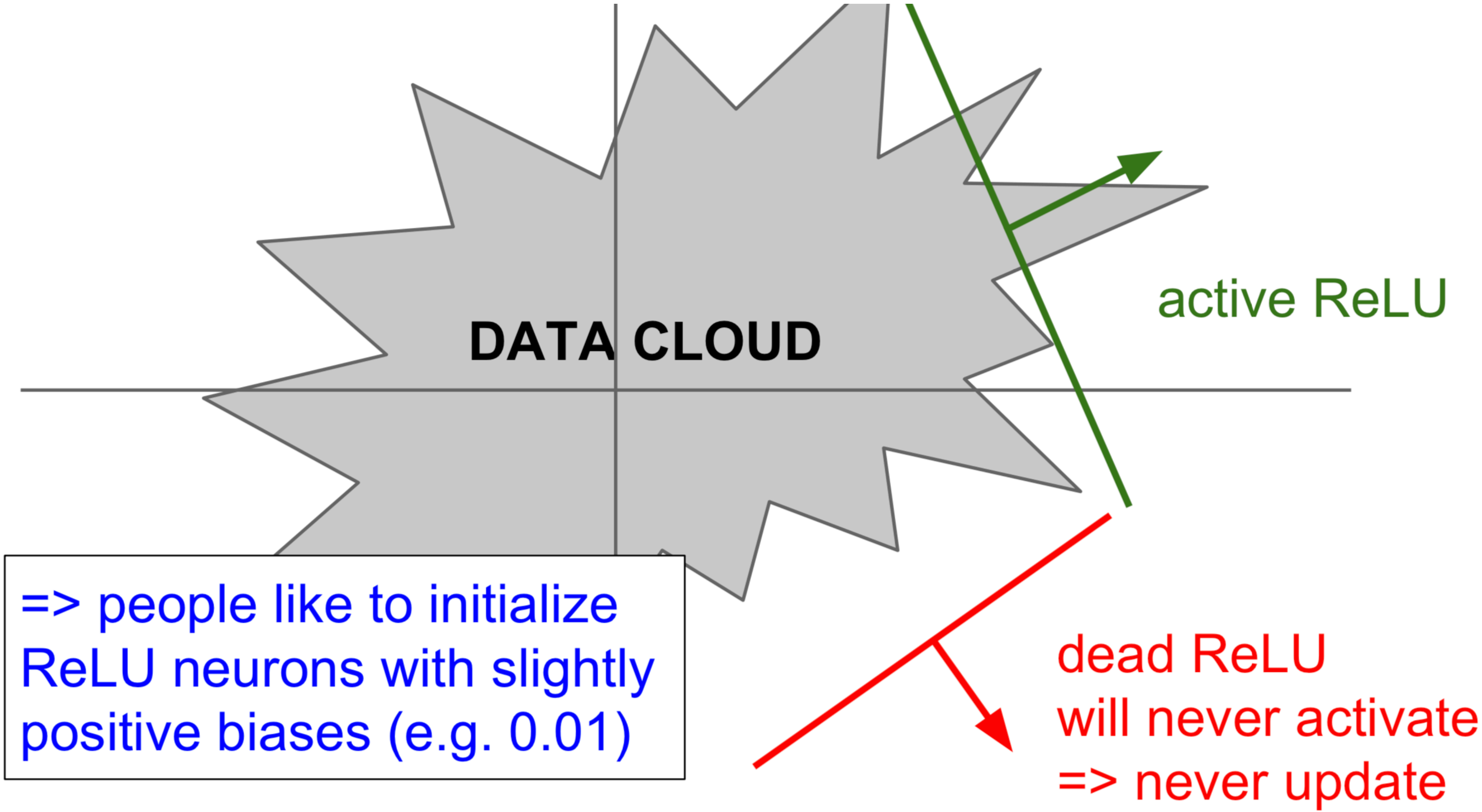
hint: what is the gradient when $x < 0$?





What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

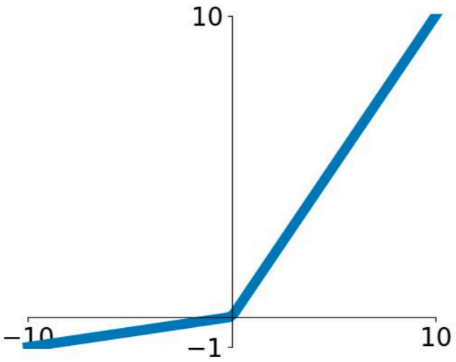

DATA CLOUD
active ReLU
dead ReLU
will never activate
=> never update



=> people like to initialize ReLU neurons with slightly positive biases (e.g. 0.01)

Activation Functions

[Mass et al., 2013]
[He et al., 2015]



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not “die”.**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

업데이트가 발생하지 않음

모두의연구소

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

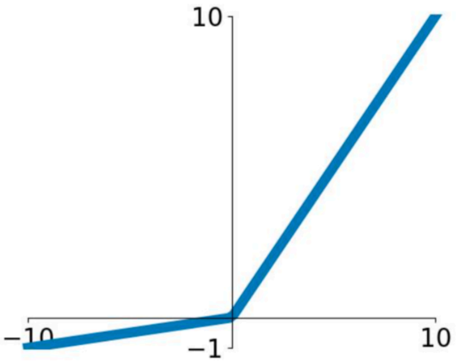
– Gradient 가 0

– Update :

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial L}{\partial \mathbf{w}}$$

Activation Functions

[Mass et al., 2013]
[He et al., 2015]



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not “die”.**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Parametric Rectifier (PReLU)

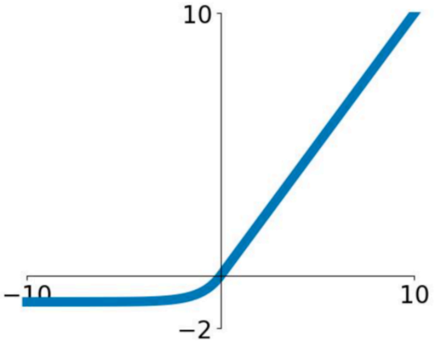
$$f(x) = \max(\alpha x, x)$$

backprop into \alpha
(parameter)

Activation Functions

[Clevert et al., 2015]

Exponential Linear Units (ELU)



- All benefits of ReLU
- Closer to zero mean outputs
- Negative saturation regime compared with Leaky ReLU adds some robustness to noise

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

- Computation requires exp()

Maxout "Neuron"

[Goodfellow et al., 2013]

- Does not have the basic form of dot product -> nonlinearity
- Generalizes ReLU and Leaky ReLU
- Linear Regime! Does not saturate! Does not die!

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

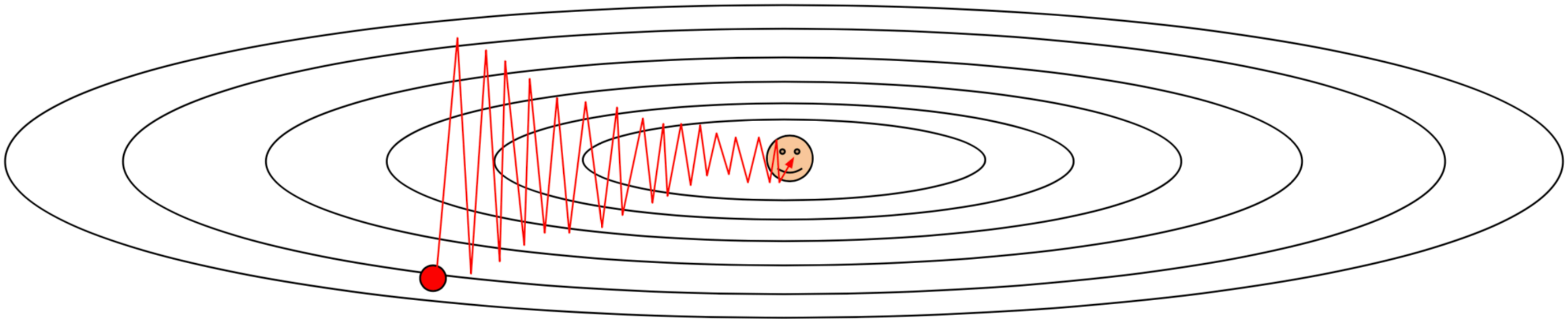
Problem: doubles the number of parameters/neuron :(

TLDR: In practice:

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU
- Try out tanh but don’t expect much
- Don’t use sigmoid

Optimization: Problems with SGD

What if loss changes quickly in one direction and slowly in another?
What does gradient descent do?
Very slow progress along shallow dimension, jitter along steep direction



Loss function has high **condition number**: ratio of largest to smallest singular value of the Hessian matrix is large

