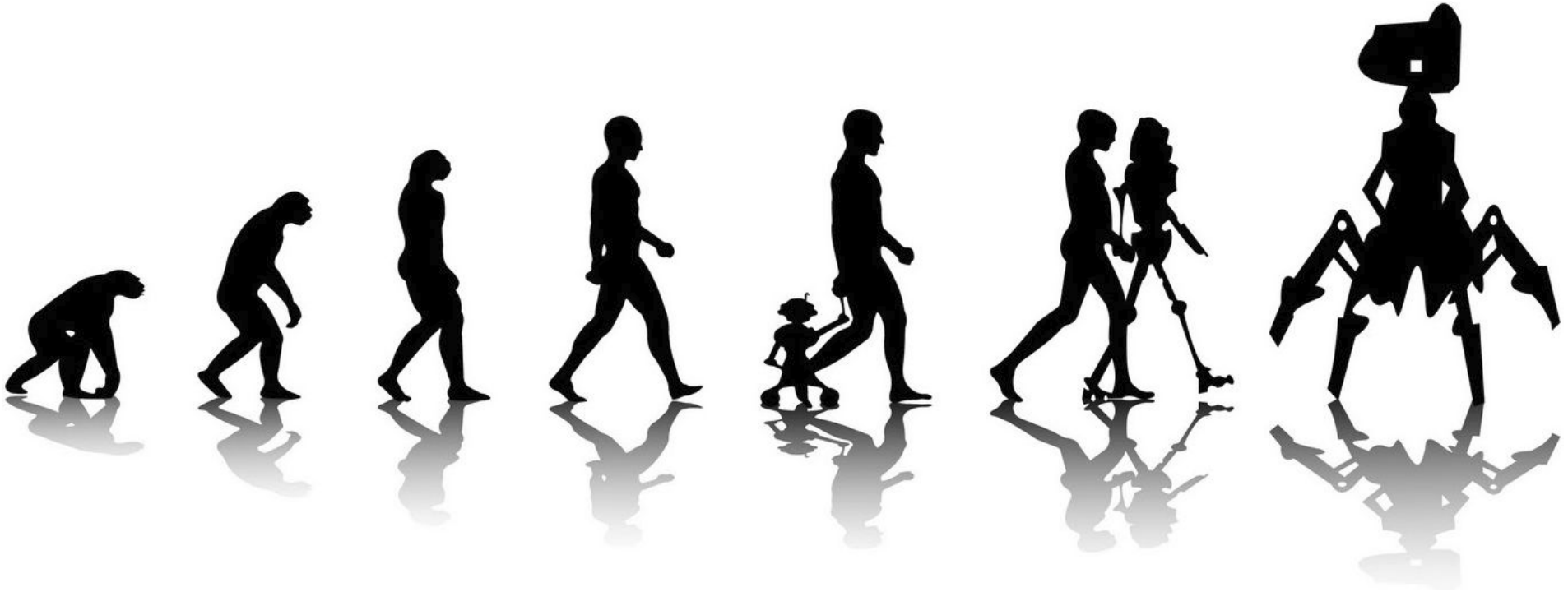


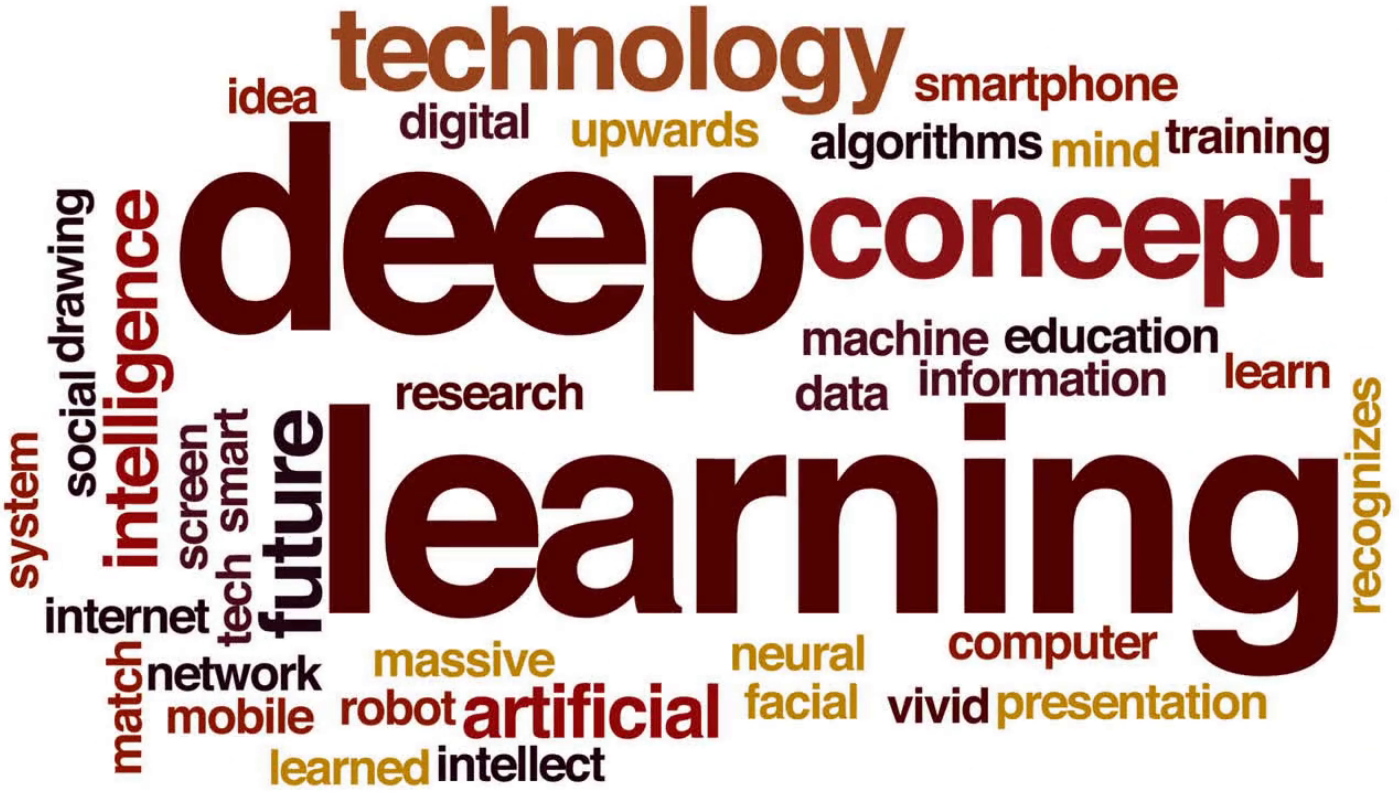
Modern Robotics: Evolutionary Robotics

COSC 4560 / COSC 5560

