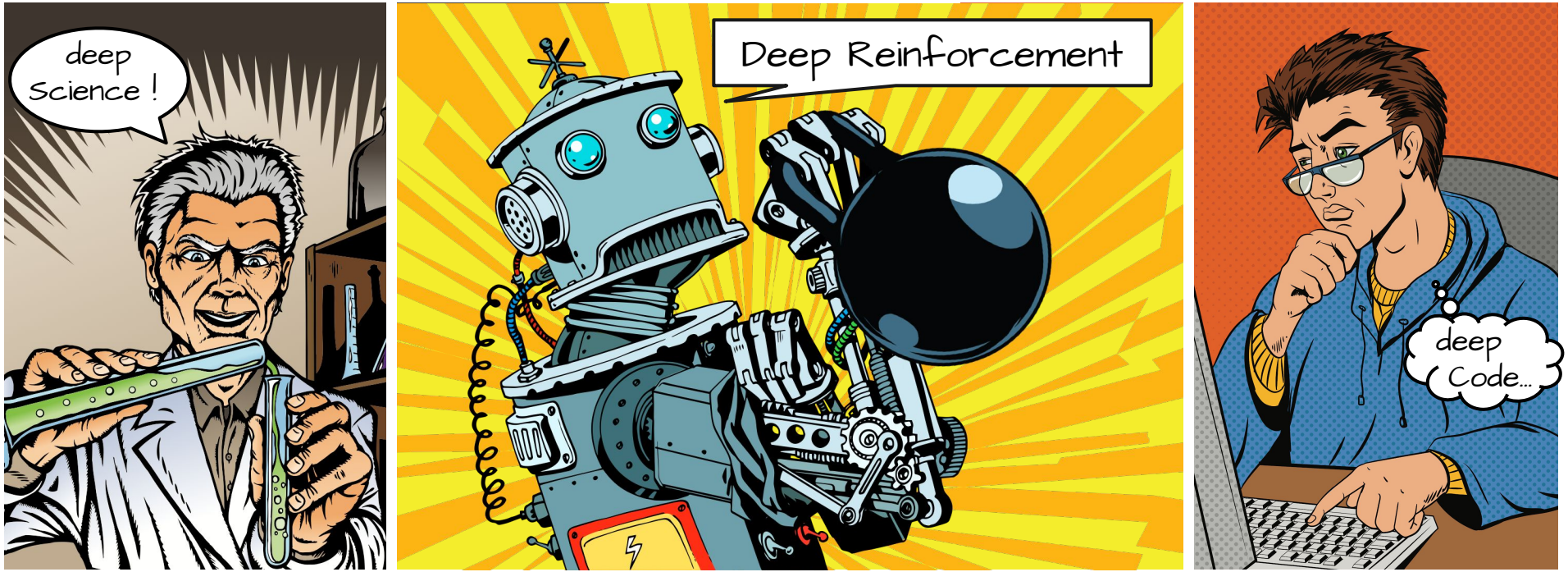
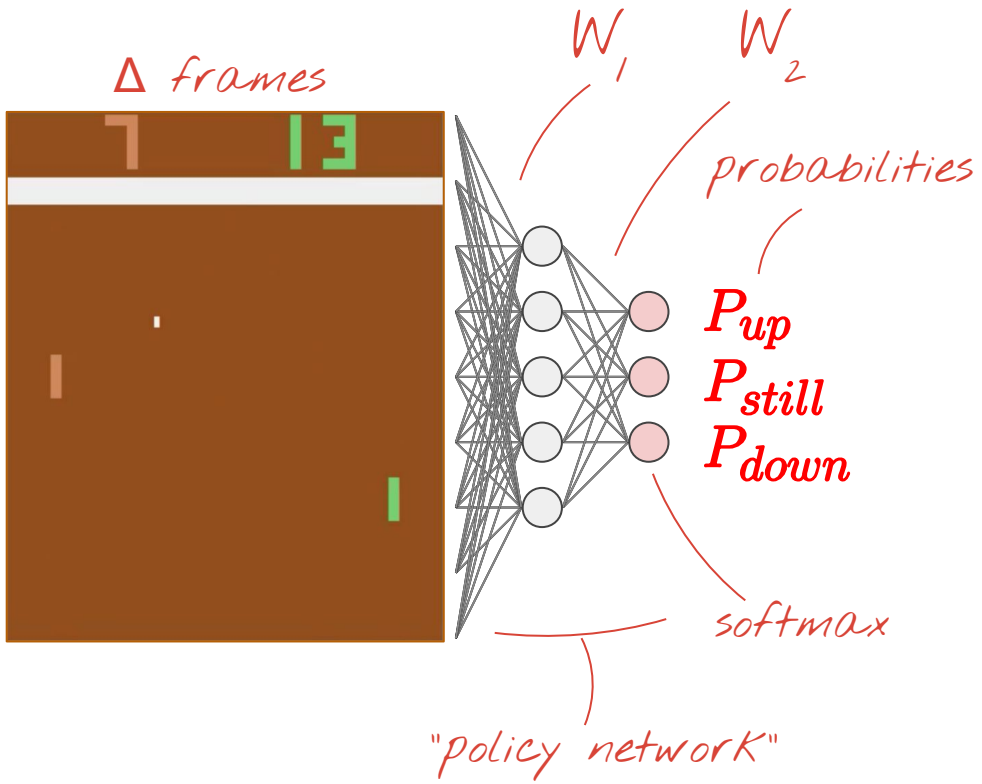
>TensorFlow and \
deep reinforcement learning
without a PhD_