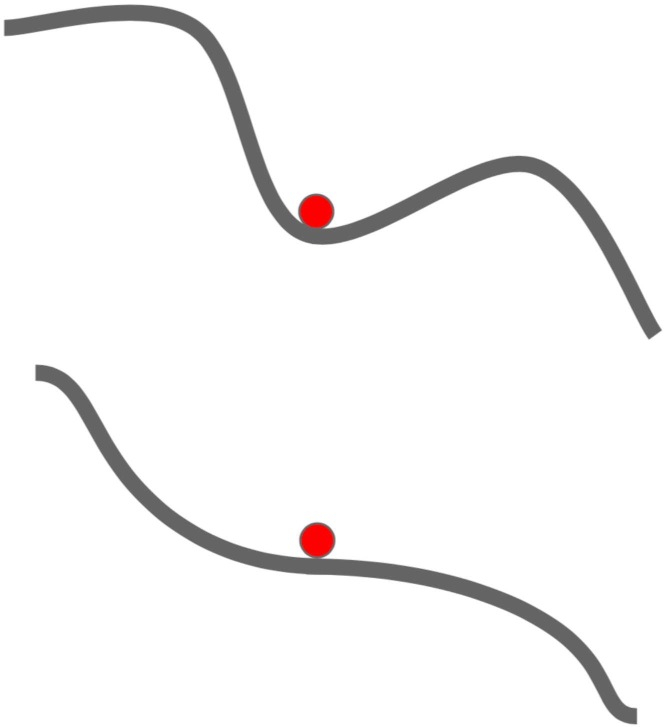
Optimization: Problems with SGD

What if the loss function has a **local minima** or **saddle point**?

Optimization: Problems with SGD

What if the loss function has a **local minima** or **saddle point**?

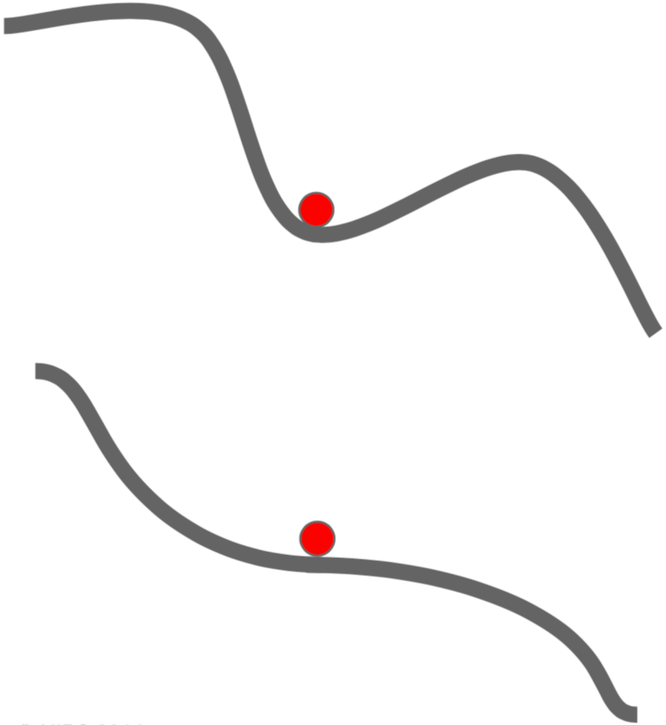
Zero gradient, gradient descent gets stuck



Optimization: Problems with SGD

What if the loss function has a **local minima** or **saddle point**?

Saddle points much more common in high dimension



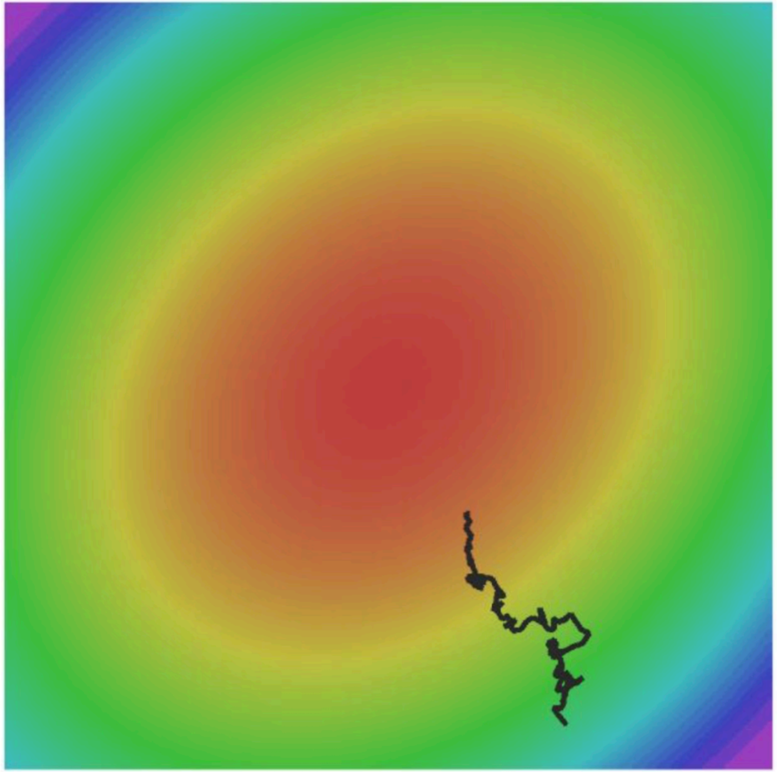
Dauphin et al, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization”, NIPS 2014

Optimization: Problems with SGD

Our gradients come from minibatches so they can be noisy!

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W)$$



SGD + Momentum

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

```
while True:
  dx = compute_gradient(x)
  x += learning_rate * dx
```

