Administrative

- **Poster Session** on Wednesday, worth 3% of final grade, +2% for top few posters. There will be food
- **CS224D** (Deep Learning for NLP) was announced for next quarter taught by Richard Socher, natural followup for more DL.

CS224D Syllabus and Schedule

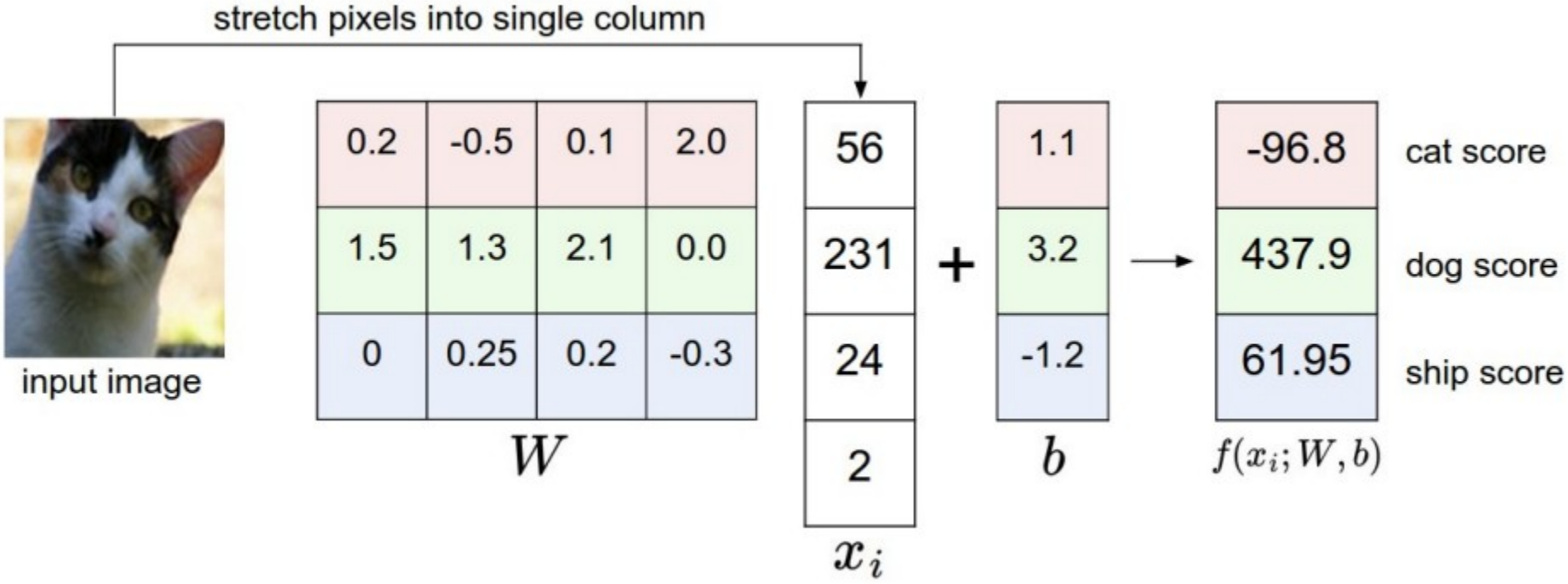
Event Type	Date	Description
Lecture	Week 1	Intro to NLP
Lecture	Week 1	Simple Word Vector representations: word2vec, GloVe
Lecture	Week 2	Optimization (SGD, mini-batches), Visualization (PCA, t-sne)
Lecture	Week 2	Advanced word vector representations: language models, softmax, clustering (k-means)
Lecture	Week 3	Neural Networks and backpropagation
Lecture	Week 3	Practical tips: gradient checks, overfitting, regularization, activation functions, details
Lecture	Week 4	Recurrent neural networks
Lecture	Week 4	GRUs and LSTMs
Lecture	Week 5	Recursive neural networks
Lecture	Week 6	Convolutional neural networks
Lecture	Week 6	Novel Memory Models
Lecture	Week 7	Additional applications not covered as motivating examples yet
Lecture	Week 7	Efficient implementations and GPUs
Lecture	Week 8	Invited Speaker: TBD
Lecture	Week 8	Future applications and open problems

Tiny ImageNet Spotlights

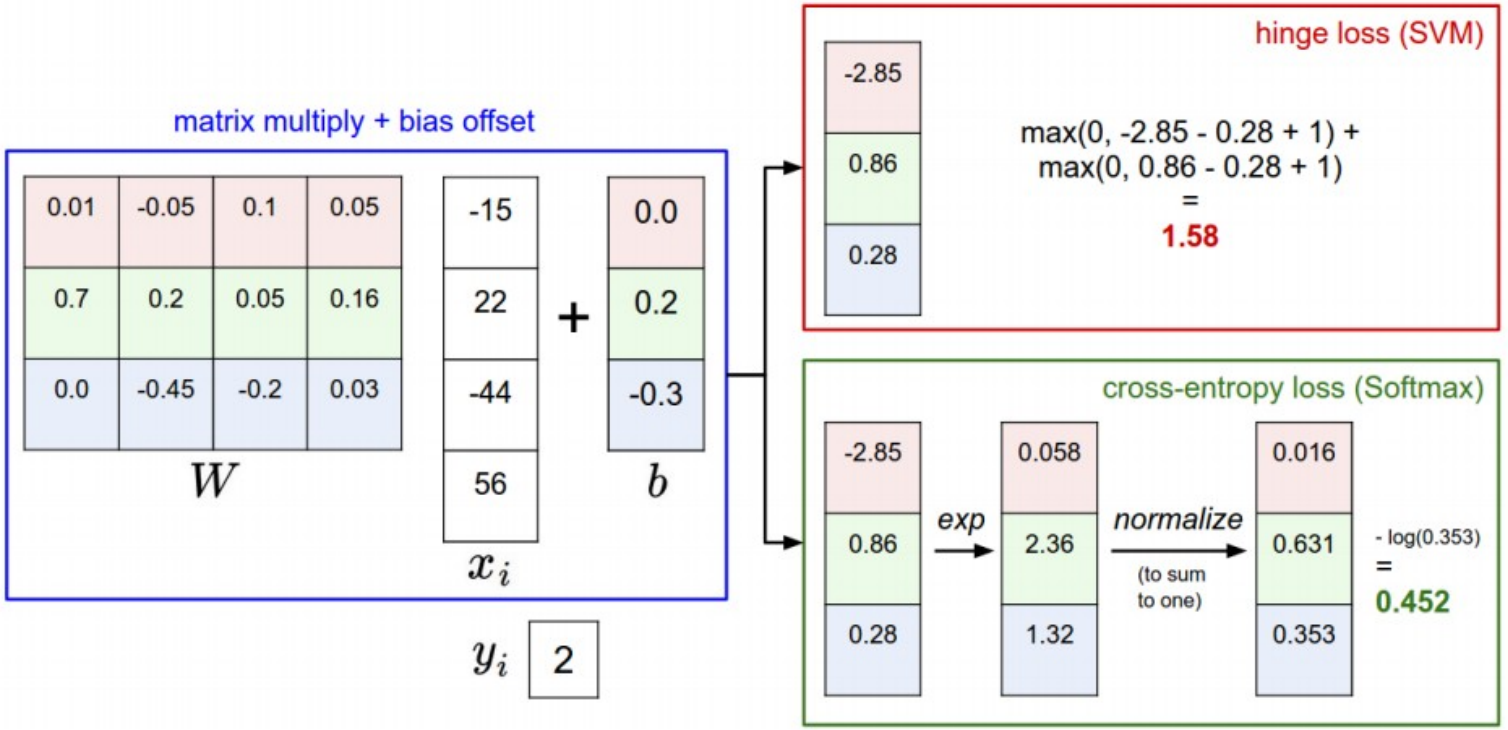
Leaderboard

#	Name	Error Rate	# Submissions
1	Dasgupta, Saumitro	0.438	1
2	Feng, Shaoming	0.505	3
3	Hansen, Lucas	0.562	6
4	Yao, Leon	0.573	2
5	Miller, John	0.580	9
6	Wang, Keven	0.610	5
7	Yang, Xuan	0.645	3
8	Boin, Jean-Baptiste	0.696	3
9	Pour Ansari, Mohammad Hadi	0.759	8
10	Ghili, Saman	0.773	8
11	Bodington, Dash	0.953	6
12	Banerjee, Arijit	0.982	1
13	Random Guesser	0.995	2