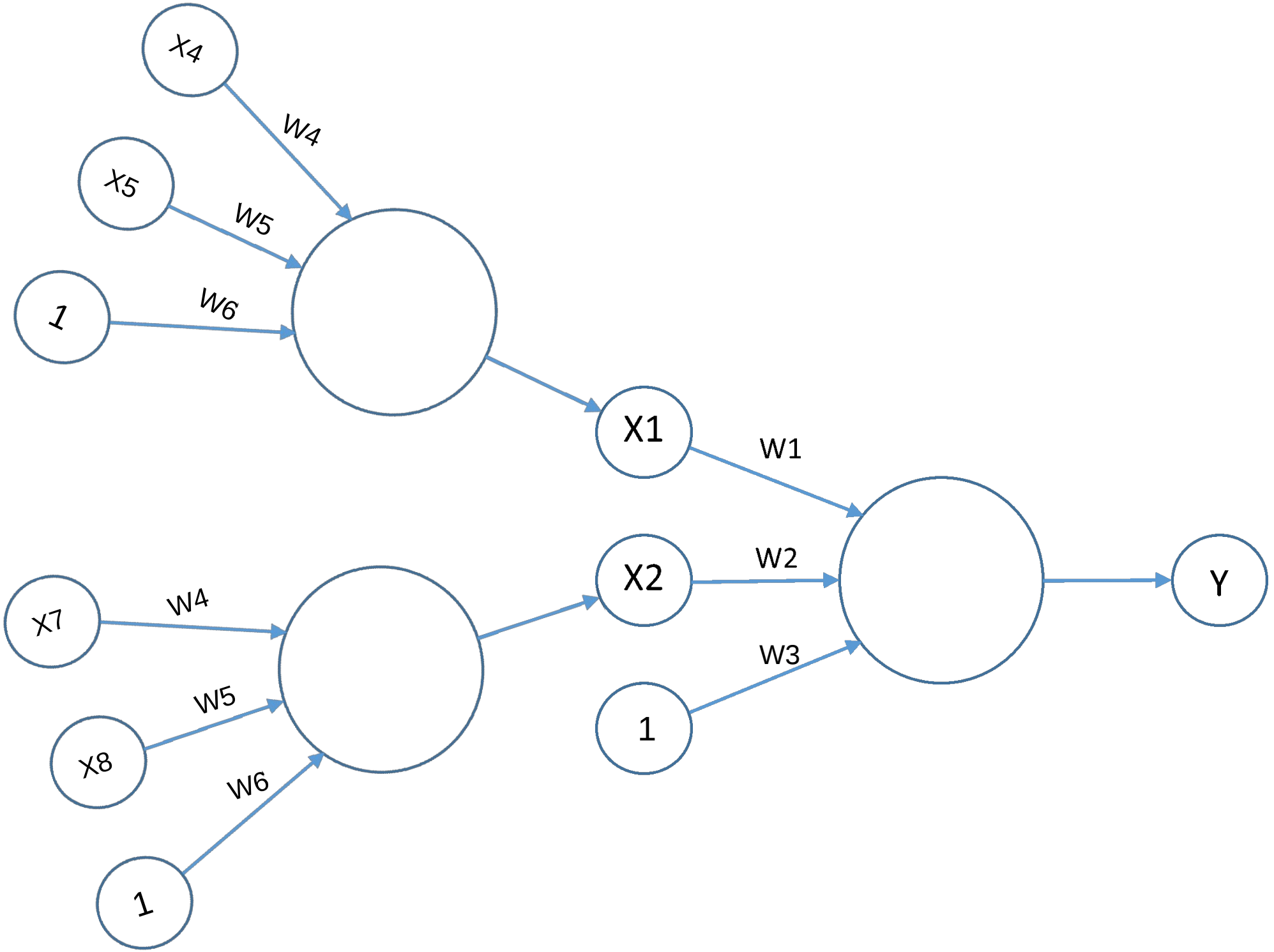
Professor Cheney

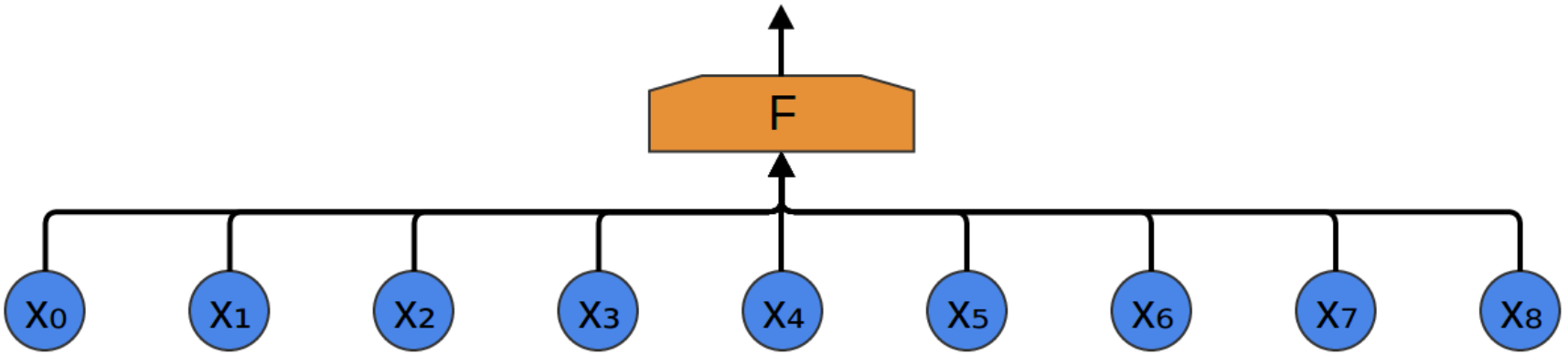
4/18/18

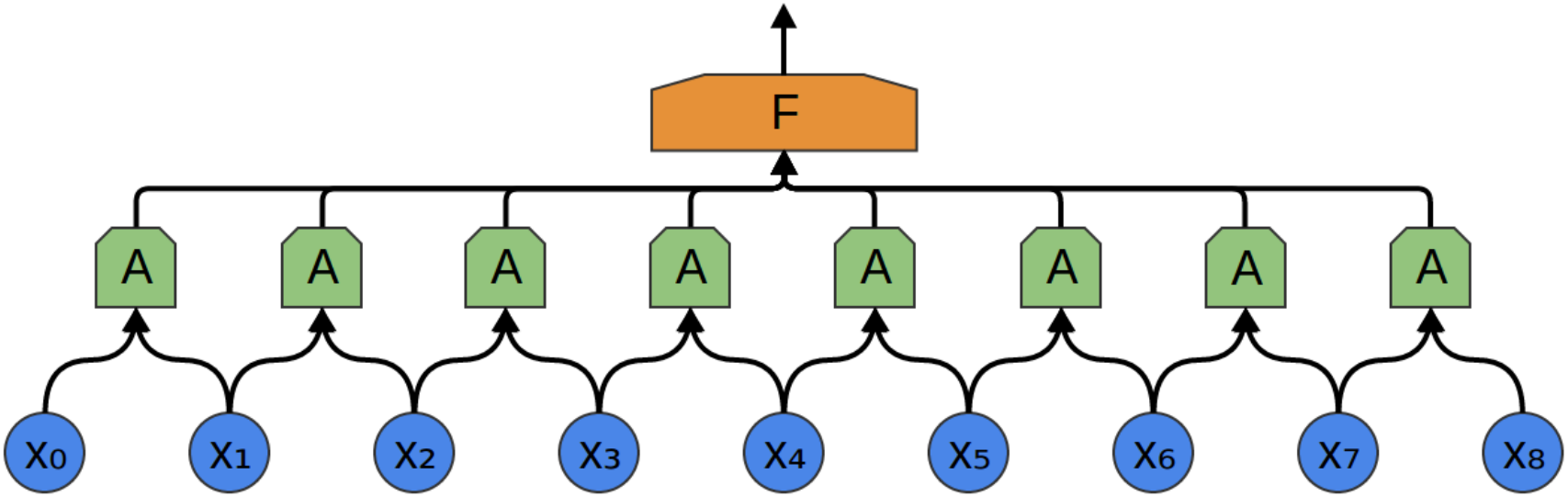
Tricks that make this work/scale!