Pong ?



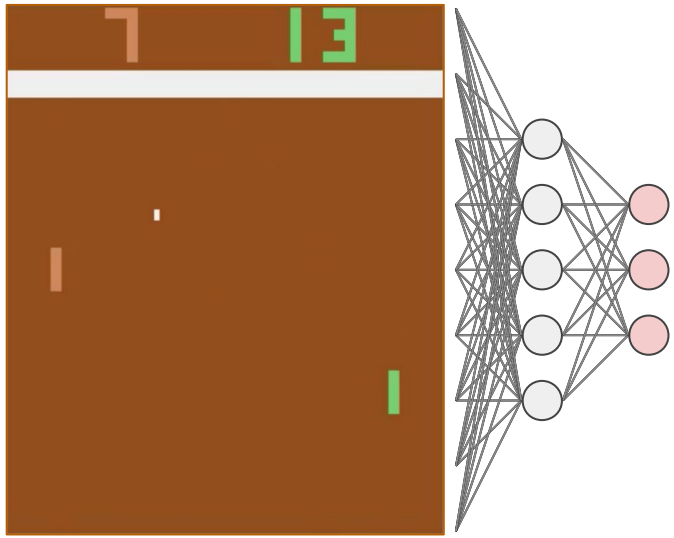
cross-entropy

$$loss = -(P'_{up}.log(P_{up}) + P'_{still}.log(P_{still}) + P'_{down}.log(P_{down}))$$

"correct move"
Ex: 1 0 0

Adapted from A. Karpathy's "Pong from Pixels" post

Pong ?



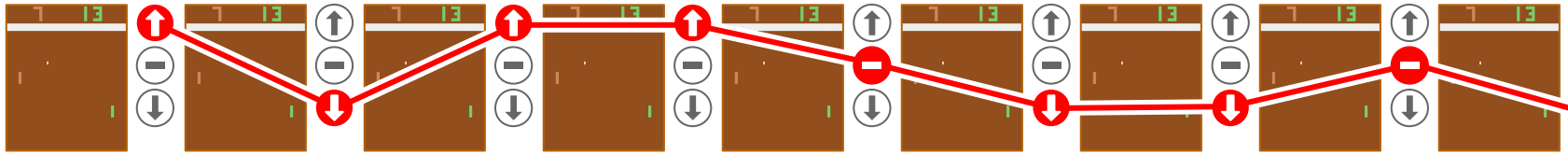
probabilities

P_{up}
P_{still}
P_{down}

UP → Keep playing
~~STILL~~
~~DOWN~~

sample from probabilities

Policy gradients



$$loss = -\boldsymbol{R}\,(P'_{up}.log(\boldsymbol{P_{up}}) + P'_{down}.log(\boldsymbol{P_{down}}) + P'_{down}.log(\boldsymbol{P_{down}}))$$