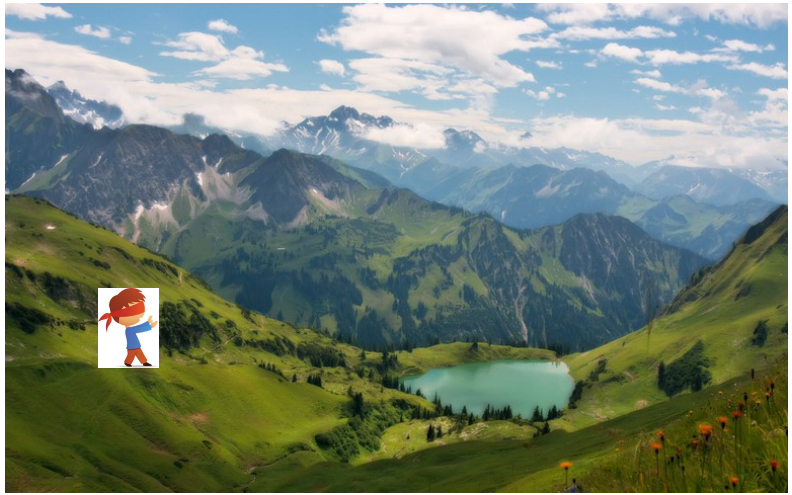
Together, we've defined Score Functions...



And Loss Functions...

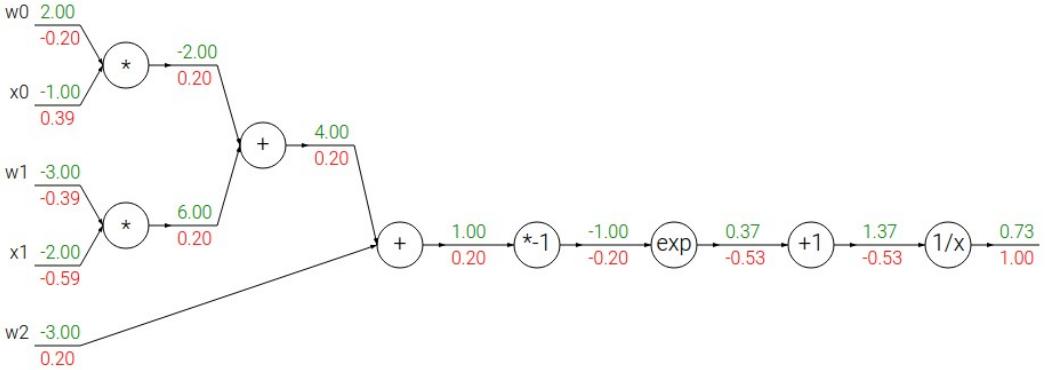


We've learned how to optimize them...

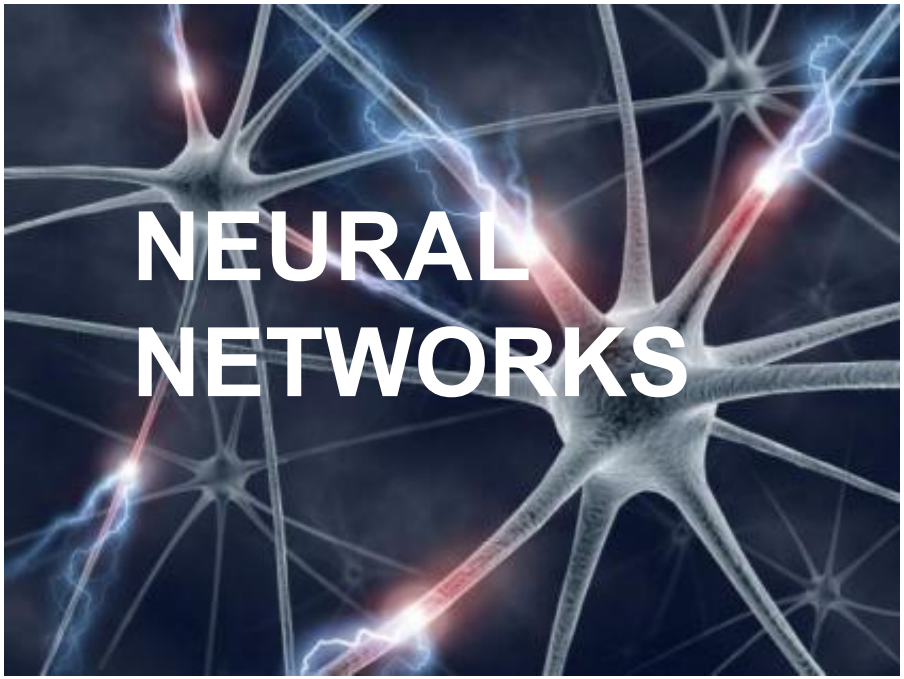


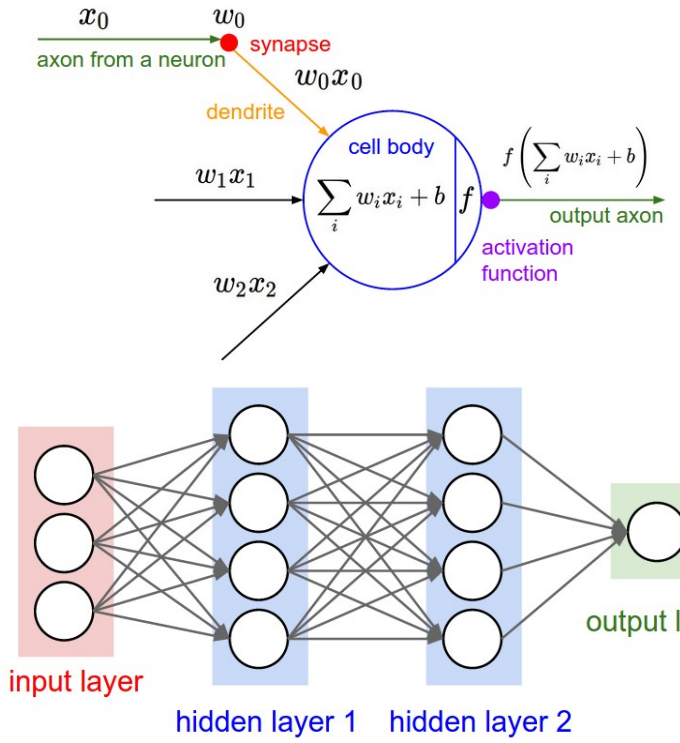
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