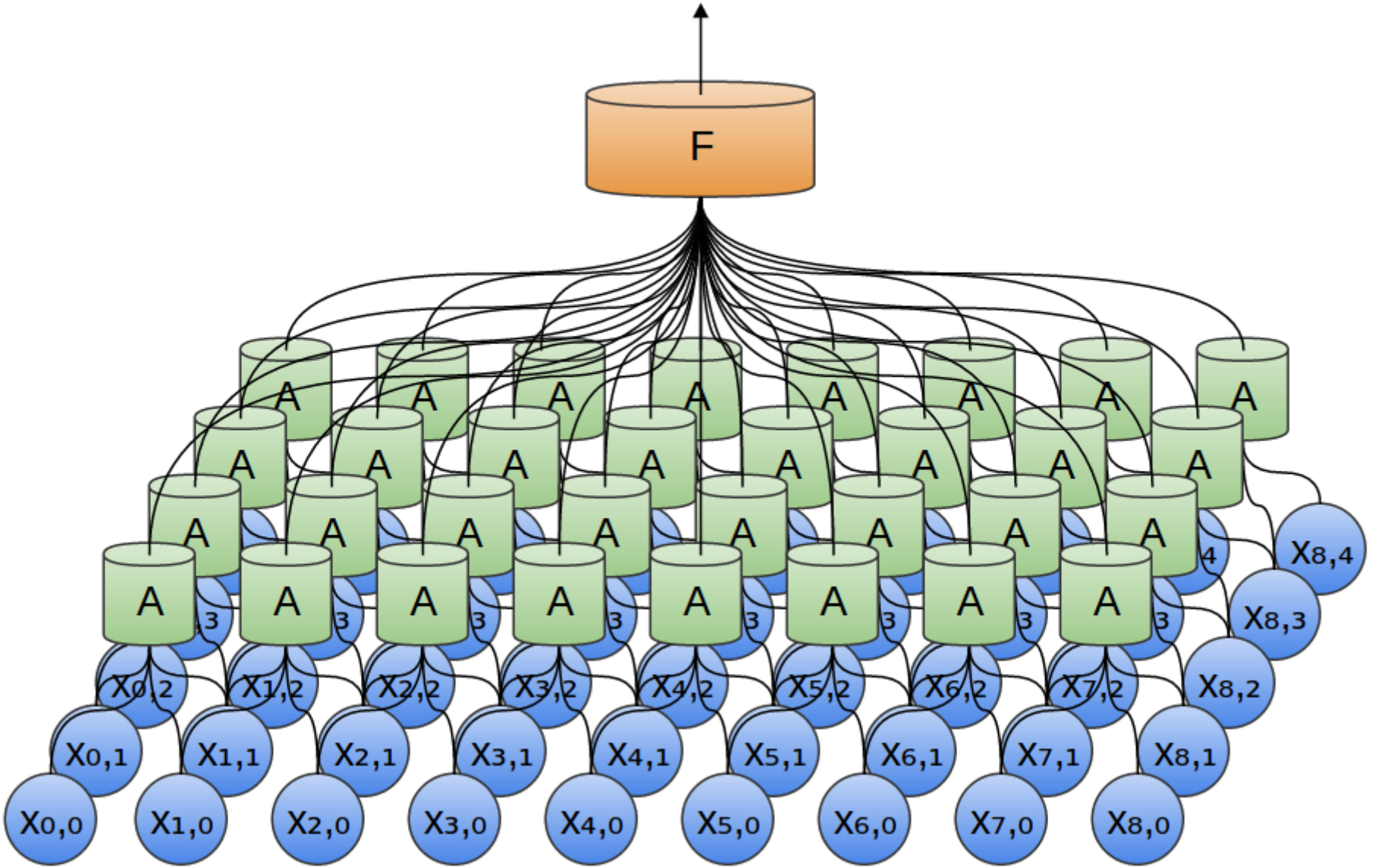

technology
idea
digital
upwards
smartphone
algorithms
mind
training
deep
concept
drawing
social
intelligence
machine
education
research
data
information
learn
system
screen
smart
future
learning
recognizes
internet
tech
match
network
massive
neural
computer
mobile
robot
artificial
facial
vivid
presentation
learned
intellect

Weight Sharing (via convolution)