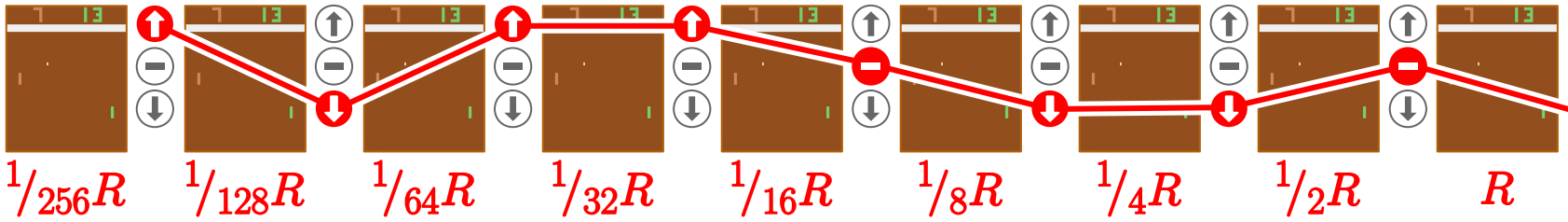
for each move

R: reward

"sampled move" Ex: 1 0 0

predicted

Policy gradient refinements



① discounted rewards

$$loss_i = R_i log(P_i^{up/still/down})$$

whichever was chosen

R: reward

② normalized losses

$$loss_i = \frac{loss_i - \overline{loss}}{stdev(loss)}$$

move #i

across a "batch" of moves

Training data

move #	[UP, STILL, DOWN]	Policy network probabilitiess	Discounted rewards
0	[1, 0, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d^7$
1	[0, 1, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d^6$
2	[0, 1, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d^5$
3	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d^4$
4	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d^3$
5	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d^2$
6	[0, 1, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$+1*d$
7	[1, 0, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	+1 ← WIN! +1
8	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d^7$
9	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d^6$
10	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d^5$
11	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d^4$
12	[0, 1, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d^3$
13	[1, 0, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d^2$
14	[0, 0, 1]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	$-1*d$
15	[0, 1, 0]	$[P_{UP}, P_{STILL}, P_{DOWN}]$	-1 ← LOSE -1

Hyperparams:
Batch size: ~20k moves
Discount factor d=0.95~0.99
Optimizer=tf.train.RMSPropOptimizer
- Learning rate lr=0.0001~0.005
- Decay=0.95~0.99

1 hidden layer with 200 units
Beta (“laziness”) = 0.01~0.02

Tensorflow code

```
observations = tf.placeholder(shape=[None, 80x80], dtype=tf.float32) # pixels
actions      = tf.placeholder(shape=[None]) # 0, 1, 2 for UP, STILL, DOWN
rewards      = tf.placeholder(shape=[None]) # +1, -1, with discounts

# model
Y = tf.layers.dense(observations, 200, activation=tf.nn.relu)
Ylogits = tf.layers.dense(Y, 3)

# sample an action from predicted probabilities
sample_op = tf.multinomial(logits=tf.reshape(Ylogits, shape=(1, 3)), num_samples=1)

#loss
cross_entropies = tf.losses.softmax_cross_entropy(one_hot_labels=tf.one_hot(actions,3),
                                                  logits=Ylogits)
loss = tf.reduce_sum(rewards * cross_entropies)

# training operation
optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001, decay=0.99)
train_op = optimizer.minimize(loss)
```