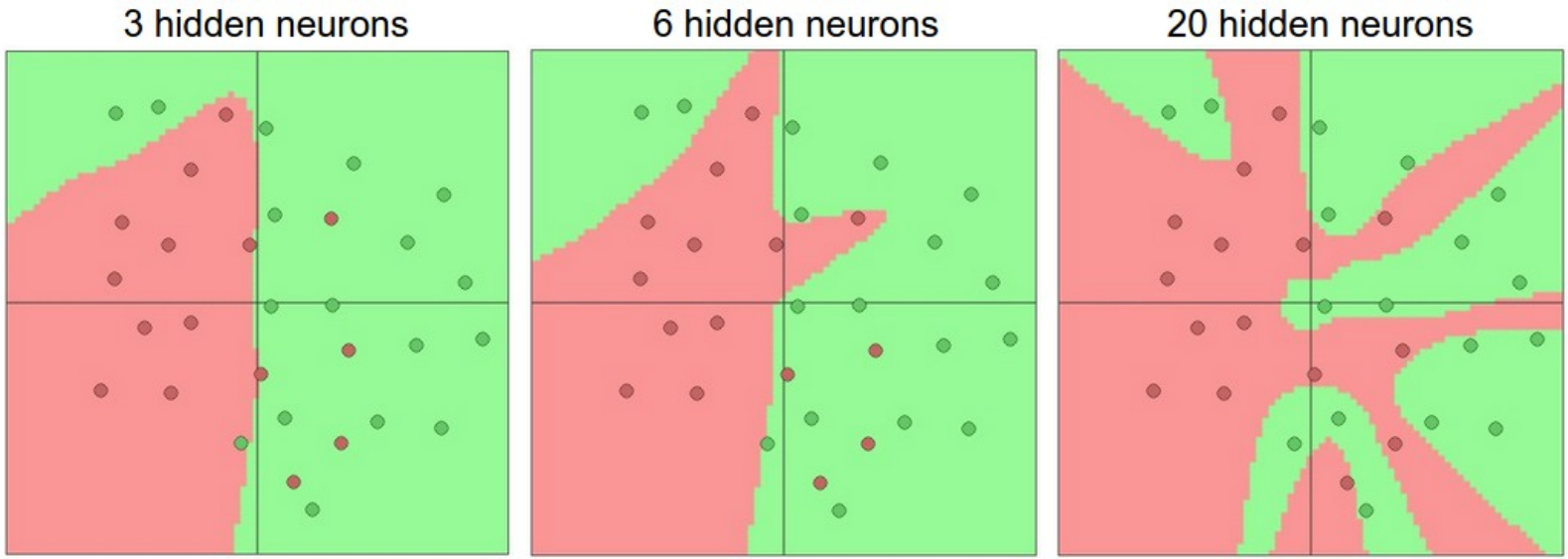


We learned to express more powerful Score Functions...

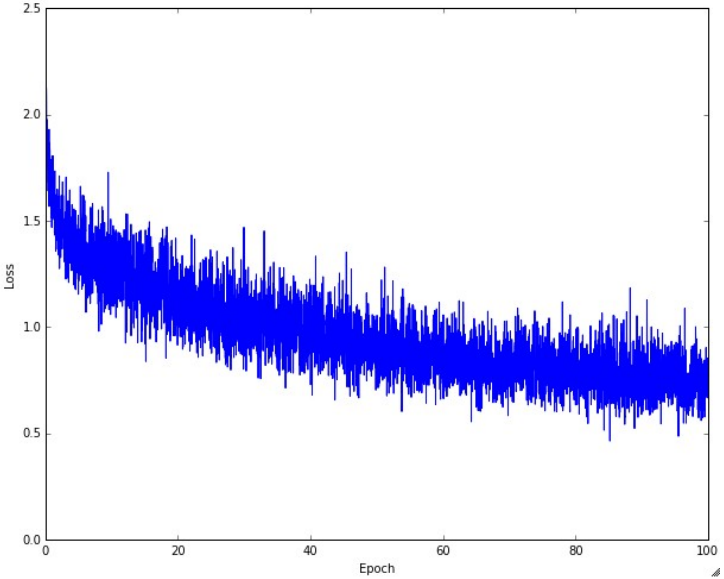


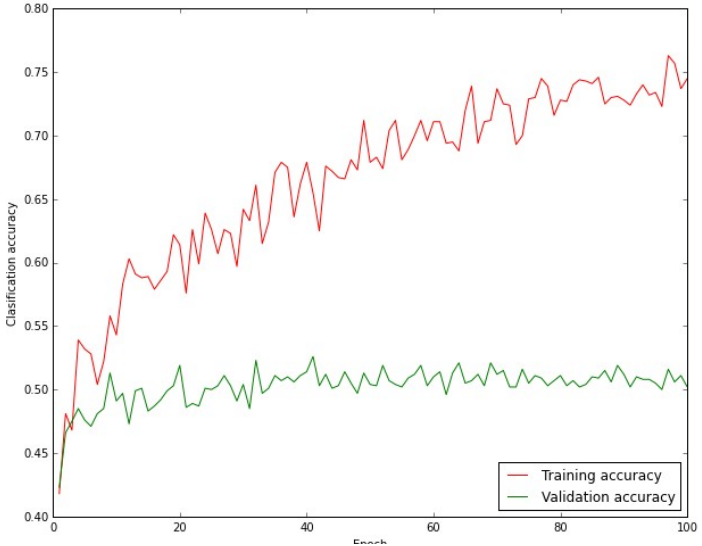


For an extra wiggle...

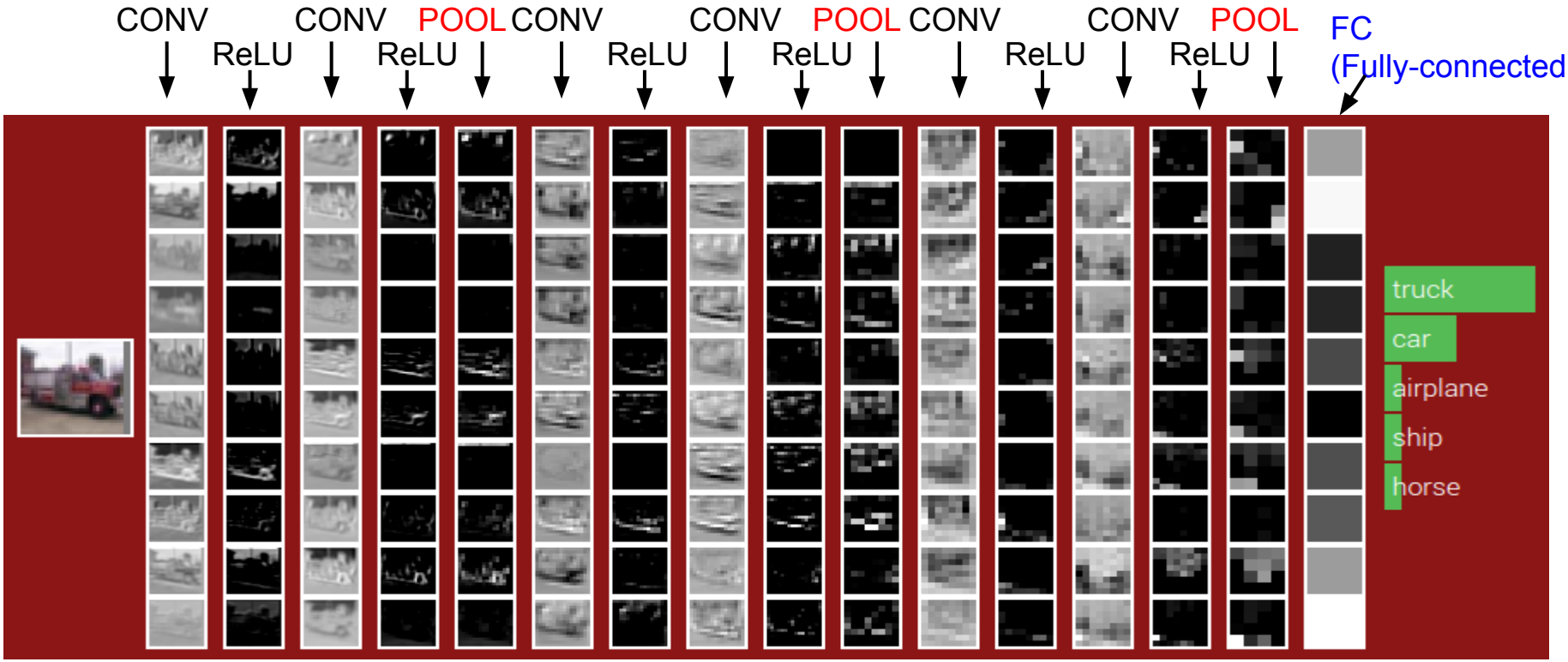


Together we tamed the learning process...