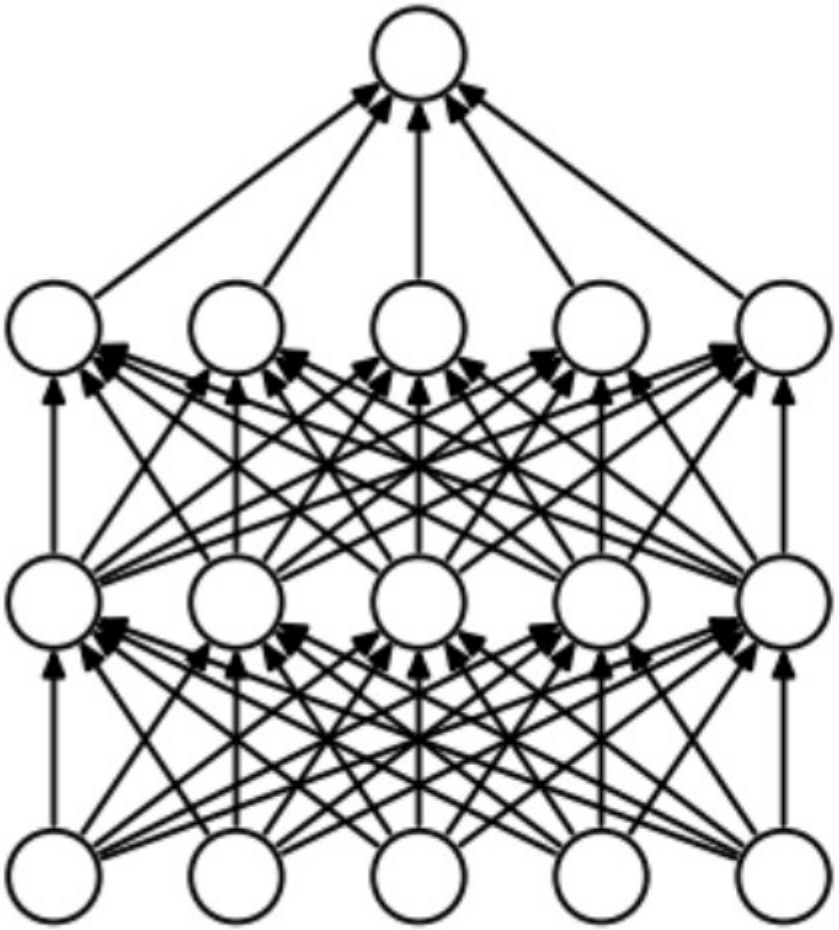

X4
W4
X5
W5
1
W6
X1
W1
X7
W4
X2
W2
Y
W5
X8
W3
1
W6
1


F
X0
X1
X2
X3
X4
X5
X6
X7
X8

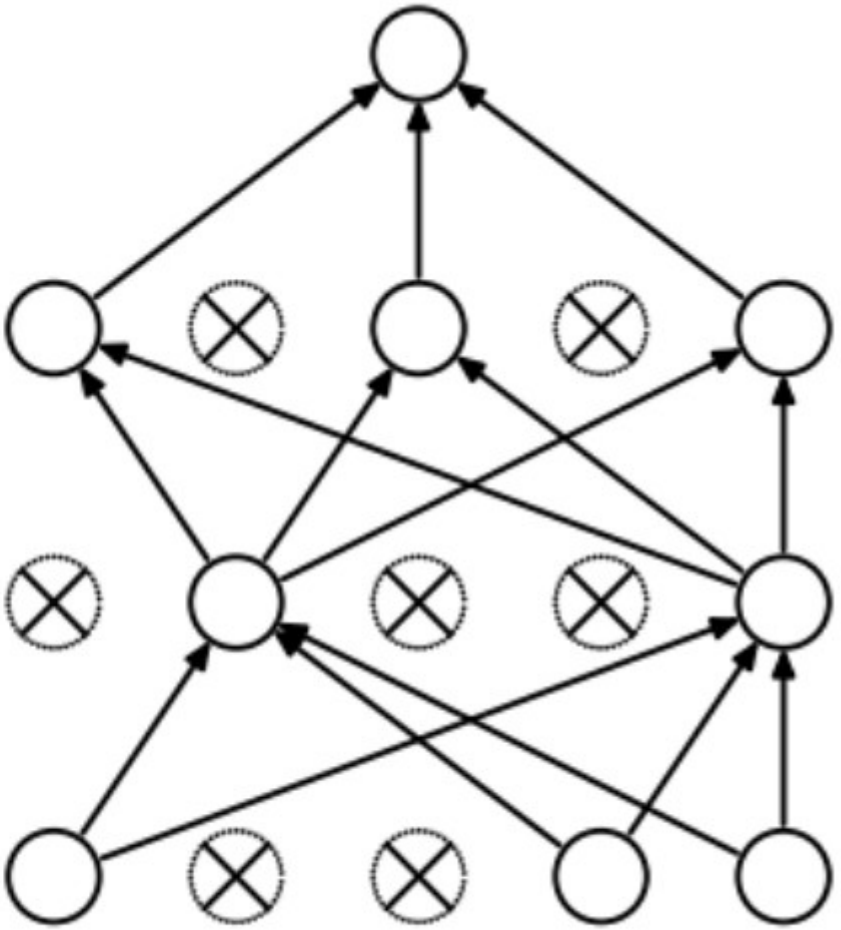

F
A
A
A
A
A
A
A
A
X0
X1
X2
X3
X4
X5
X6
X7
X8


F
A
X0,0
X1,0
X2,0
X3,0
X4,0
X5,0
X6,0
X7,0
X8,0
X0,1
X1,1
X2,1
X3,1
X4,1
X5,1
X6,1
X7,1
X8,1
X0,2
X1,2
X2,2
X3,2
X4,2
X5,2
X6,2
X7,2
X8,2
X8,3
X8,4

Dropout



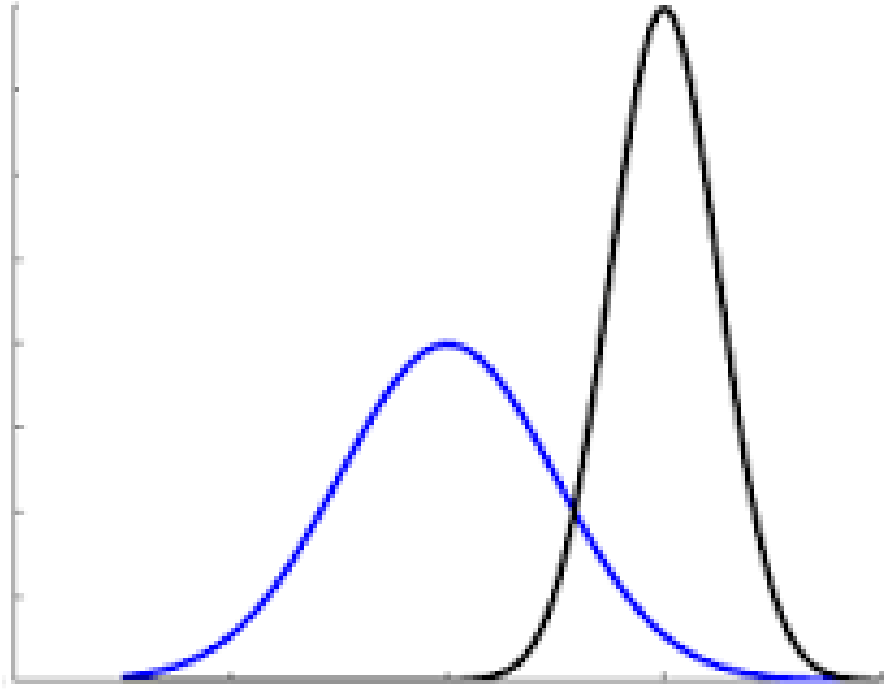
(a) Standard Neural Net



(b) After applying dropout.