SGD+Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

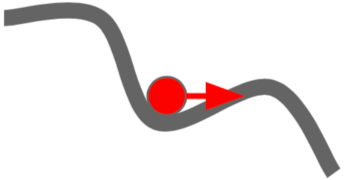
$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
vx = 0
while True:
  dx = compute_gradient(x)
  vx = rho * vx + dx
  x += learning_rate * vx
```

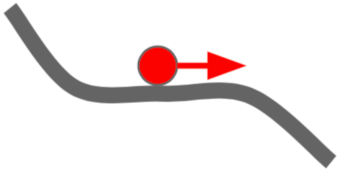
- Build up “velocity” as a running mean of gradients
- Rho gives “friction”; typically rho=0.9 or 0.99

SGD + Momentum

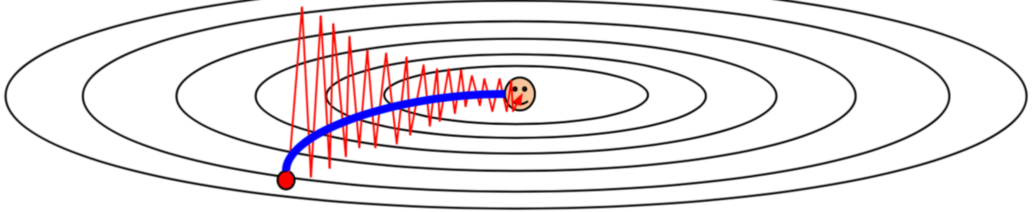
Local Minima



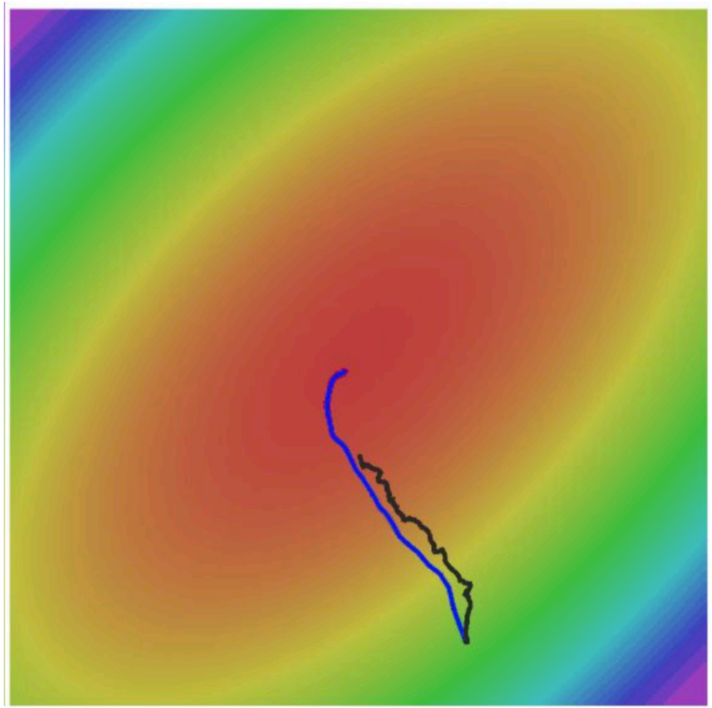
Saddle points



Poor Conditioning

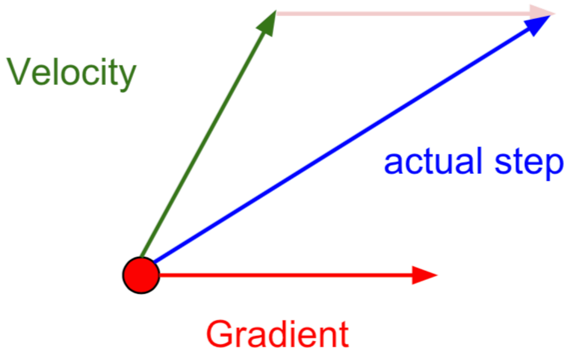


Gradient Noise

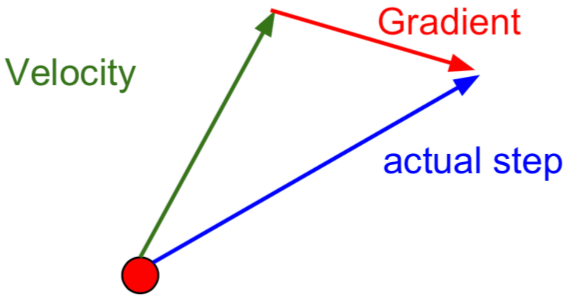


Nesterov Momentum

Momentum update:



Nesterov Momentum

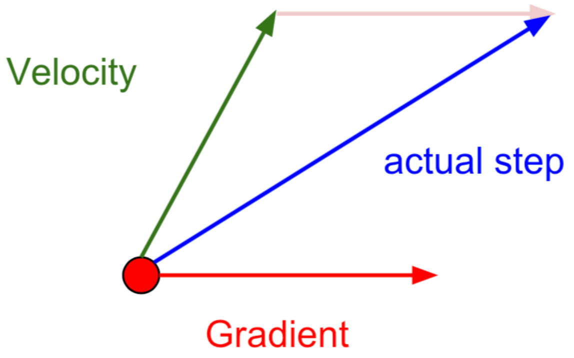


Nesterov, “A method of solving a convex programming problem with convergence rate O(1/k^2)”, 1983
Nesterov, “Introductory lectures on convex optimization: a basic course”, 2004
Sutskever et al, “On the importance of initialization and momentum in deel learning”, ICML 2013

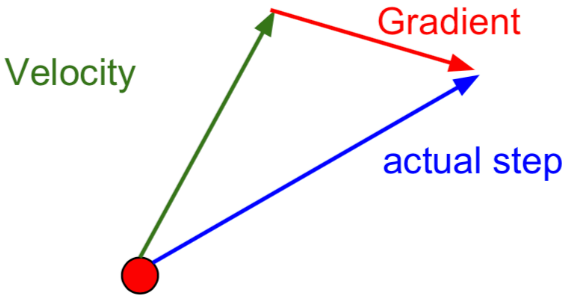
Nesterov Momentum

SGD+Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$



$$\boxed{\begin{aligned} v_{t+1} &= \rho v_t - \alpha \nabla f(x_t + \rho v_t) \\ x_{t+1} &= x_t + v_{t+1} \end{aligned}}$$



Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k^2)", 1983
Nesterov, "Introductory lectures on convex optimization: a basic course", 2004
Sutskever et al, "On the importance of initialization and momentum in deel learning", ICML 2013

Nesterov Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(\boxed{x_t + \rho v_t})$$
$$x_{t+1} = x_t + v_{t+1}$$

Annoying, usually we want update in terms of $x_t, \nabla f(x_t)$

Change of variables $\tilde{x}_t = x_t + \rho v_t$ and rearrange:

$$
\begin{aligned}
v_{t+1} &= \rho v_t - \alpha \nabla f(\tilde{x}_t) \\
\tilde{x}_{t+1} &= \tilde{x}_t - \rho v_t + (1 + \rho) v_{t+1} \\
&= \tilde{x}_t + v_{t+1} + \rho(v_{t+1} - v_t)
\end{aligned}
$$

```
dx = compute_gradient(x)
old_v = v
v = rho * v - learning_rate * dx
x += -rho * old_v + (1 + rho) * v
```

Nesterov Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(\boxed{x_t + \rho v_t})$$
$$x_{t+1} = x_t + v_{t+1}$$

Annoying, usually we want update in terms of $x_t, \nabla f(x_t)$

Change of variables $\boxed{\tilde{x}_t = x_t + \rho v_t}$ and rearrange:

$$\begin{aligned} v_{t+1} &= \rho v_t - \alpha \nabla f(\tilde{x}_t) \\ \tilde{x}_{t+1} &= \boxed{\tilde{x}_t - \rho v_t} + (1 + \rho) v_{t+1} \\ &= \tilde{x}_t + v_{t+1} + \rho(v_{t+1} - v_t) \end{aligned}$$

```
dx = compute_gradient(x)
old_v = v
v = rho * v - learning_rate * dx
x += -rho * old_v + (1 + rho) * v
```

지난시간 돌아보기 ...

모두의연구소

- Momentum

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta\frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

업데이트 1) $v_1 \leftarrow \alpha * 0 - K_o : -K_o$
업데이트 2) $v_2 \leftarrow \alpha v_1 - K_1 : -\alpha K_o - K_1$
업데이트 3) $v_3 \leftarrow \alpha v_2 - K_2 : -\alpha^2 K_o - \alpha K_1 - K_2$
업데이트 4) $v_4 \leftarrow \alpha v_3 - K_3 : -\alpha^3 K_o - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1) W ← W + v_1
- 2) W ← W + v_2
- 3) W ← W + v_3
- 4) W ← W + v_4

- Adagrad

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\frac{1}{\sqrt{\boxed{K_0^2}}} K_o$

업데이트 2) $\frac{1}{\sqrt{\boxed{K_1^2+K_0^2}}} K_1$

업데이트 3) $\frac{1}{\sqrt{\boxed{K_3^2+K_1^2+K_0^2}}} K_3$

업데이트 4) $\frac{1}{\sqrt{\boxed{K_4^2+K_3^2+K_1^2+K_0^2}}} K_4$

두 방법의 같이쓰자

Adam

- RMSprop

$$\mathbf{h} \leftarrow \alpha\mathbf{h} + (1-\alpha)(\frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}})$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\mathbf{h}_1 = (1-a)\mathbf{K}_1^2$
업데이트 2) $\mathbf{h}_2 = a(1-a)\mathbf{K}_1^2 + (1-a)\mathbf{K}_2^2$
업데이트 3) $\mathbf{h}_3 = a^2(1-a)\mathbf{K}_1^2 + a(1-a)\mathbf{K}_2^2 + (1-a)\mathbf{K}_3^2$
업데이트 4) $\mathbf{h}_4 = a^3(1-a)\mathbf{K}_1^2 + a^2(1-a)\mathbf{K}_2^2 + a(1-a)\mathbf{K}_3^2 + (1-a)\mathbf{K}_4^2$

Adam (almost)

```
first_moment = 0
second_moment = 0
while True:
  dx = compute_gradient(x)
  first_moment = beta1 * first_moment  + (1 - beta1) * dx
  second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
  x -= learning_rate * first_moment / (np.sqrt(second_moment) + 1e-7))
```

Momentum

AdaGrad / RMSProp

Sort of like RMSProp with momentum

Q: What happens at first timestep?

Kingma and Ba, "Adam: A method for stochastic optimization", ICLR 2015

Adam (full form)

```
first_moment = 0
second_moment = 0
for t in range(num_iterations):
  dx = compute_gradient(x)
  first_moment = beta1 * first_moment  + (1 - beta1) * dx
  second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
  first_unbias = first_moment / (1 - beta1 ** t)
  second_unbias = second_moment / (1 - beta2 ** t)
  x -= learning_rate * first_unbias / (np.sqrt(second_unbias) + 1e-7))
```

Momentum

Bias correction

AdaGrad / RMSProp

Bias correction for the fact that first and second moment estimates start at zero

Kingma and Ba, “Adam: A method for stochastic optimization”, ICLR 2015

Adam (full form)