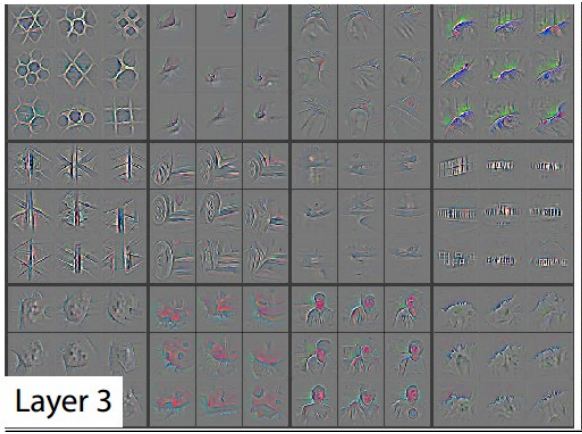


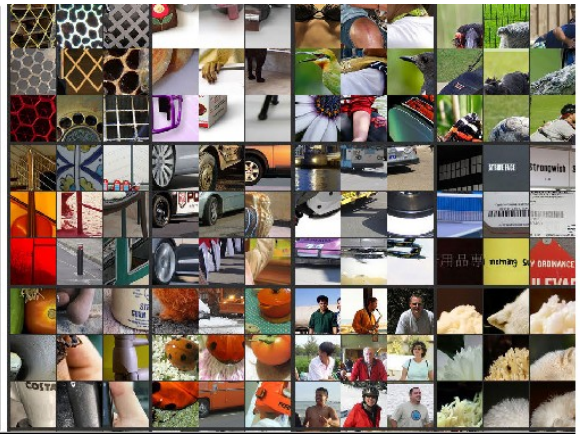


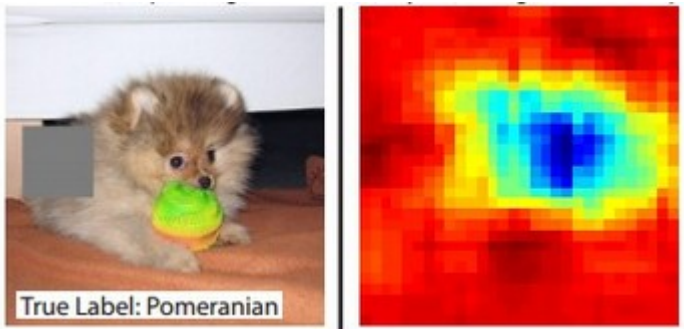
Together we explored image-specific Neural Nets...



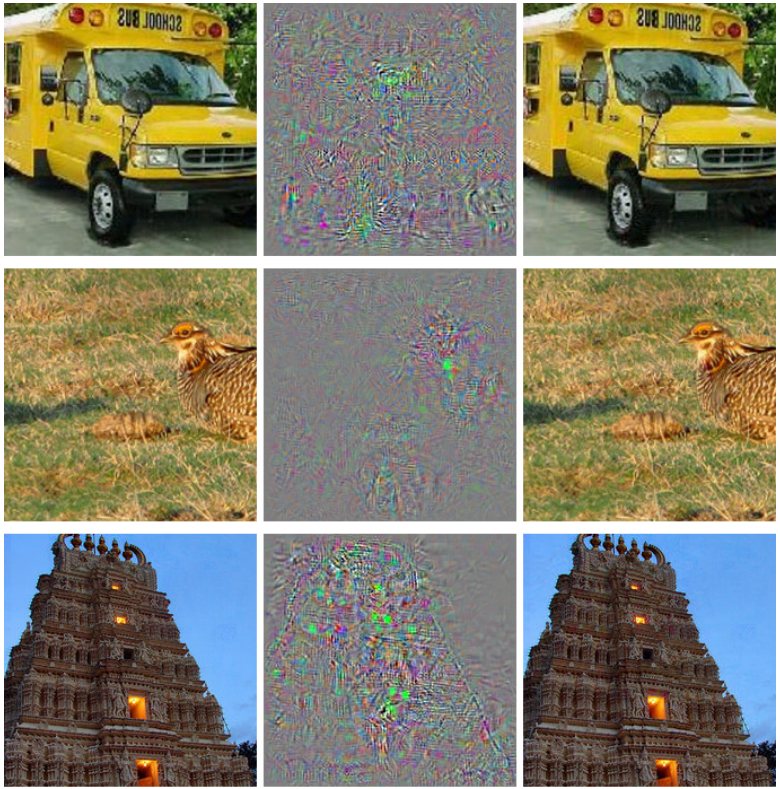
We explored how they work...

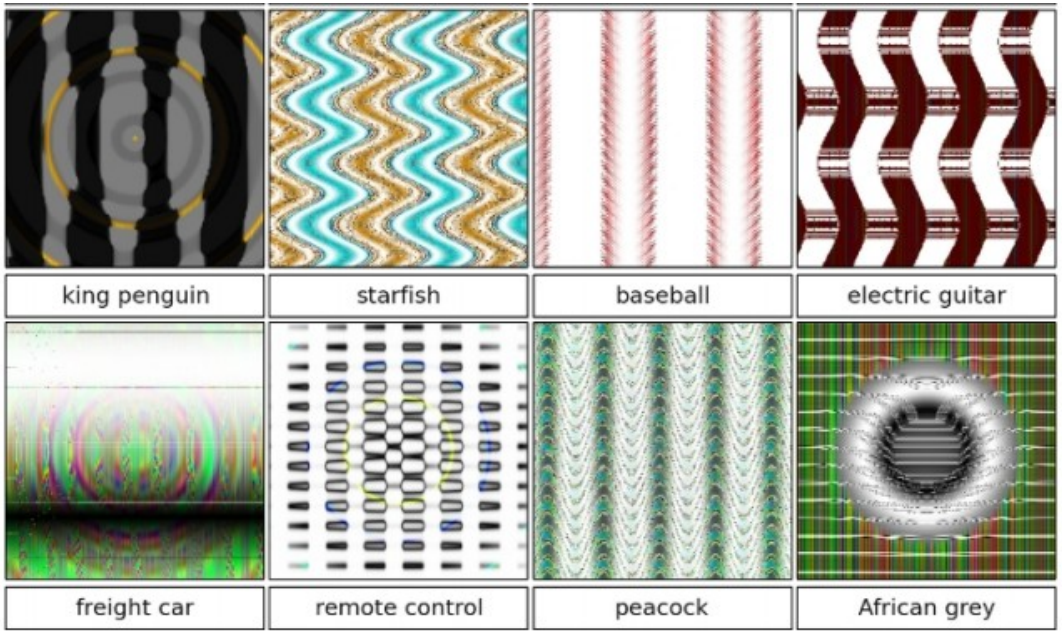






And how they don’t… (but really they still do)





We looked at what makes ConvNets “tick”...

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

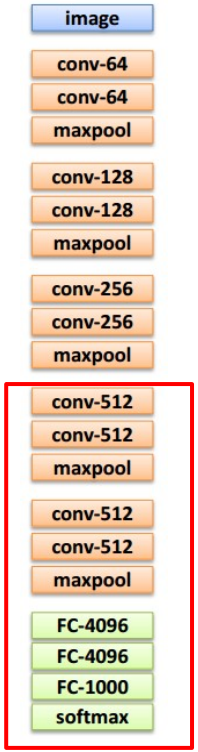
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

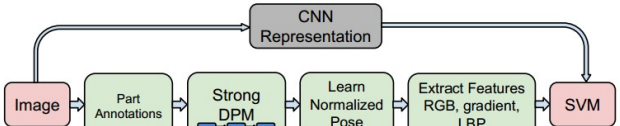
Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1