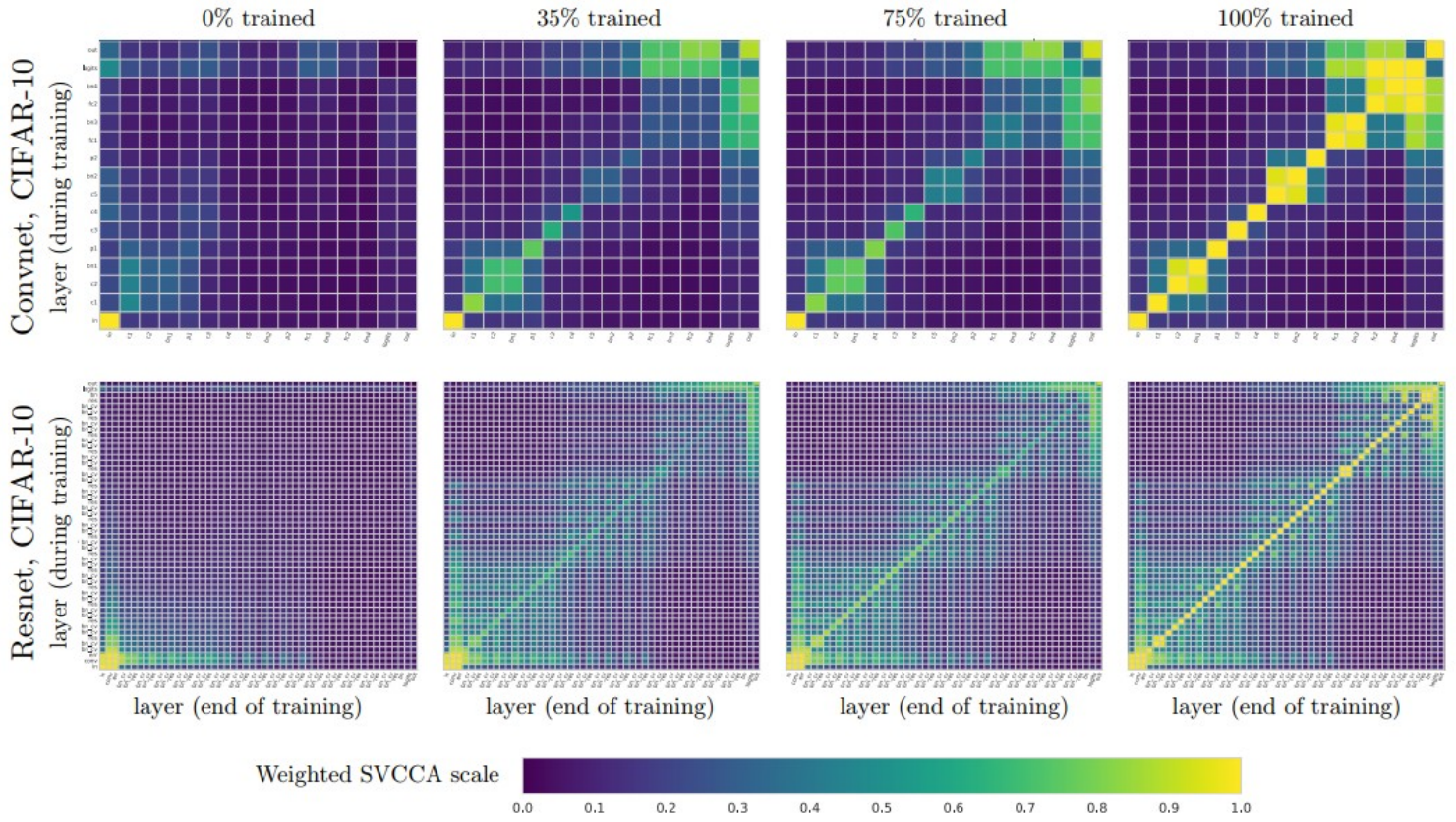
Batch Normalization

Internal Covariate Shift



SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability

Maithra Raghu,[1,2] **Justin Gilmer,**[1] **Jason Yosinski,**[3] **& Jascha Sohl-Dickstein**[1]
[1]Google Brain [2]Cornell University [3]Uber AI Labs
maithrar@gmail.com, gilmer@google.com, yosinski@uber.com, jaschasd@google.com



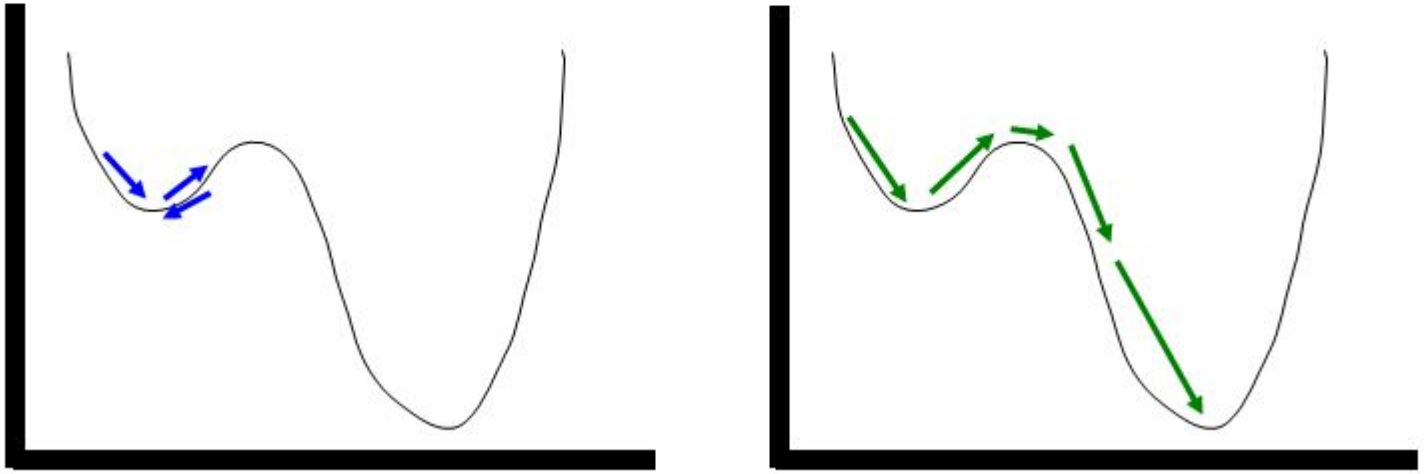
1. compute the empirical mean and variance independently for each dimension.