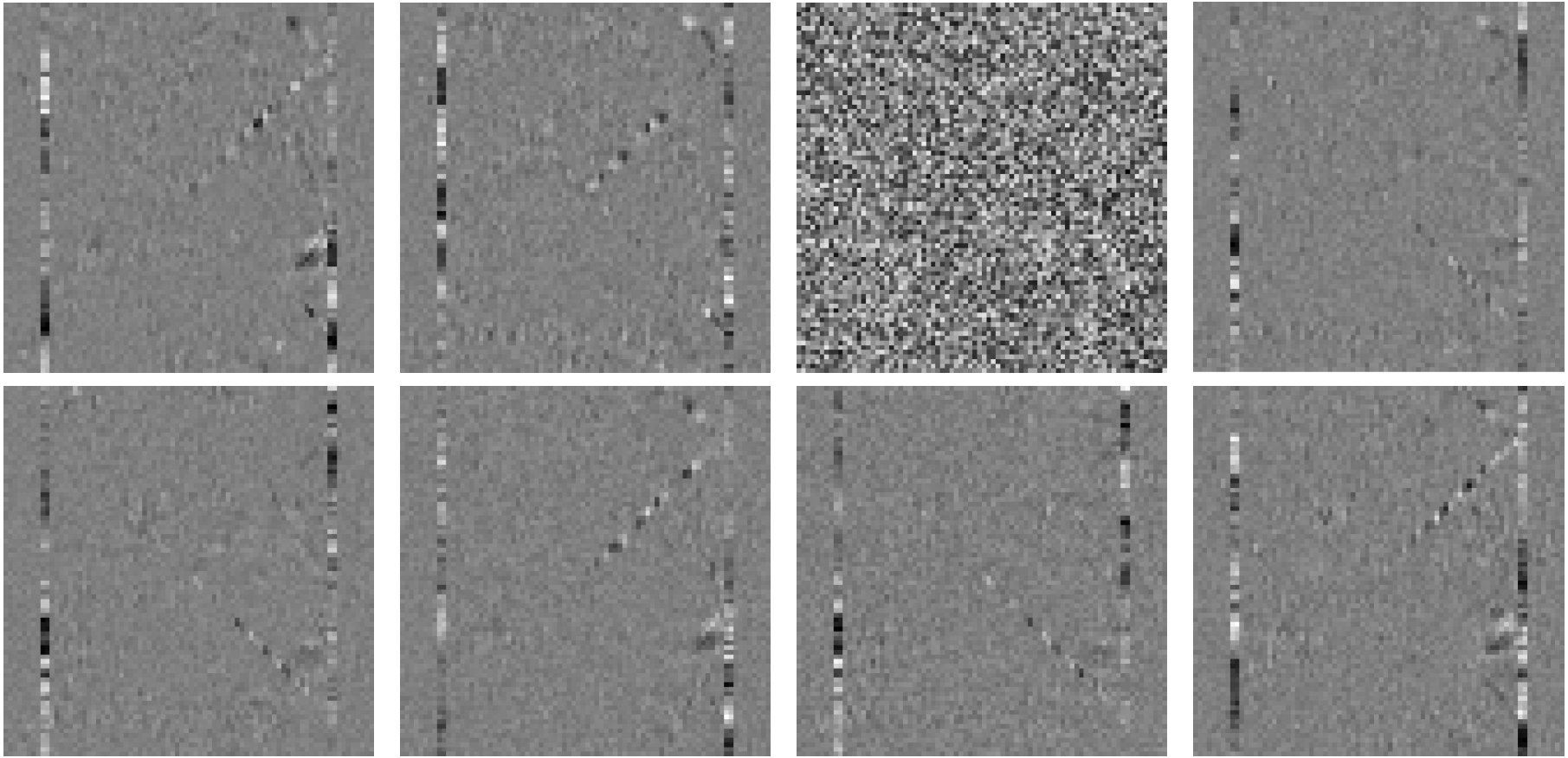
Training loop

```
with tf.Session() as sess:
    while len(observations) < BATCH_SIZE:
        … # reset everything
        while not done:        # play a game in 21 points
            # get pixels
            current_pix = read_pixels(game_state)
            observation = current_pix - previous_pix
            previous_pix = current_pix
            # decide what move to play: UP, STILL, DOWN (through NN model)
            action = sess.run(sample_op, feed_dict={observations: [observation]})
            # play it (through opneAI gym pong simulator)
            game_state, reward, done, info = pong_sim.step(action)
            # collect results
            observations.append(observation)
            actions.append(action)
            rewards.append(reward)

        # Process the rewards after each episode
        processed_rewards = discount_rewards(rewards, args.gamma)
        processed_rewards = normalize_rewards(rewards, args.gamma)

    feed_dict = {observations: observations, actions: actions, rewards: processed_rewards }
    sess.run(train_op, feed_dict=feed_dict)
```