```
first_moment = 0
second_moment = 0
for t in range(1, num_iterations):
  dx = compute_gradient(x)
  first_moment = beta1 * first_moment  + (1 - beta1) * dx
  second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
  first_unbias = first_moment / (1 - beta1 ** t)
  second_unbias = second_moment / (1 - beta2 ** t)
  x -= learning_rate * first_unbias / (np.sqrt(second_unbias) + 1e-7))
```

Momentum

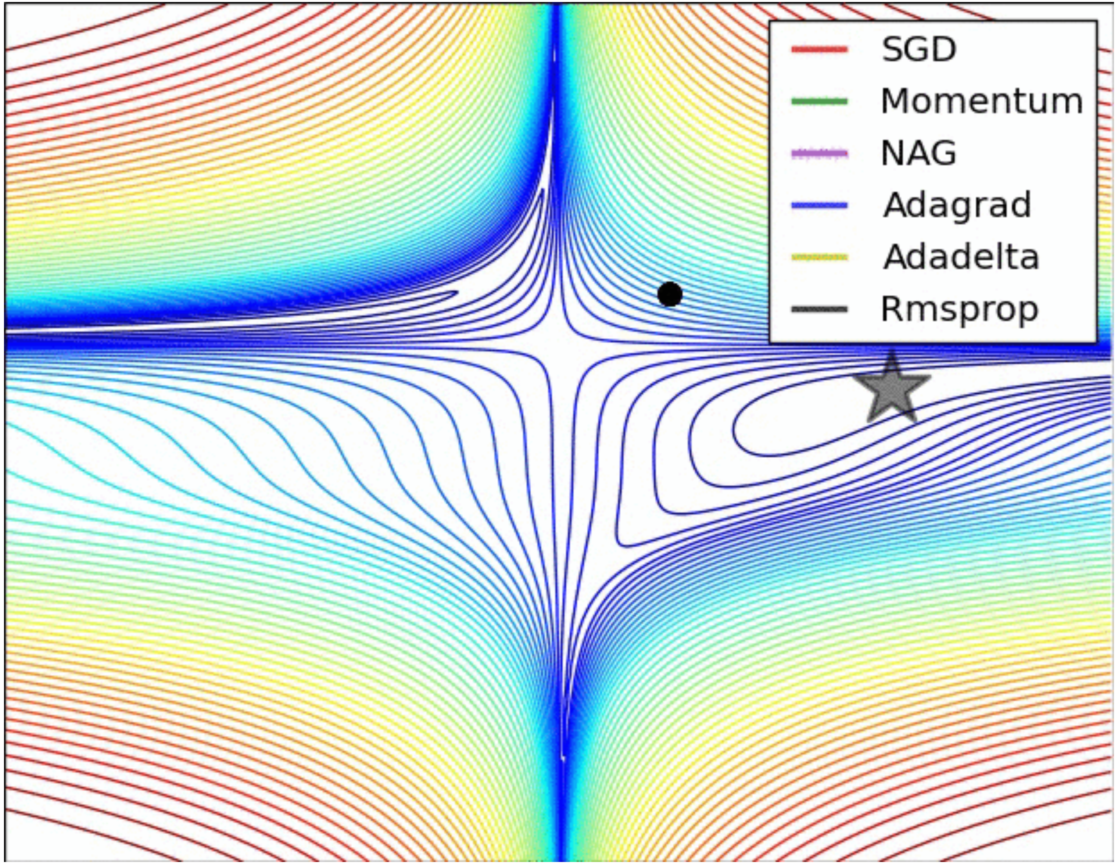
Bias correction

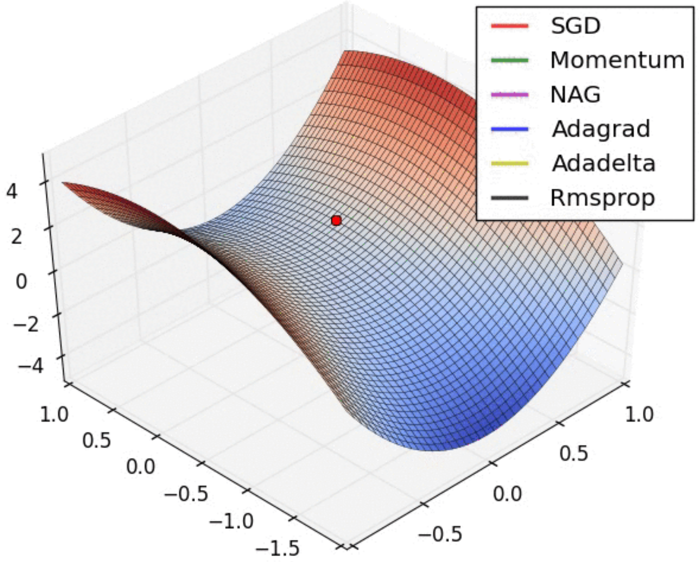
AdaGrad / RMSProp

Bias correction for the fact that first and second moment estimates start at zero

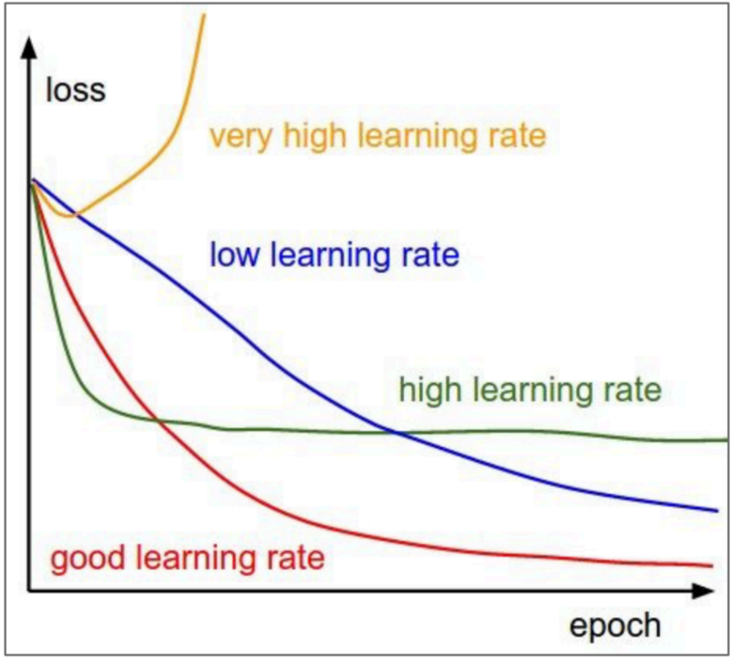
Adam with beta1 = 0.9, beta2 = 0.999, and learning_rate = 1e-3 or 5e-4 is a great starting point for many models!

Kingma and Ba, “Adam: A method for stochastic optimization”, ICLR 2015


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

SGD, SGD+Momentum, Adagrad, RMSProp, Adam all have **learning rate** as a hyperparameter.



=> Learning rate decay over time!

step decay:
e.g. decay learning rate by half every few epochs.

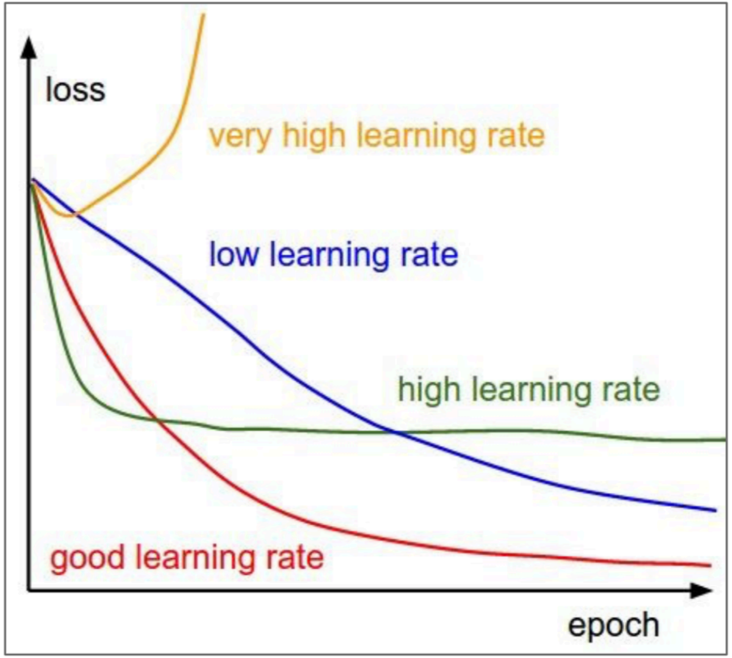
exponential decay:

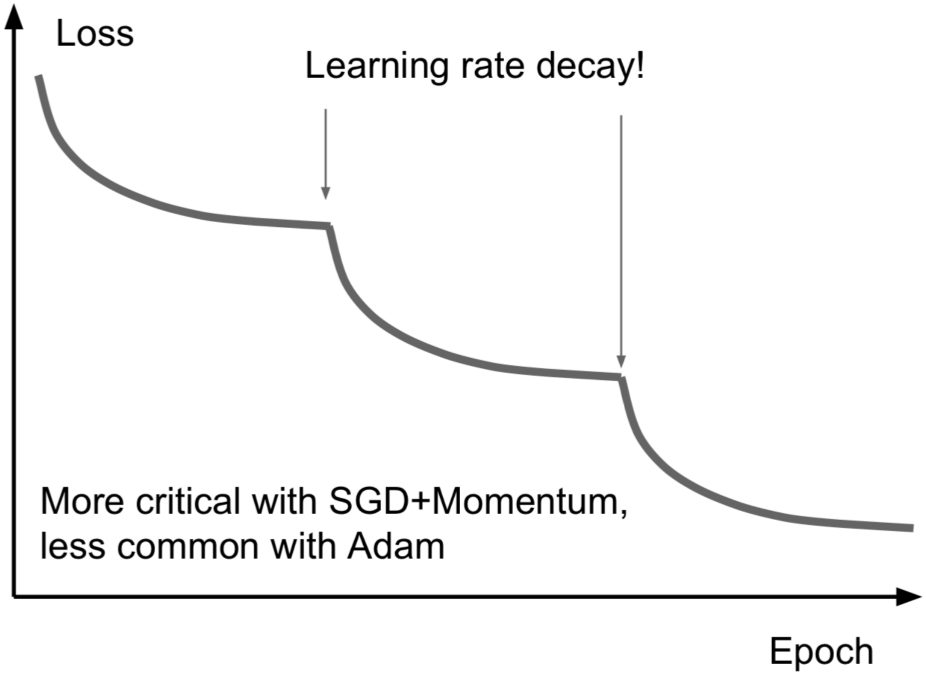
$$\alpha = \alpha_0 e^{-kt}$$

1/t decay:

$$\alpha = \alpha_0 / (1 + kt)$$

SGD, SGD+Momentum, Adagrad, RMSProp, Adam all have **learning rate** as a hyperparameter.





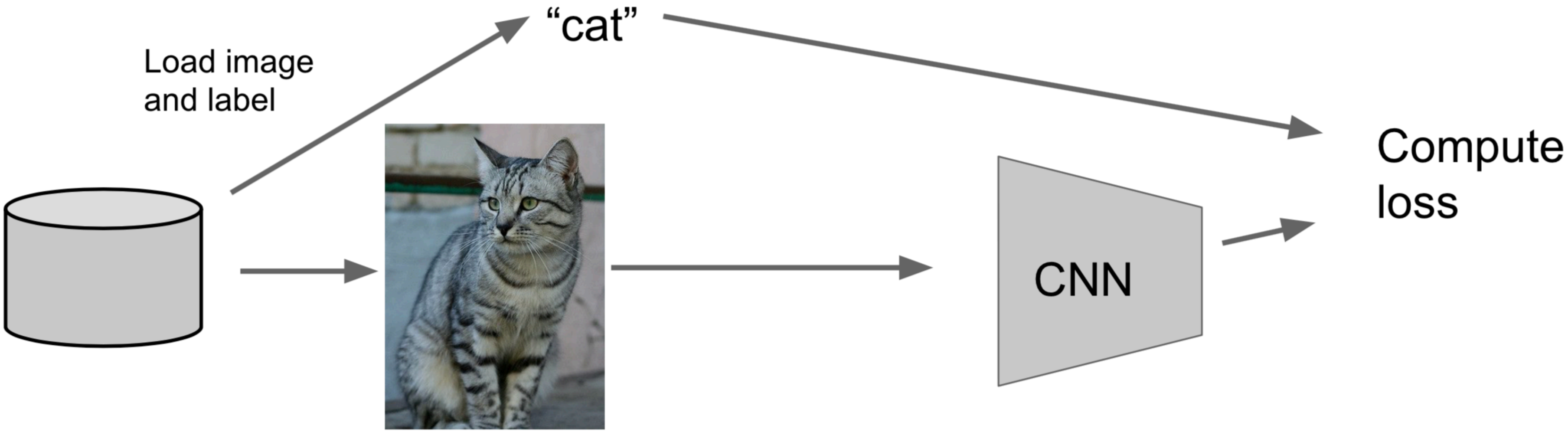
지난 시간엔 ...

- Regularization
  - Weight Decay
  - Batch normalization
  - Dropout

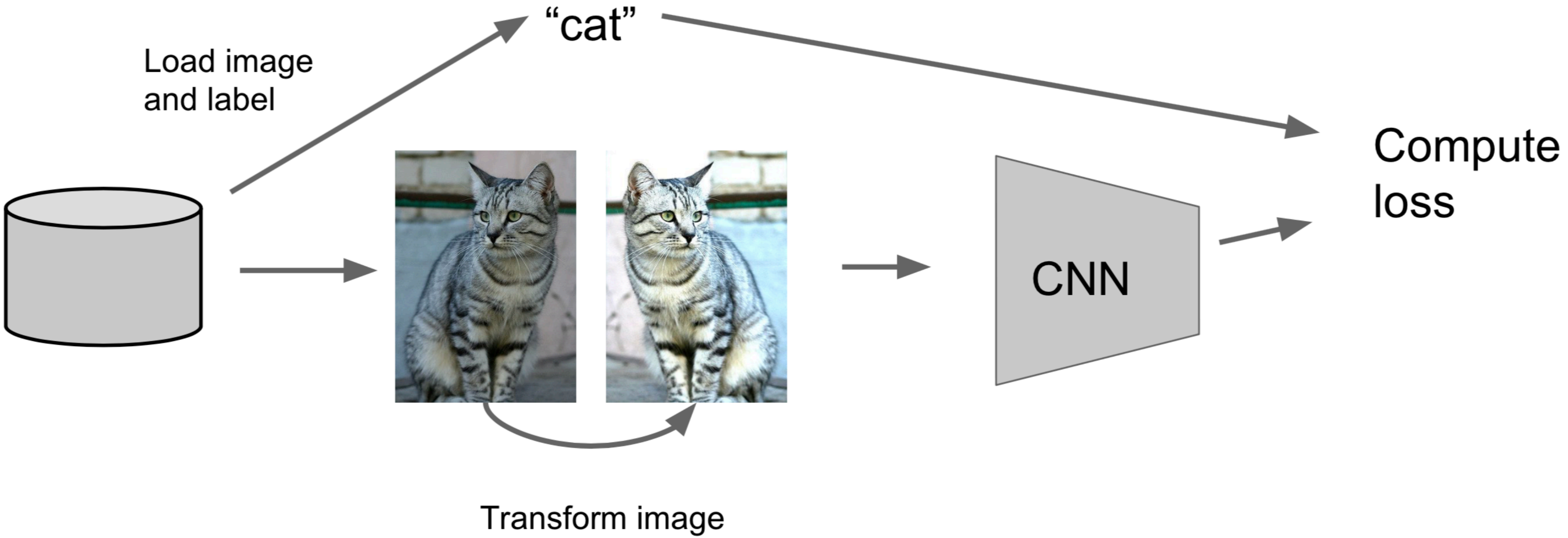
지난 시간엔 ...

- Regularization
  - Weight Decay
  - Batch normalization
  - Dropout
  - Data augmentation

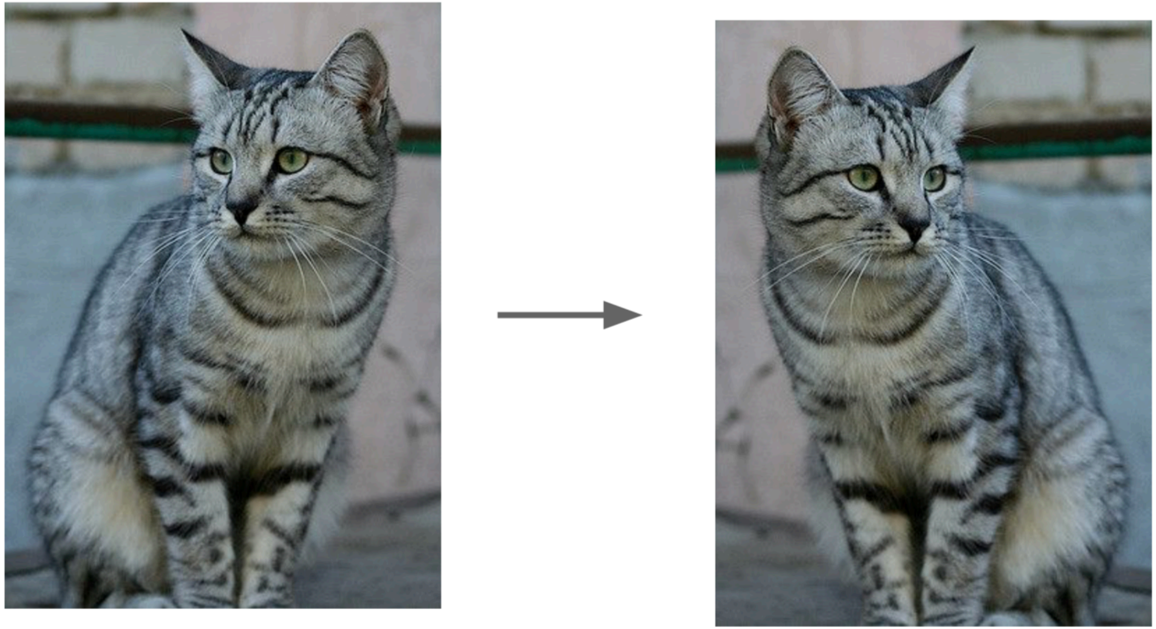
Regularization: Data Augmentation



Regularization: Data Augmentation



Data Augmentation
Horizontal Flips

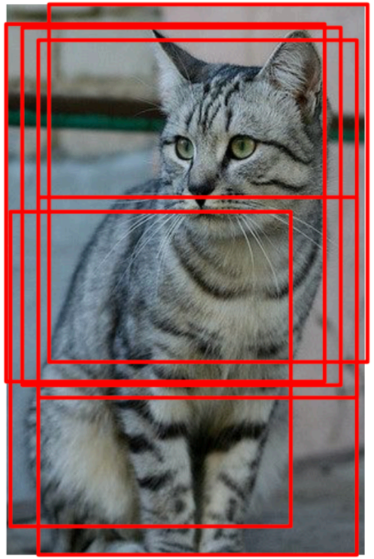


Data Augmentation
Random crops and scales

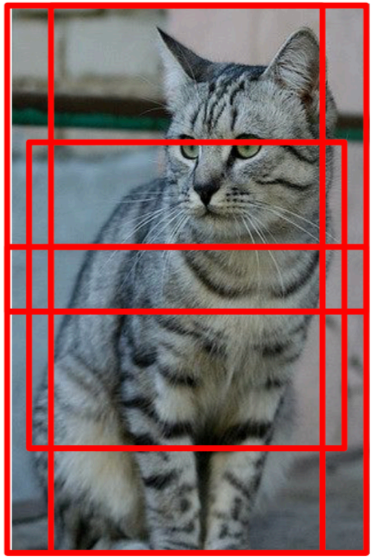
Training: sample random crops / scales

ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch



Data Augmentation
Random crops and scales



Training: sample random crops / scales

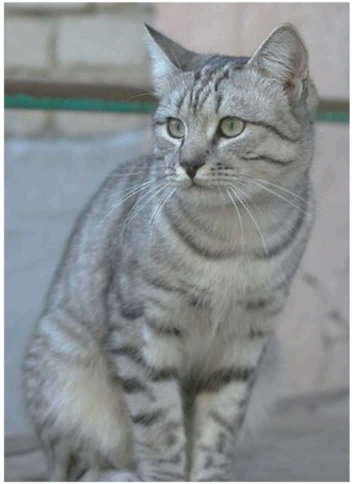
ResNet:
1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch

Testing: average a fixed set of crops

ResNet:
1. Resize image at 5 scales: {224, 256, 384, 480, 640}