2. Normalize

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

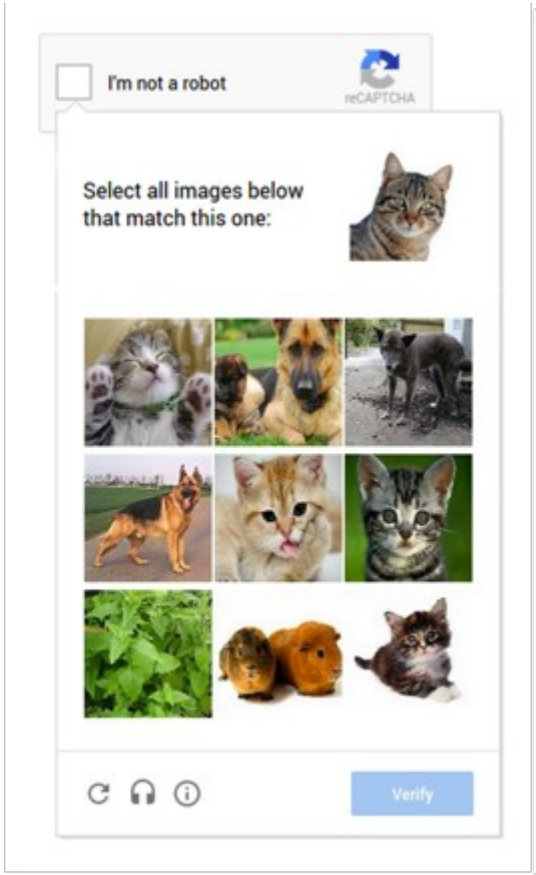
Momentum

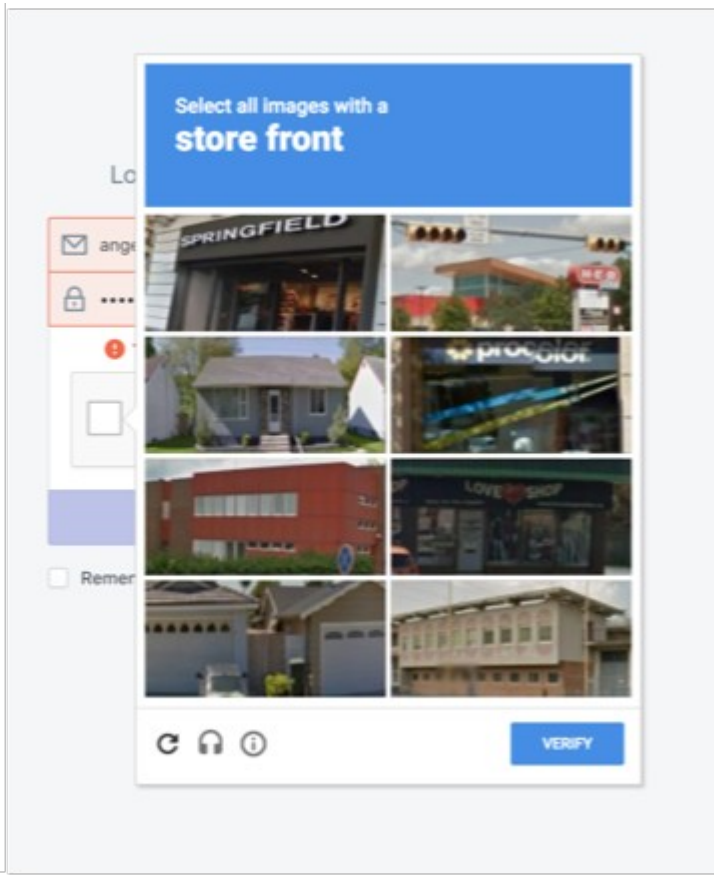
$$\Delta w_{ij}(k) = -\eta \nabla E + \beta \cdot \Delta w_{ij}(k-1)$$



Big Data


I'm not a robot
reCAPTCHA
Type the text
250
Verify


I'm not a robot
reCAPTCHA
Select all images below
that match this one:
Verify


Select all images with a
store front
SPRINGFIELD
VERIFY

ImageNet Challenge

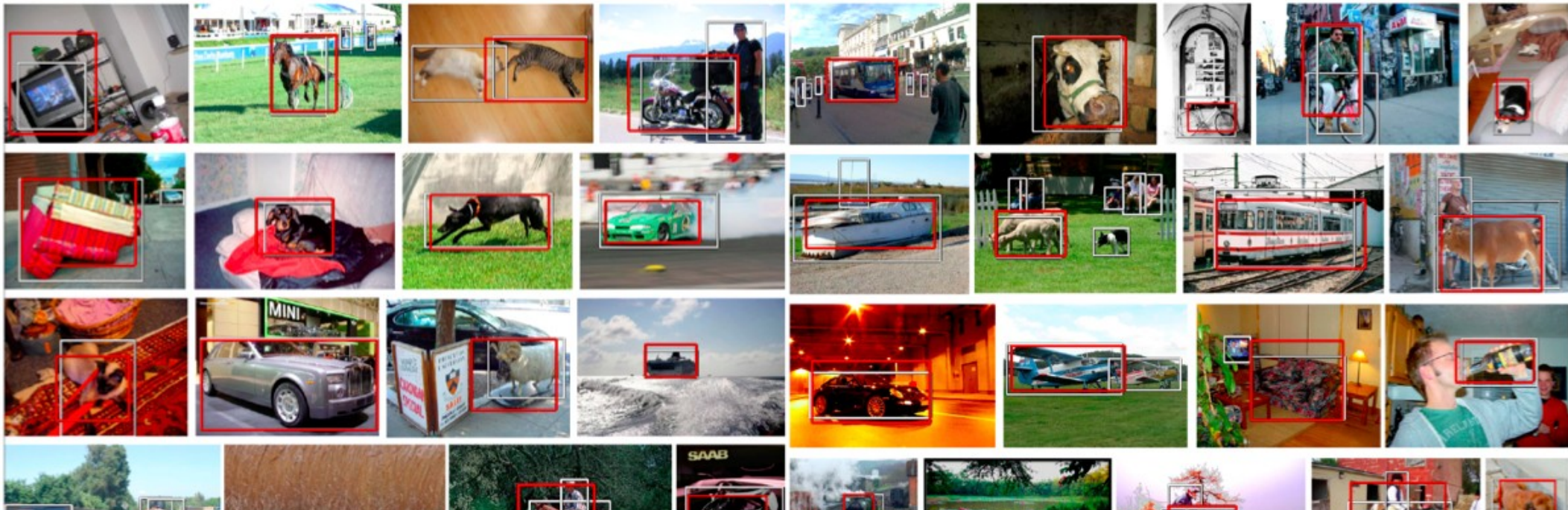
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Visual Object Classes Challenge 2009 (VOC2009)