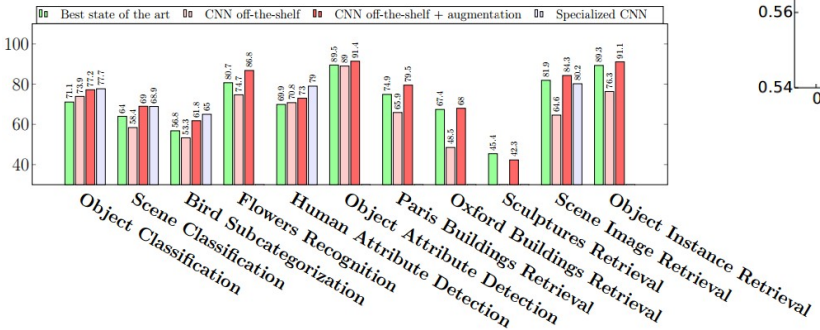
Our Model (as per Fig. 3)	33.1	38.4	16.5
Adjust layers 6,7: 2048 units	38.2	40.2	17.6
Adjust layers 6,7: 8192 units	22.0	38.8	17.0
Adjust layers 3,4,5: 512,1024,512 maps	18.8	**37.5**	**16.0**
Adjust layers 6,7: 8192 units and Layers 3,4,5: 512,1024,512 maps	**10.0**	38.3	16.9

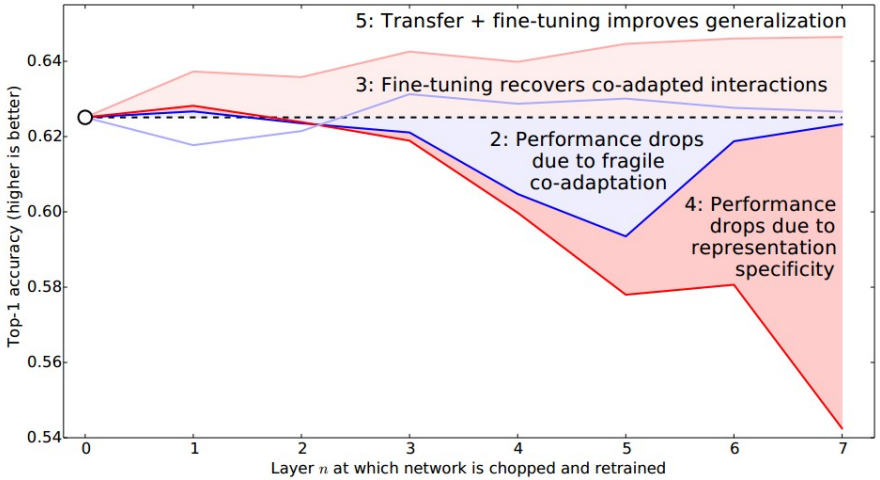
(p) CNN M	–	(C)	f	s	4K	**79.89**
(x) CNN M 2048	–	(C)	f	s	2K	**80.10**
(y) CNN M 1024	–	(C)	f	s	1K	**79.91**
(z) CNN M 128	–	(C)	f	s	128	**78.60**

And studied their mysterious generalization powers…

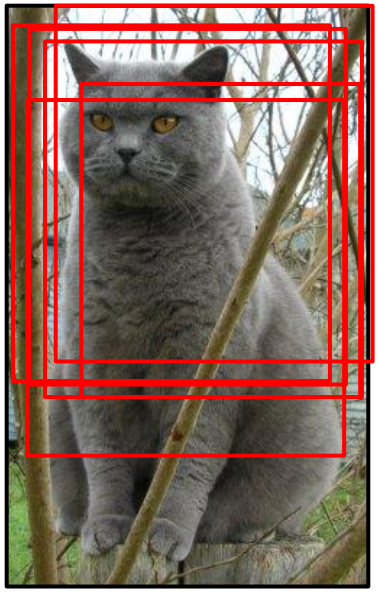






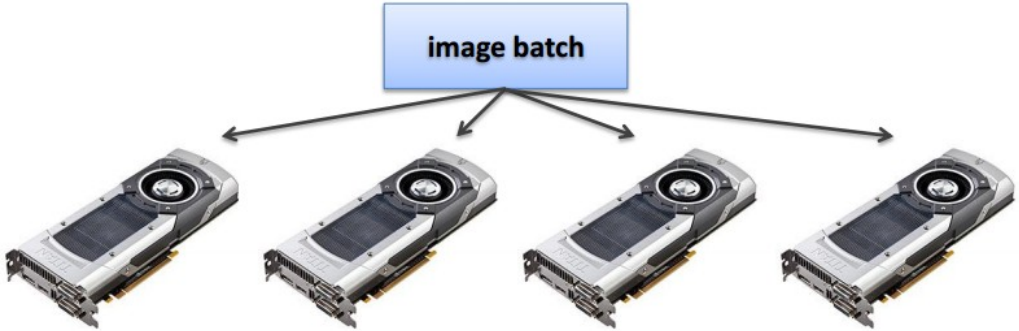


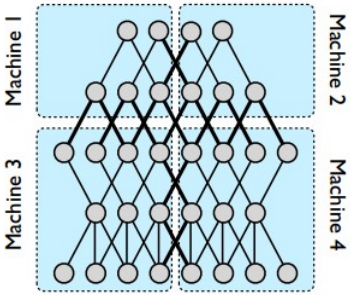
We learned tips/tricks for making ConvNets work well in practice

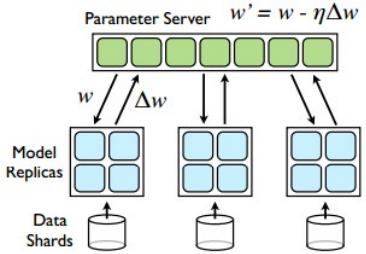


And explored their practical bottlenecks...

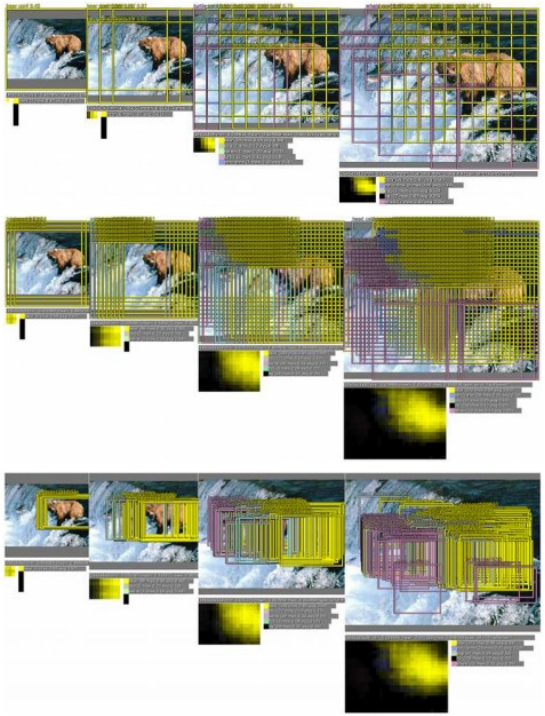
Moving parts lol

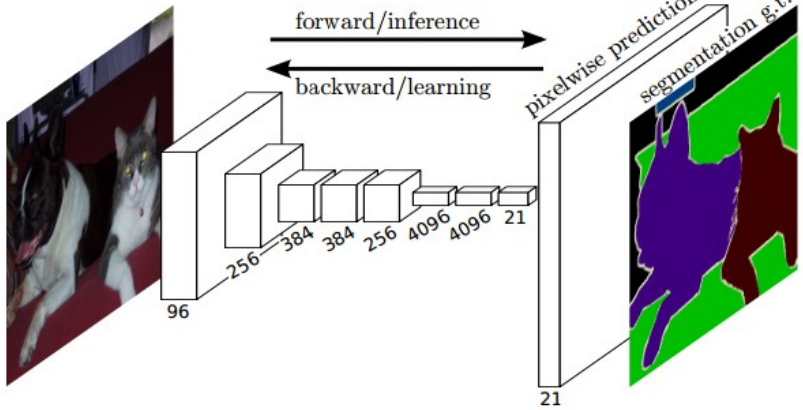


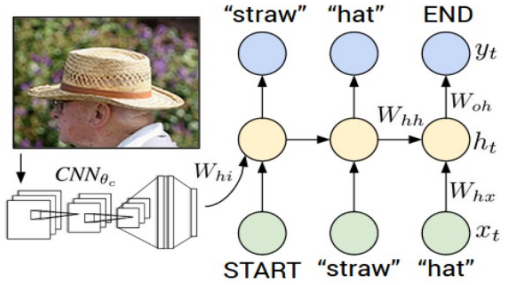




And we bravely ventured beyond Image Classification...