2. For each size, use 10 224 x 224 crops: 4 corners + center, + flips

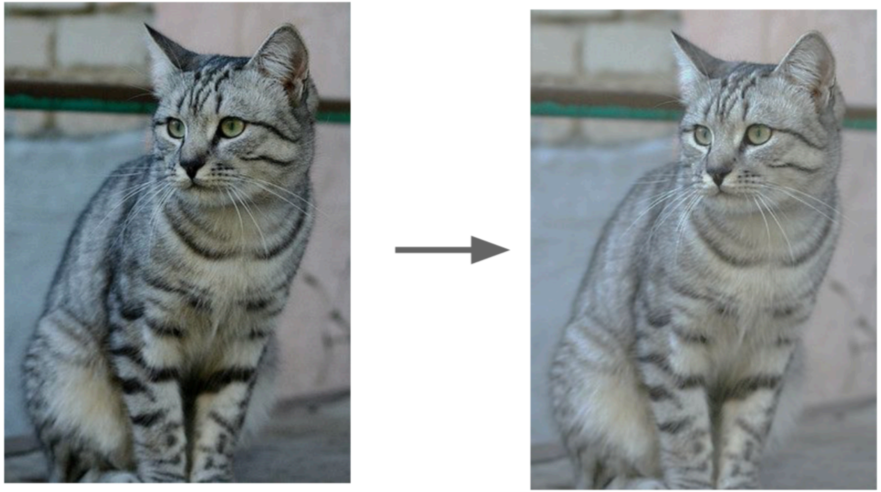
Data Augmentation
Color Jitter

Simple: Randomize contrast and brightness



Data Augmentation
Color Jitter

Simple: Randomize contrast and brightness



More Complex:

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a "color offset" along principal component directions
3. Add offset to all pixels of a training image

(As seen in *[Krizhevsky et al. 2012]*, ResNet, etc)

Data Augmentation

Get creative for your problem!

Random mix/combinations of :

- translation
- rotation
- stretching
- shearing,
- lens distortions, … (go crazy)

Regularization: A common pattern

Training: Add random noise

Testing: Marginalize over the noise

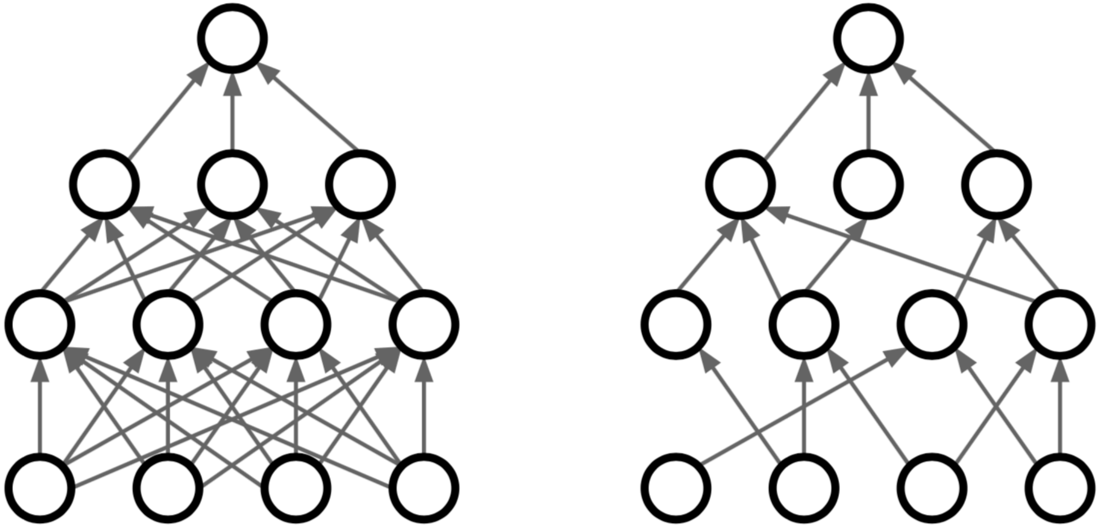
Examples:

Dropout

Batch Normalization

Data Augmentation

DropConnect

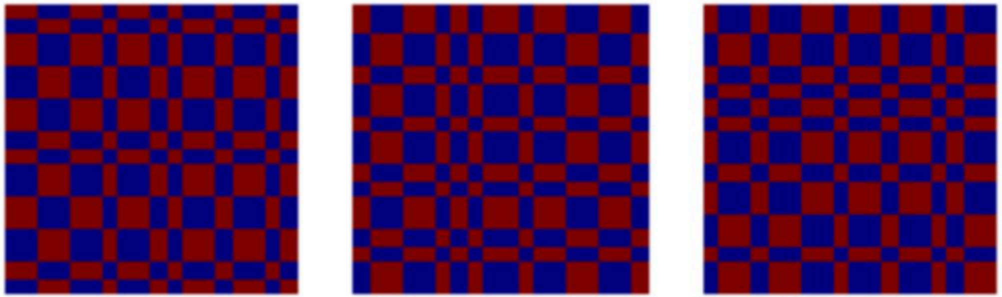


Wan et al, “Regularization of Neural Networks using DropConnect”, ICML 2013

Regularization: A common pattern

Training: Add random noise
Testing: Marginalize over the noise

Examples:
Dropout
Batch Normalization