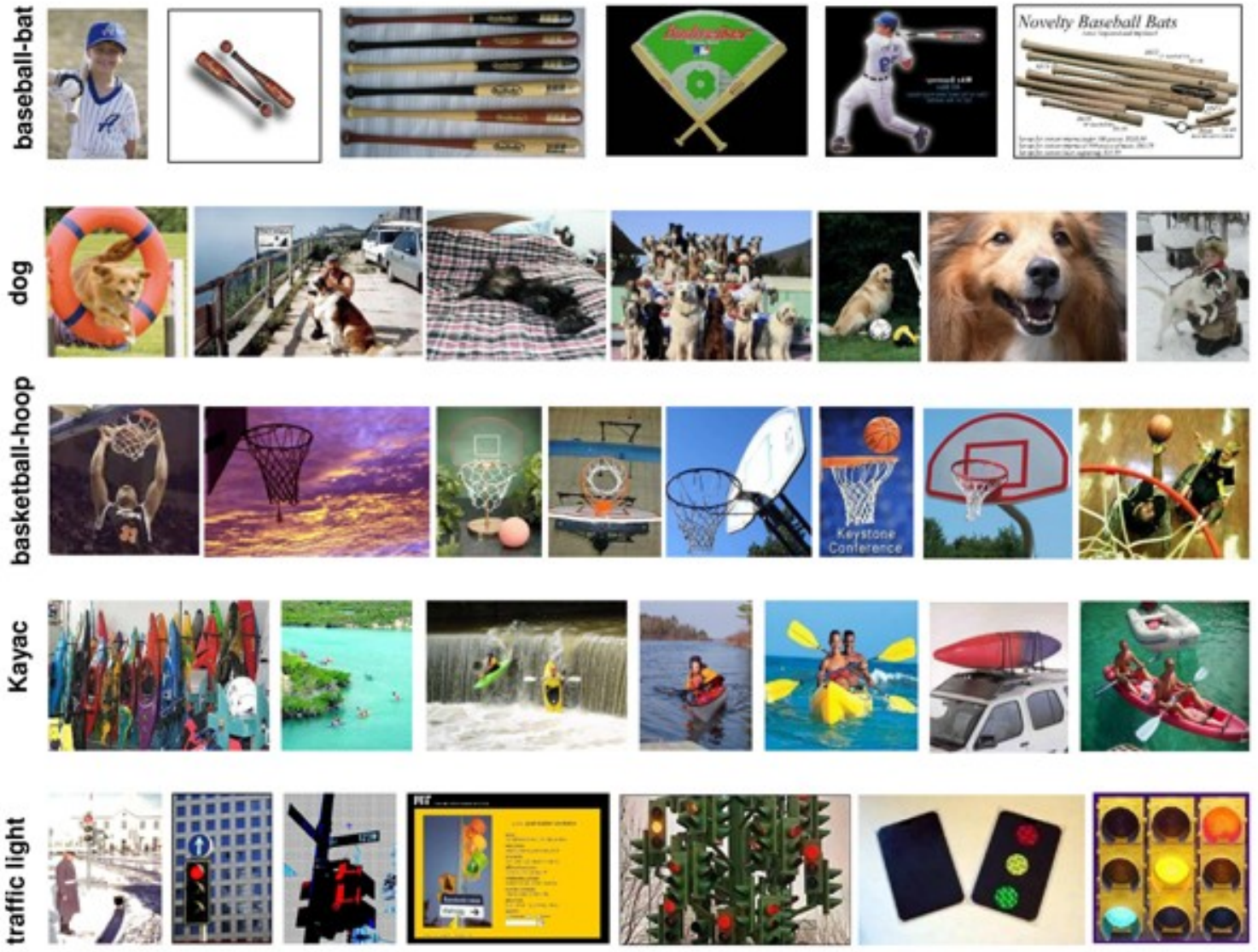
PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.



Caltech 256

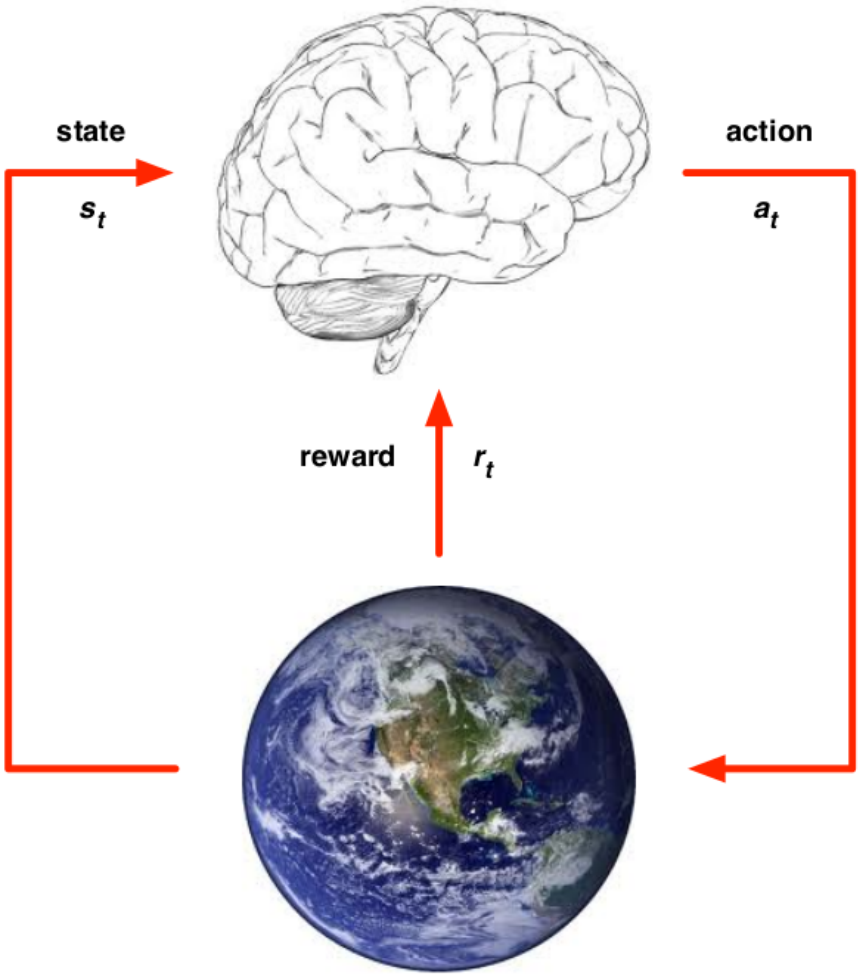
(256 object categories, 30607 images)



Deep Reinforcement Learning

David Silver, Google DeepMind

Agent and Environment



- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_t
  - Emits scalar reward r_t

Approaches To Reinforcement Learning

Policy-based RL

- Search directly for the optimal policy π^*
- This is the policy achieving maximum future reward

Value-based RL

- Estimate the optimal value function $Q^*(s, a)$
- This is the maximum value achievable under any policy

Model-based RL

- Build a transition model of the environment
- Plan (e.g. by lookahead) using model

Deep Reinforcement Learning

- Can we apply deep learning to RL?
- Use deep network to represent value function / policy / model
- Optimise value function / policy /model end-to-end
- Using stochastic gradient descent

learning value networks

Bellman Equation

- Value function can be unrolled recursively

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right] \\ &= \mathbb{E}_{s'}\left[r + \gamma Q^{\pi}(s', a') \mid s, a\right] \end{aligned}$$

- Optimal value function $Q^*(s,a)$ can be unrolled recursively

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s', a') \mid s, a\right]$$

- Value iteration algorithms solve the Bellman equation

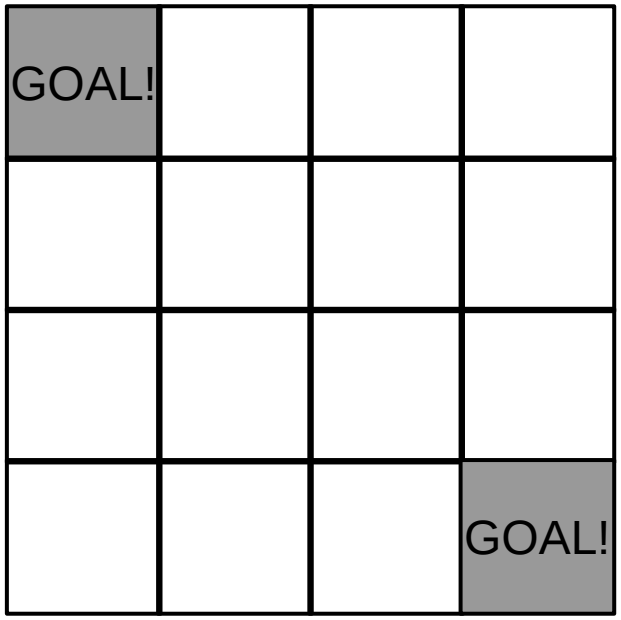
$$Q_{i+1}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q_i(s', a') \mid s, a\right]$$

grid world:

each timestep has -1 reward

the game terminates when
you reach a goal state

actions: N, S, E, W



intuitive description: “get to the goal as soon as possible”
(but let's pretend we're a robot, who doesn't know this!)