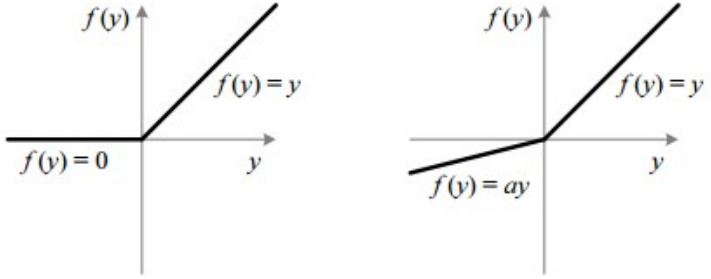




And developed an understanding of cutting-edge research

We saw many **2015** citations...
e.g.

Delving Deep into Rectifiers:
Surpassing Human-Level Performance on ImageNet Classification
[Kaiming He et al., 2015] (MSR)



4.94% error
Top 5 ImageNet error

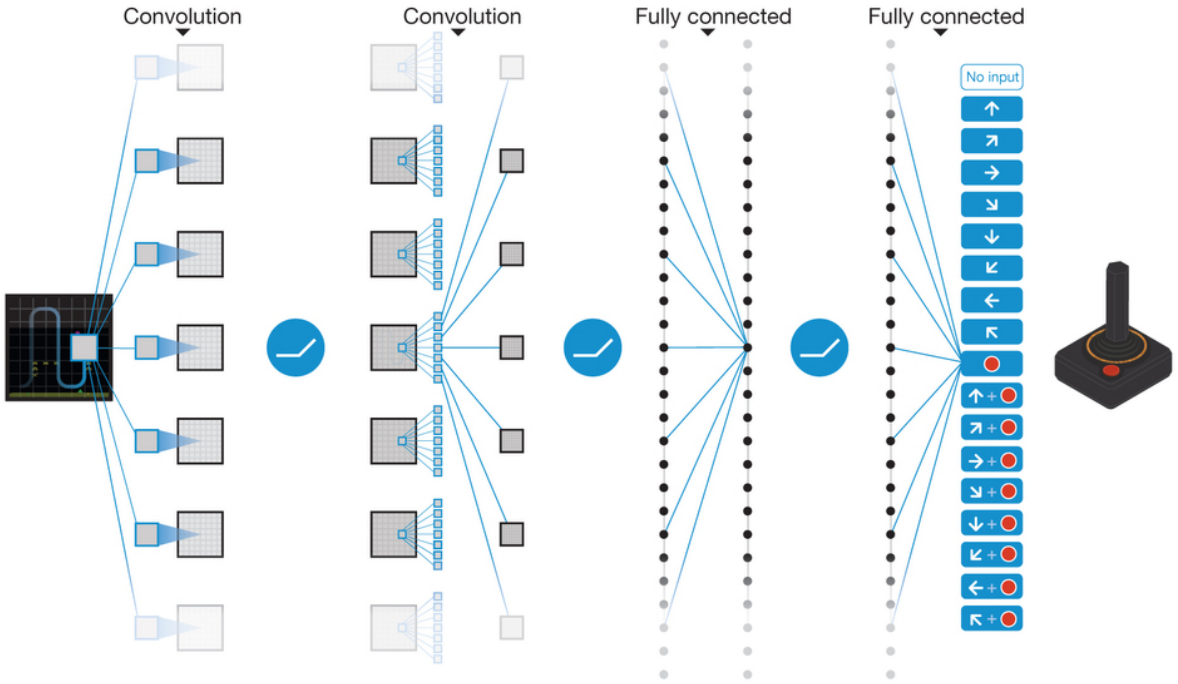
You are now ready.

You are now ready.

You are now ready.

Hints of beyond

Reinforcement Learning meets Computer Vision

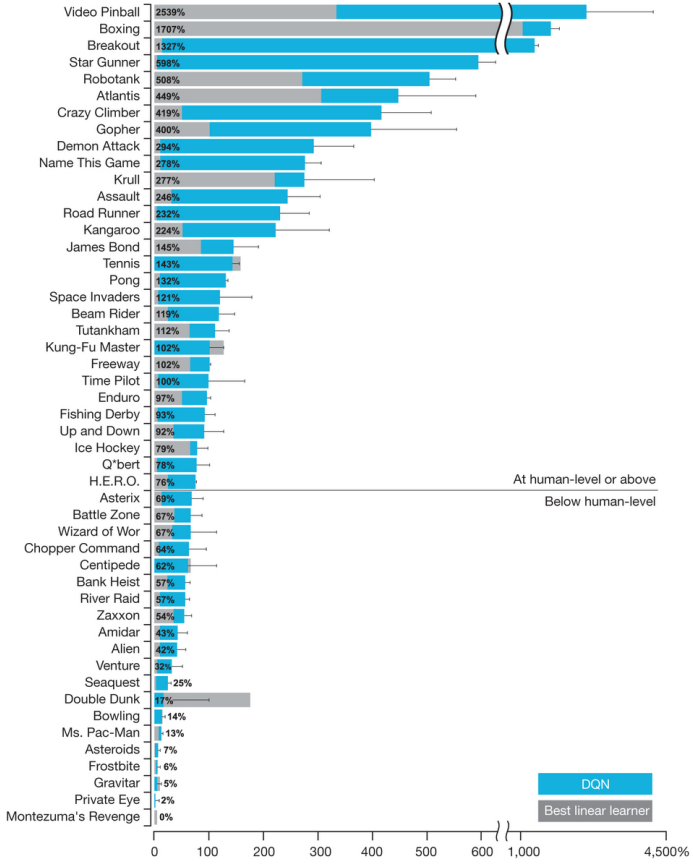


Human-level control through deep reinforcement learning
[Mnih et al.], **Nature** 2015
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

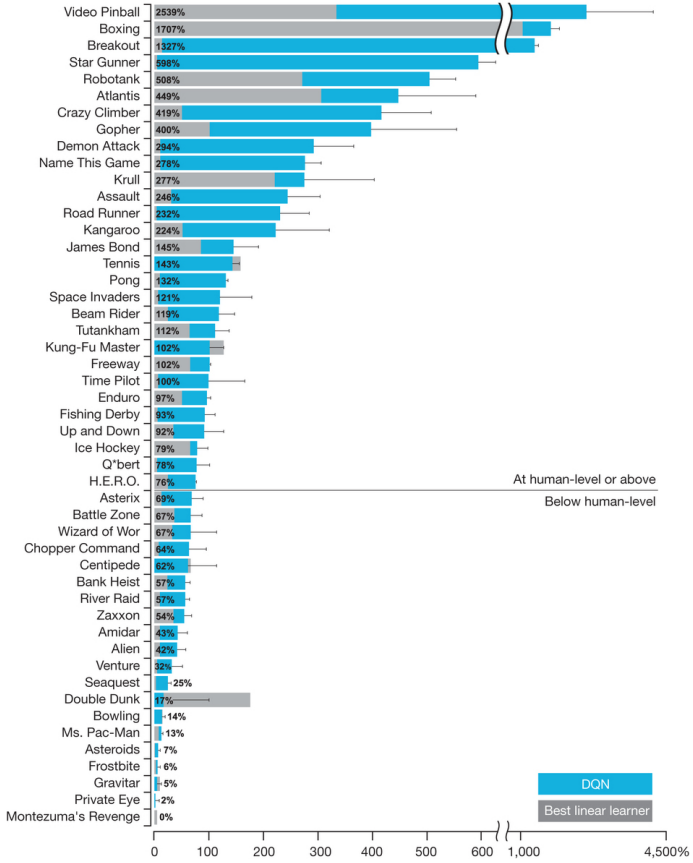
(play videos)