Data Augmentation
DropConnect
Fractional Max Pooling

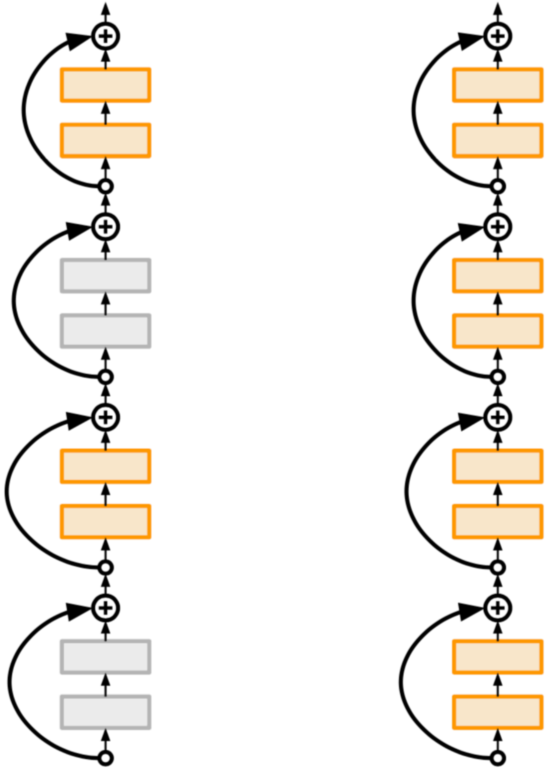


Graham, “Fractional Max Pooling”, arXiv 2014

Regularization: A common pattern

Training: Add random noise
Testing: Marginalize over the noise

Examples:
Dropout
Batch Normalization
Data Augmentation
DropConnect
Fractional Max Pooling
Stochastic Depth



Huang et al, “Deep Networks with Stochastic Depth”, ECCV 2016

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

BUSTED

Transfer Learning with CNNs

Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

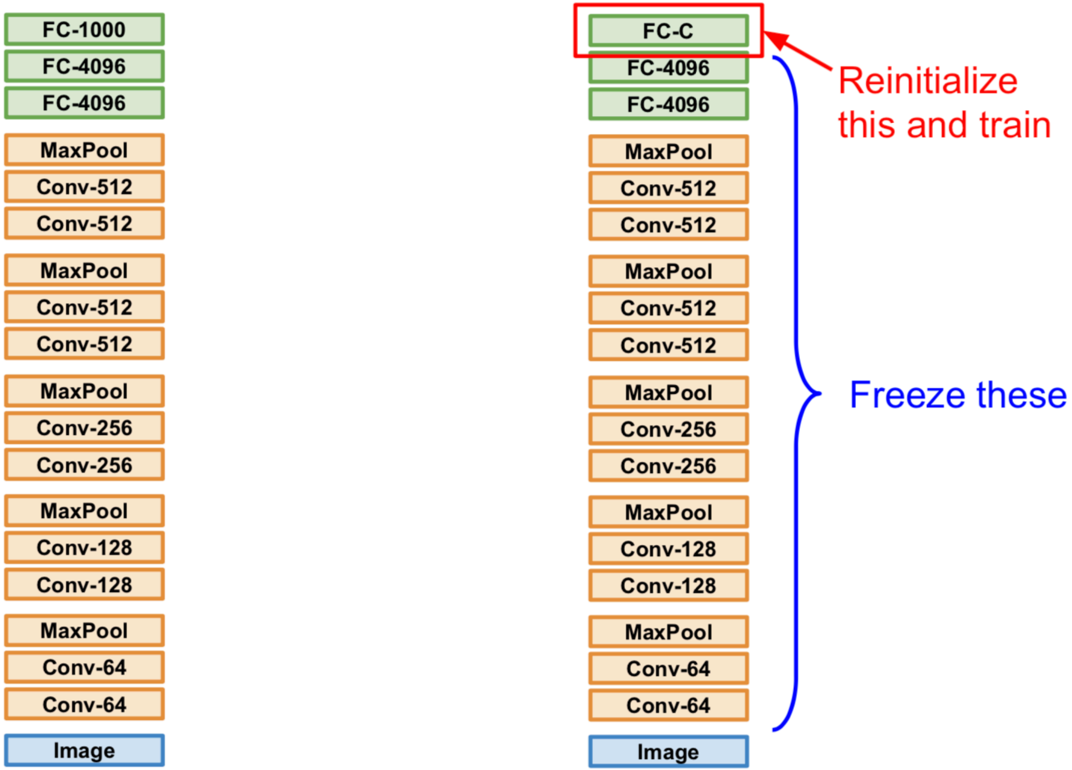
1. Train on Imagenet

Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

1. Train on Imagenet

2. Small Dataset (C classes)



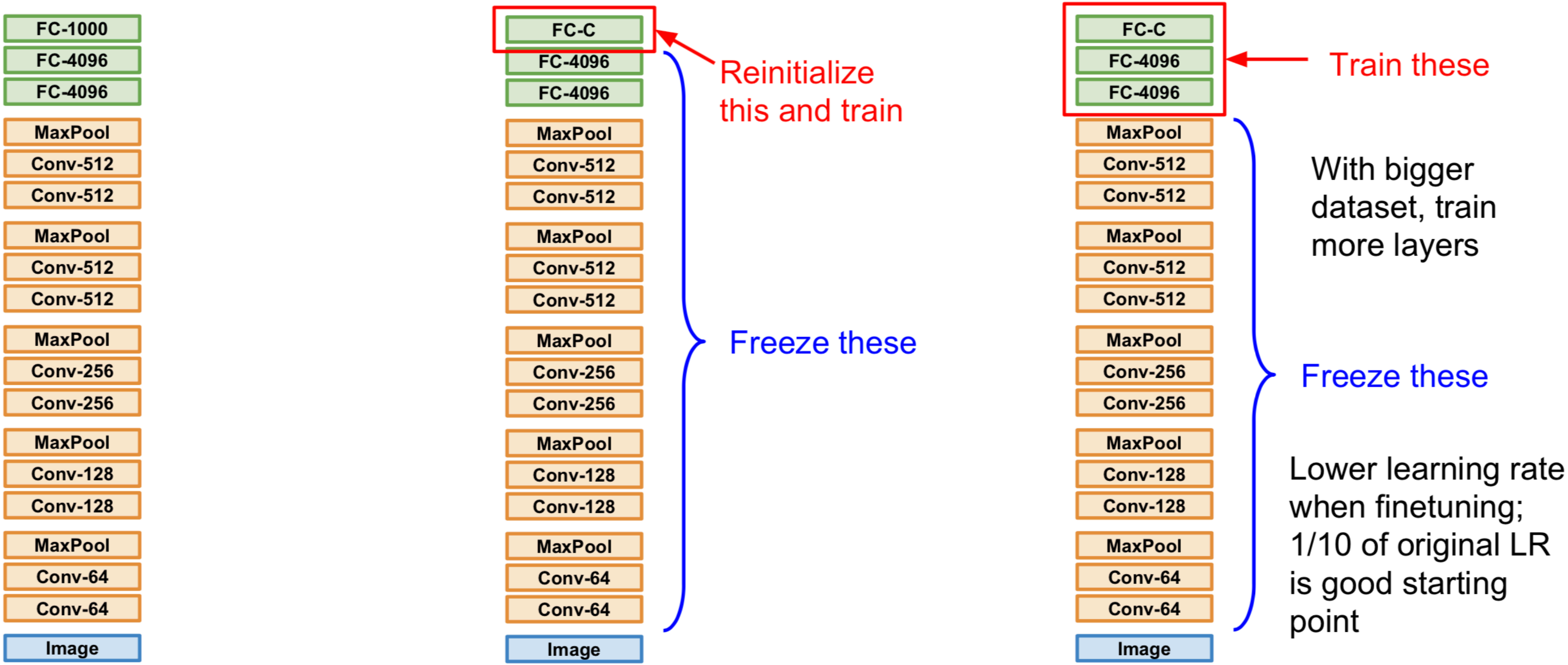
Transfer Learning with CNNs

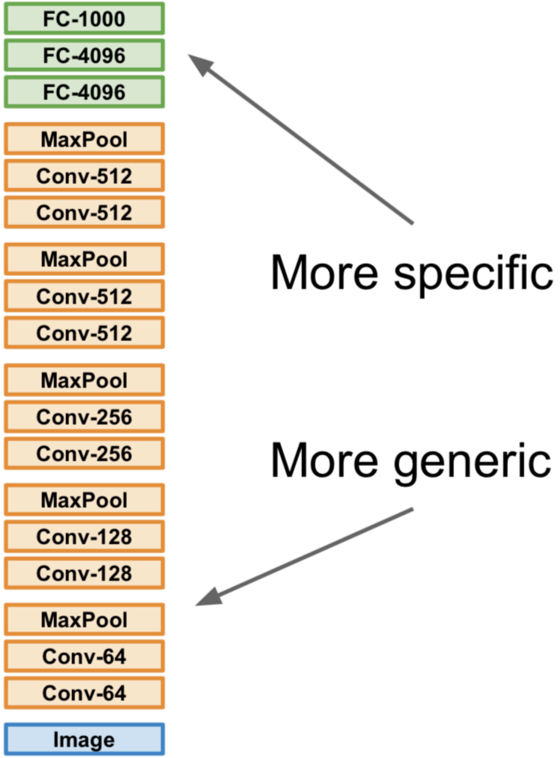
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

1. Train on Imagenet

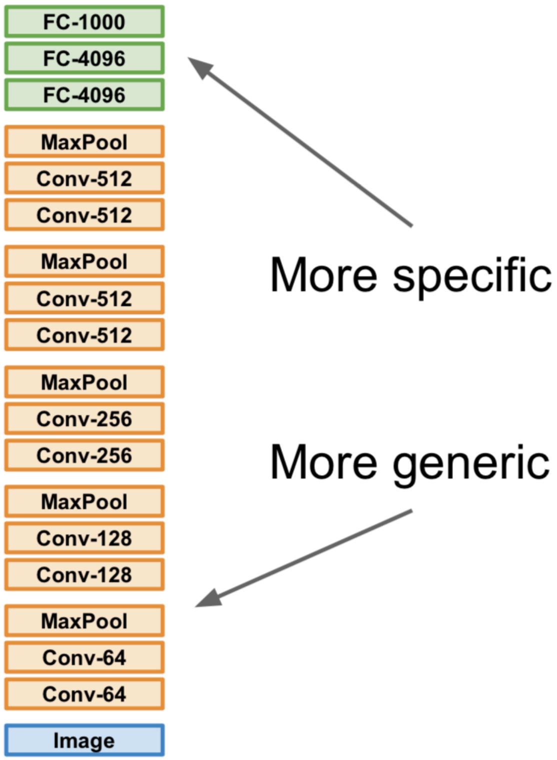
2. Small Dataset (C classes)

3. Bigger dataset

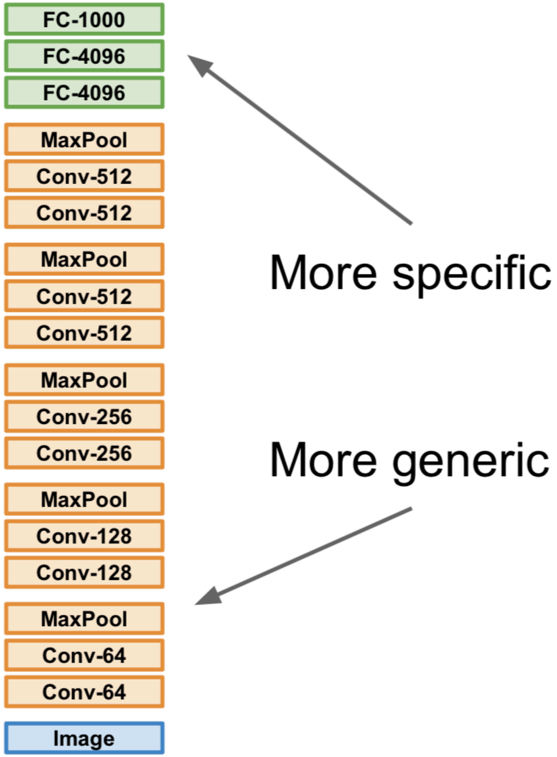




	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)

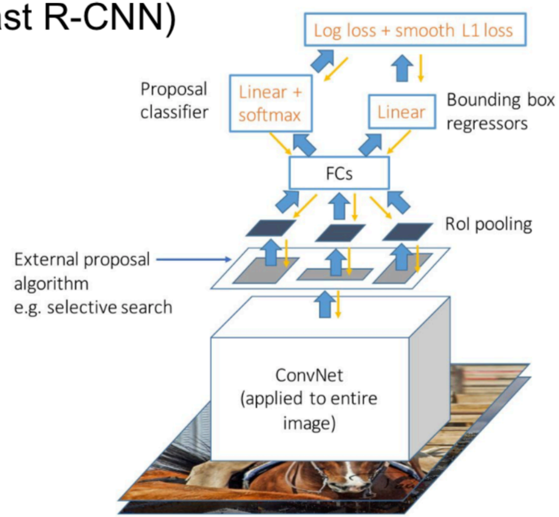