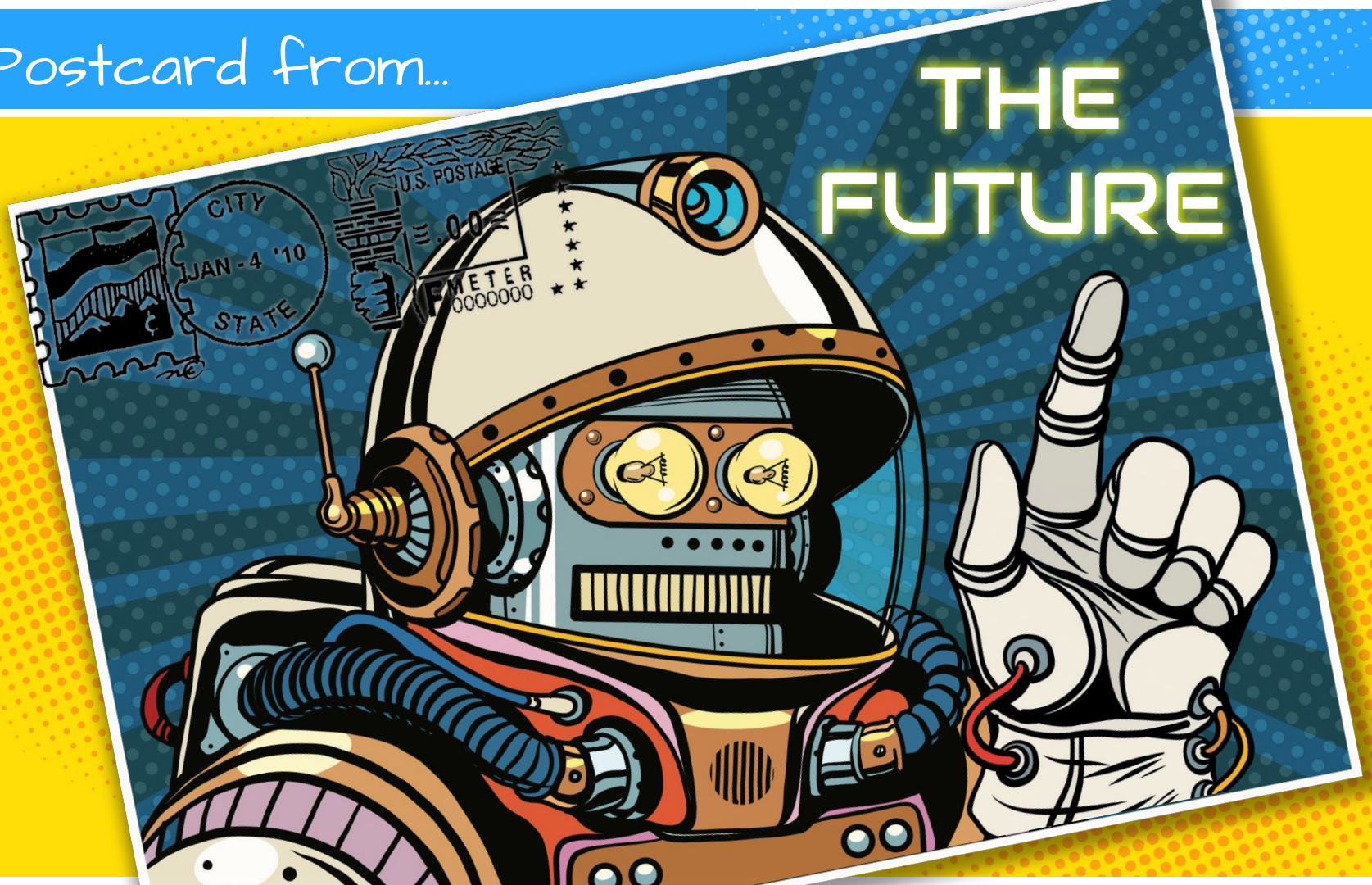
one training iteration

Pong!