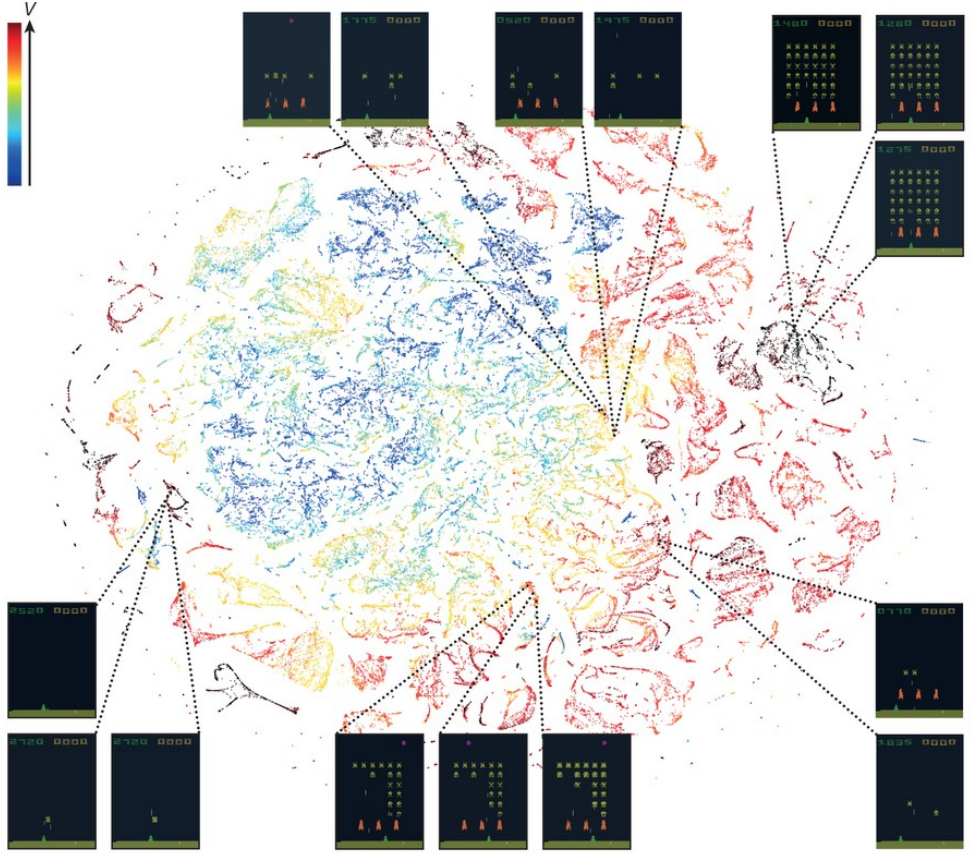
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

Reinforcement Learning meets Computer Vision

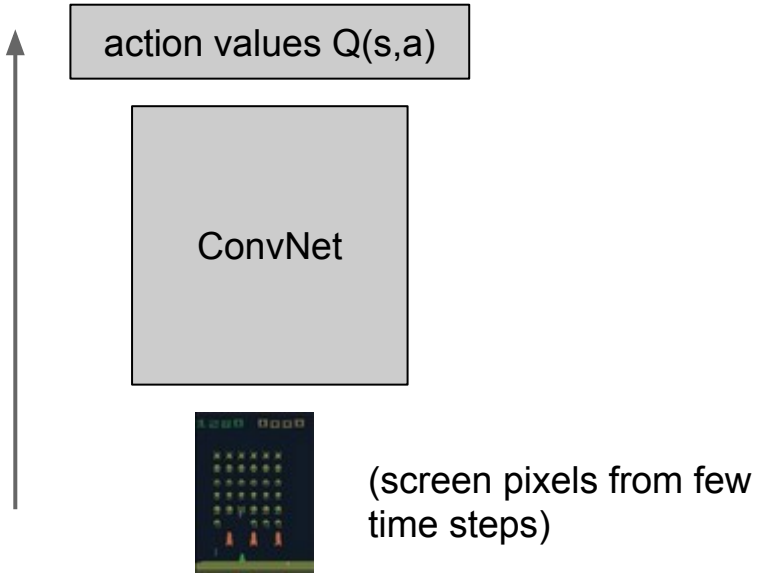


Reinforcement Learning meets Computer Vision



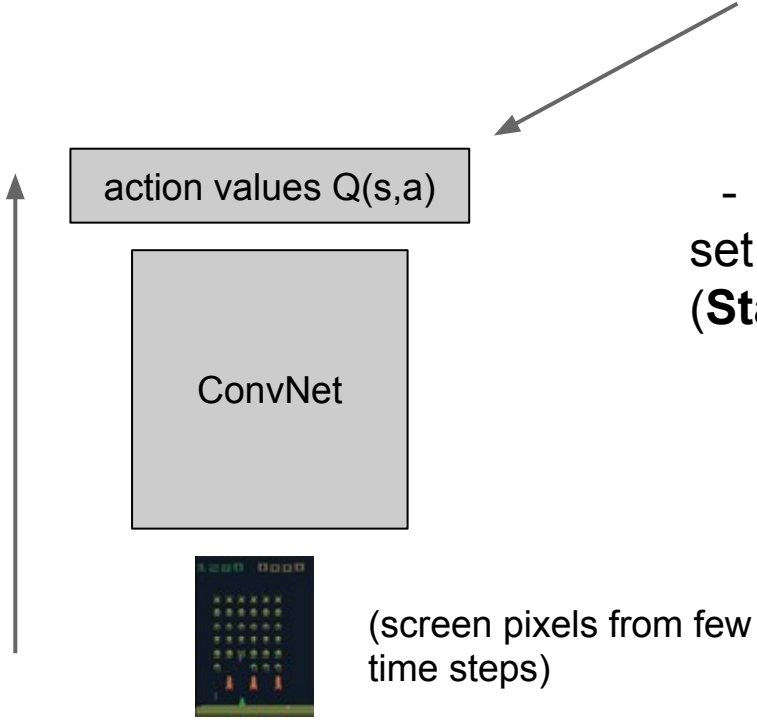


(Approximate idea of the model)



(Approximate idea of the model)

- Assume finite number of actions
- Each number here is a real-valued quantity that represents the **“Q function”** in RL



- Collect *experience* dataset:

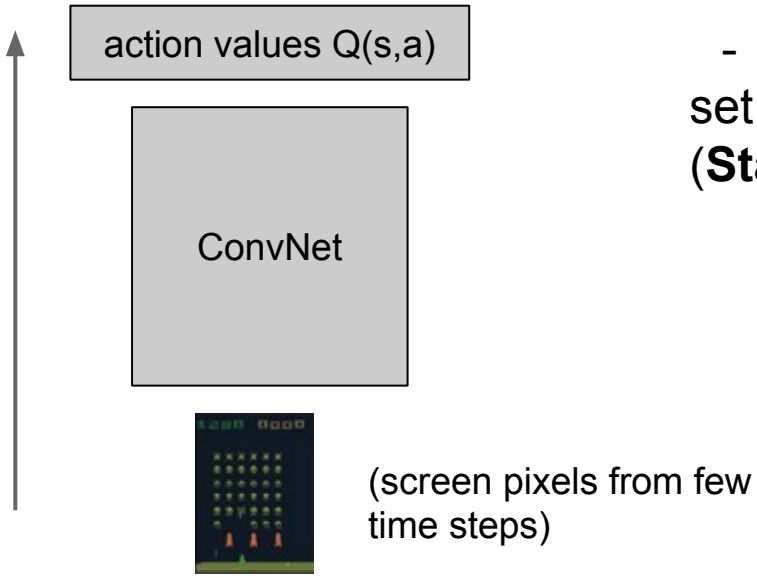
set of tuples {(s,a,s’,r), … }

(**State**, **Action taken, New state**, **Reward received**)

(Approximate idea of the model)

- Assume finite number of actions
- Each number here is a real-valued quantity

that represents the **“Q function”** in RL



- Collect *experience* dataset:

set of tuples {(s,a,s’,r), … }

(**State**, **Action taken, New state**, **Reward received**)

L2 Regression loss:

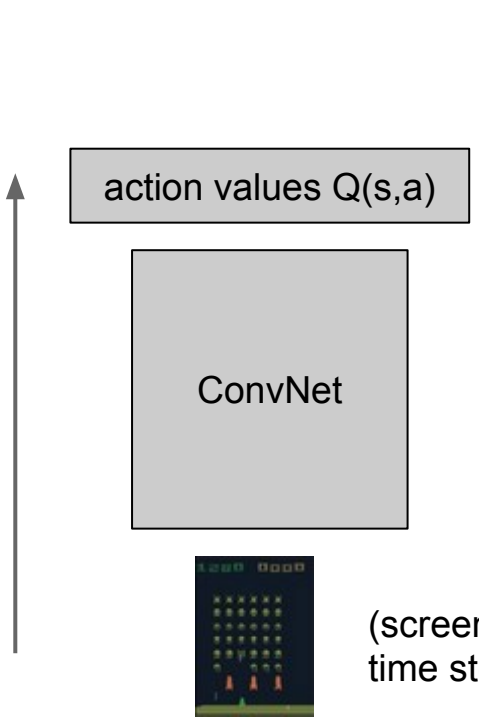
$$\left[\left(r+\gamma \max_{a'} Q(s',a';\theta_i^-)\right)-Q(s,a;\theta_i)\right]^2$$

target value predicted value

(Approximate idea of the model)

- Assume finite number of actions
- Each number here is a real-valued quantity

that represents the **“Q function”**



- Collect *experience* dataset:

set of tuples {(s,a,s’,r), … }

(**State**, **Action taken, New state**, **Reward received**)

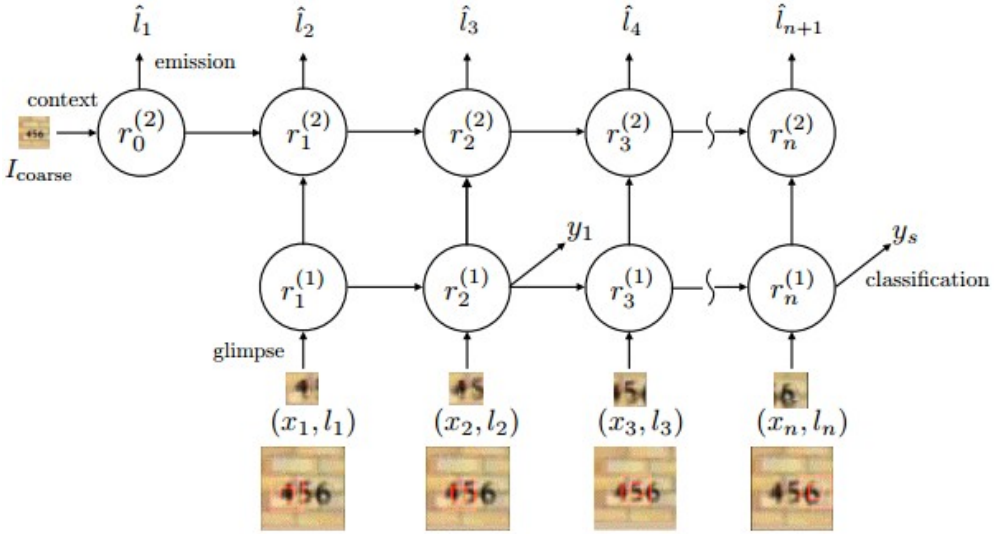
reward

estimate of future reward (discounted by \gamma)

$$\left[\left(r+\gamma \max_{a'} Q(s',a';\theta_i^-)\right)-Q(s,a;\theta_i)\right]^2$$

target value predicted value

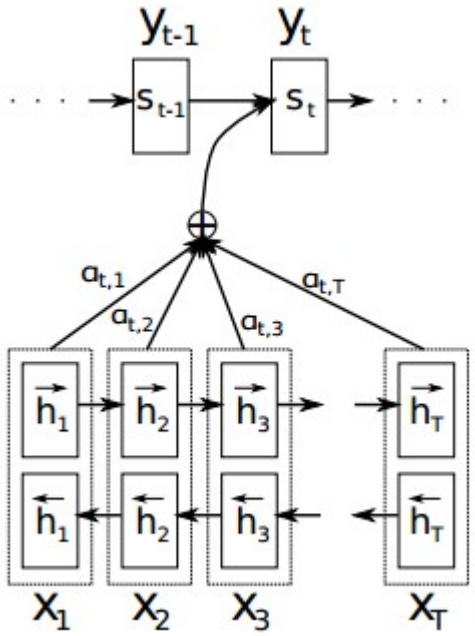
Recurrent Attention Models



Multiple Object Recognition with Visual Attention
[Jimmy Lei Ba, Volodymyr Mnih, Koray Kavukcuoglu], 2014

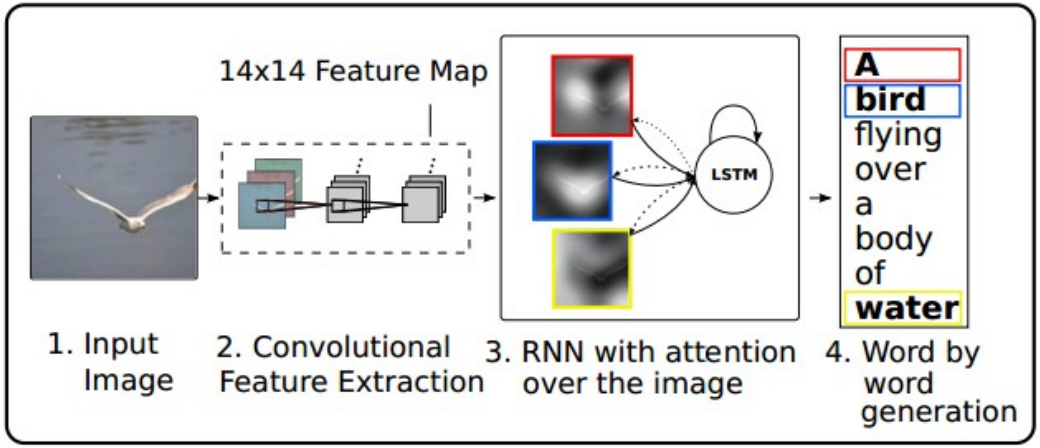
web demo
http://www.psi.toronto.edu/~jimmy/dram/
also DRAW: https://www.youtube.com/watch?v=Zt7MI9eKEo

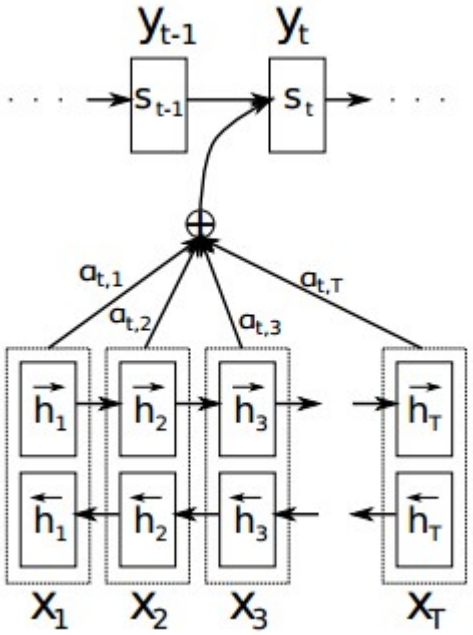
Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
[Kiros et al.] 2015



Neural machine translation by jointly learning to align and translate.
[Bahdanau et al.], 2014

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
[Kiros et al.] 2015





Neural machine translation by jointly learning to align and translate.
[Bahdanau et al.], 2014


A(1.00)


woman(0.80)


holding(0.68)


A(0.99)


large(0.49)


white(0.40)


a(0.58)


clock(0.62)


in(0.45)


her(0.39)


bird(0.35)


standing(0.29)


in(0.27)


a(0.35)


hand(0.64)


.(0.37)


forest(0.54)


.(0.46)

END

Course Instructors

Fei-Fei Li

Andrej Karpathy

Teaching Assistants

Justin Johnson

Yuke Zhu

Brett Kuprel

Ben Poole