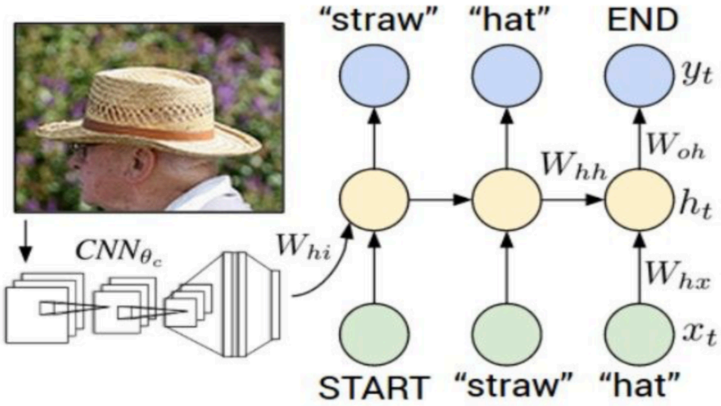


Image Captioning: CNN + RNN

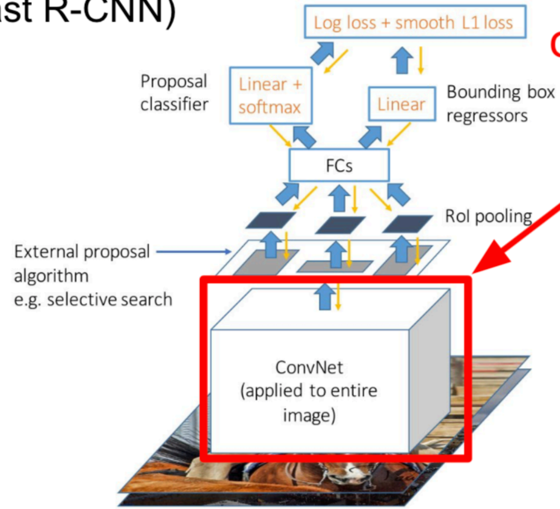


Girshick, "Fast R-CNN", ICCV 2015

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

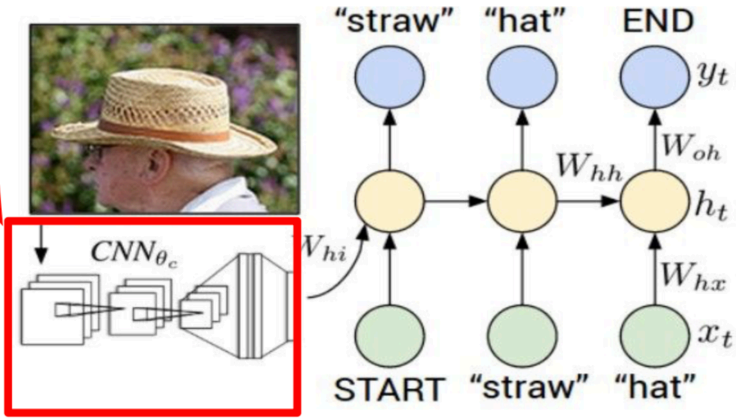
Transfer learning with CNNs is pervasive…
(it’s the norm, not an exception)

Object Detection
(Fast R-CNN)



CNN pretrained on ImageNet

Image Captioning: CNN + RNN

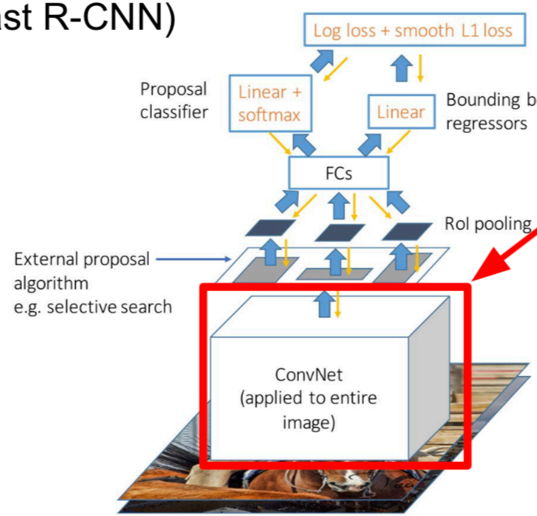


Girshick, “Fast R-CNN”, ICCV 2015

Karpathy and Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions”, CVPR 2015

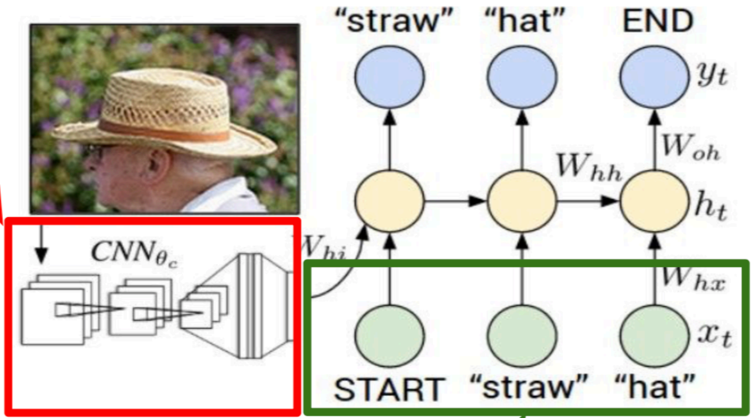
Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)



CNN pretrained on ImageNet

Image Captioning: CNN + RNN



Word vectors pretrained with word2vec

Girshick, "Fast R-CNN", ICCV 2015

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

Summary

- Optimization
  - Momentum, RMSProp, Adam, etc
- Regularization
  - Dropout, etc
- Transfer learning
  - Use this for your projects!

다음 시간

- RNN-LSTM
- Second order Optimization (조금만)

모두의연구소

박 은 수 Research Director

E-mail : es.park@modulabs.co.kr