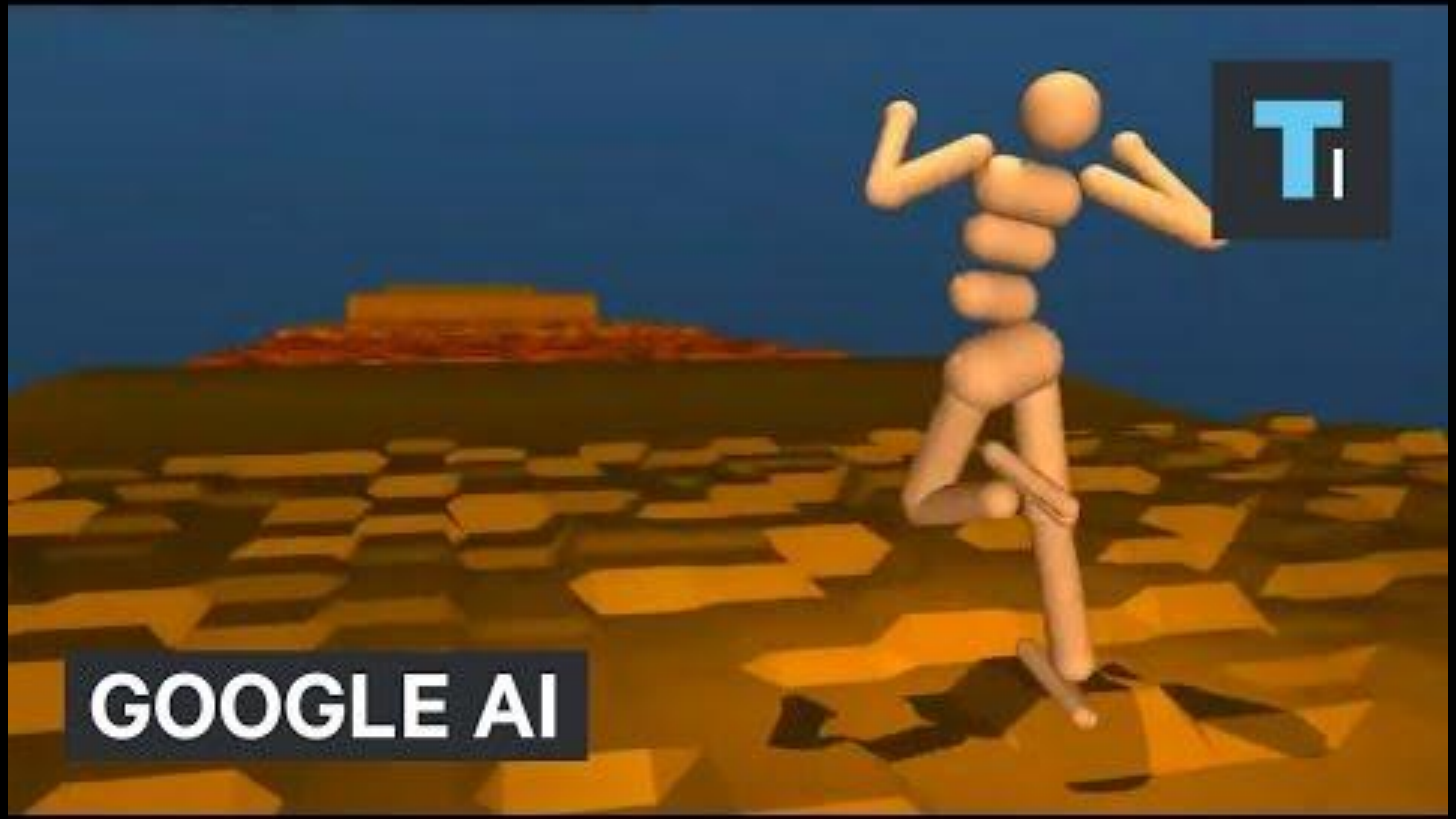
Learned weights



Postcard from...