each value function (V) is defined
with respect to some behavioral policy (π)

V^{π}

let's iteratively find V^{π} for a random policy
in our mini grid world

current value (V_k) for a random policy

k=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

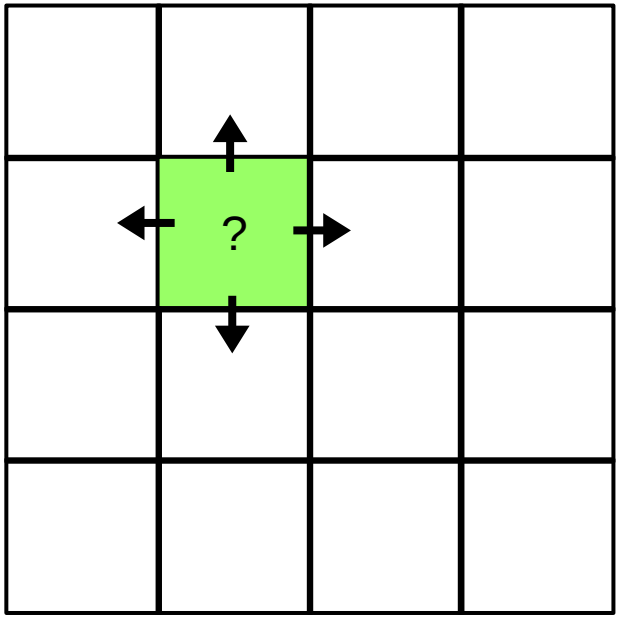
k=1

current value (V_k) for a random policy

k=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

k=1

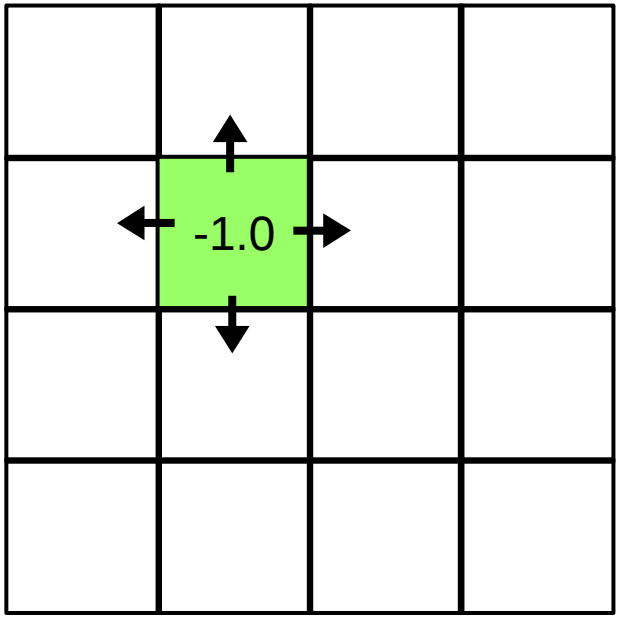


current value (V_k) for a random policy

k=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

k=1



current prediction for cumulative reward in new state

immediate reward

N: $V_{s,N} = -1 + 0 = -1$

S: $V_{s,S} = -1 + 0 = -1$

E: $V_{s,E} = -1 + 0 = -1$

W: $V_{s,W} = -1 + 0 = -1$

with a random policy,
we are equally likely to take any move,
so:

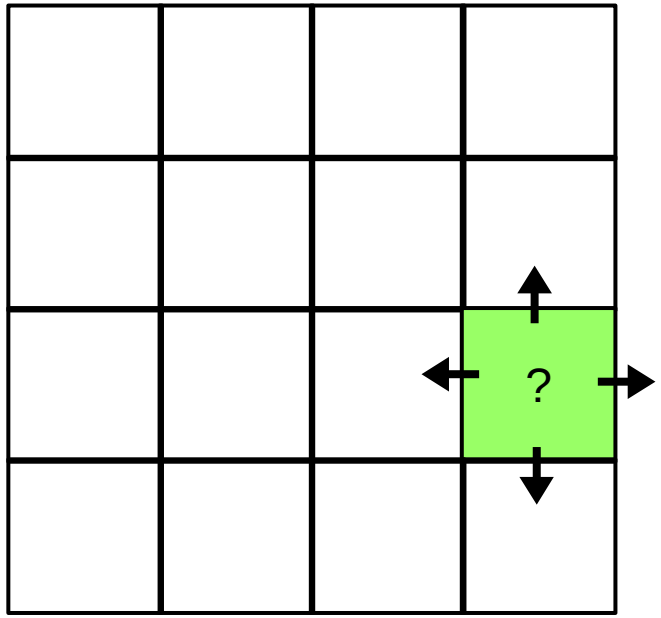
$V_s = (-1 + -1 + -1 + -1)/4 = -1$

current value (V_k) for a random policy

k=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

k=1



current value (V_k) for a random policy

k=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

k=1

			-1.0

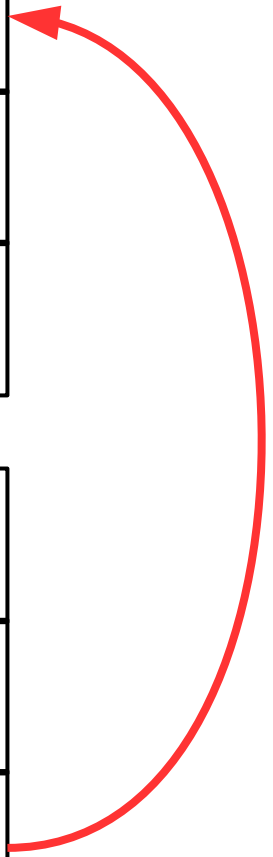
current value (V_k) for a random policy

k=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

k=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



next iteration…
new value function
becomes
old value function
(“current prediction
for cumulative reward”)

current value (V_k) for a random policy

k=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

k=2

current value (V_k) for a random policy

k=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

k=2

	-1.75		

N: $V_{s,N} = -1 + -1 = -2$
S: $V_{s,S} = -1 + -1 = -2$
E: $V_{s,E} = -1 + 0 = -1$
W: $V_{s,W} = -1 + -1 = -2$

$V_s = (-2 + -2 + -1 + -2)/4 = -1.75$

current value (V_k) for a random policy

k=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

k=2

0.0	-1.75	-2.0	-2.0
-1.75	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.75
-2.0	-2.0	-1.75	0.0

current value (V_k) for a random policy

k=2

0.0	-1.75	-2.0	-2.0
-1.75	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.75
-2.0	-2.0	-1.75	0.0

k=3

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

current value (V_k) for a random policy

k=10

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

k=∞

0.0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0.0

converged to true value function ($V^{\pi\text{-random}}$)