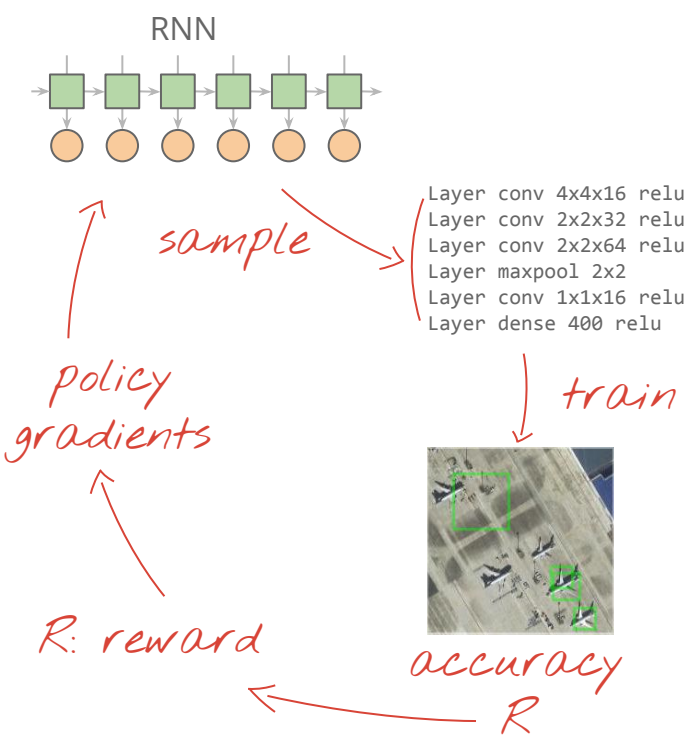


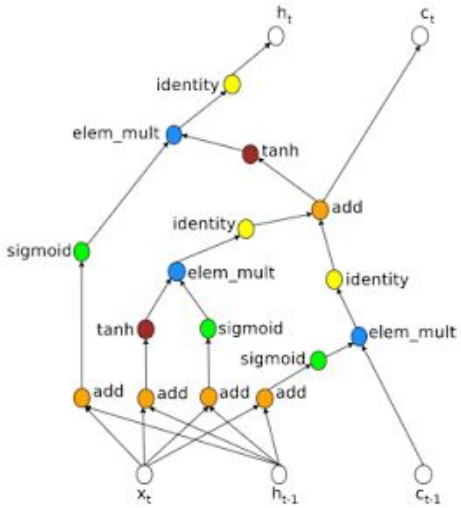

Reinforcement Learning
First trial...


GOOGLE AI

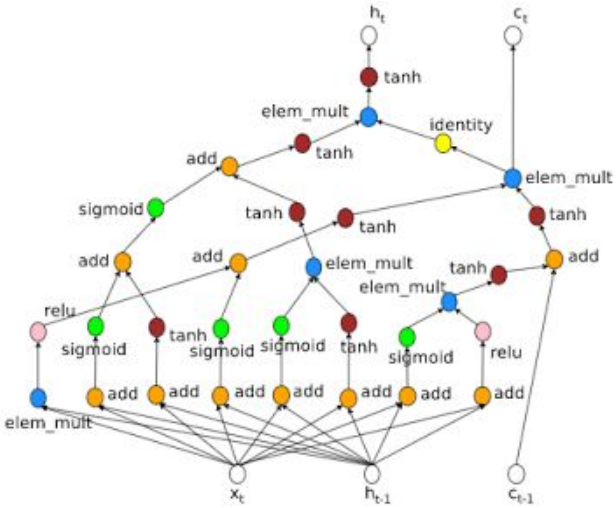

Google DeepMind
Challenge Match
8 - 15 March 2016
AlphaGo vs Lee Sedol
Match 4 - Livestream
13th March 13:00 KST, 04:00 GMT
-1 day (12th March) 20:00 PT, 23:00 ET
Pre-Match Commentary starting at 12:45 KST,
03:45 GMT -1day (12th March) 19:45 PT, 22:45 ET
Live from the Four Seasons Hotel Seoul!

Auto ML





LSTM cell



new RNN cell

arXiv:1611.01578v2, Barret Zoph, Quoc V. Le, May 2017

Replay on youtu.be/aRKOJHRbXeo

Have fun!

Martin Görner
Google Developer relations
@martin_gorner

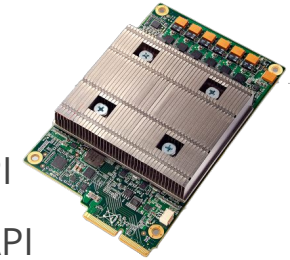
Yu-Han Liu
Google Developer relations
yuhanliu@google.com

Neeraj Kashyap
Google Developer relations
nkash@google.com

TensorFlow
tensorflow.org

Google Cloud Platform - cloud.google.com

Cloud ML Engine
your TensorFlow models trained in Google's cloud, fast.



Pre-trained models:

- Cloud Vision API
- Cloud Speech API
- Natural Language API
- Google Translate API
- Video Intelligence API BETA
- Cloud Jobs API PRIVATE BETA

This presentation:
goo.gl/CB8xNH
Tensorflow w/o a PhD:
part 1: goo.gl/pHeXe7
part 2: goo.gl/UuN41S
part 3: goo.gl/VxQDmx
part 4: goo.gl/BYT7au
part 5: goo.gl/CB8xNH
codelab: goo.gl/mVZloU

1. youtu.be/u4alGiomYP4
2. youtu.be/fTUwdXUFfI8
3. youtu.be/vaL1l2BD_xY
4. youtu.be/aRKOJHRbXeo
5. youtu.be/vq2nnJ4g6N0?t=76m
6. youtu.be/pzOzmxCR37I