

AI and Deep Learning

Applications

Jeju National University

Yung-Cheol Byun

Security

Prediction

Object recognition

Art, creation

Poet

Drawing

Composition

Healthcare

Assistant

Translation

Unmanned car

Autonomous drone

Robot

Interpretation

other applications ...

Object Recognition



Security

Real-time event detection for video surveillance applications

powered by



Robot, learn to work



Robot moving like human



Humanoid



HANSON
ROBOTICS

Learning how to play



Ping-Pong Match



HEIGHT: 181cm, WEIGHT: 78kg, AGE: 32

Image Captioning

A steam locomotive is traveling through a forest with autumn foliage. The locomotive is black and is emitting a small amount of smoke from its chimney. The trees are covered in yellow, orange, and red leaves, indicating the fall season. The locomotive is moving along a track that curves through the forest.

Artificial Intelligence

AI Guide

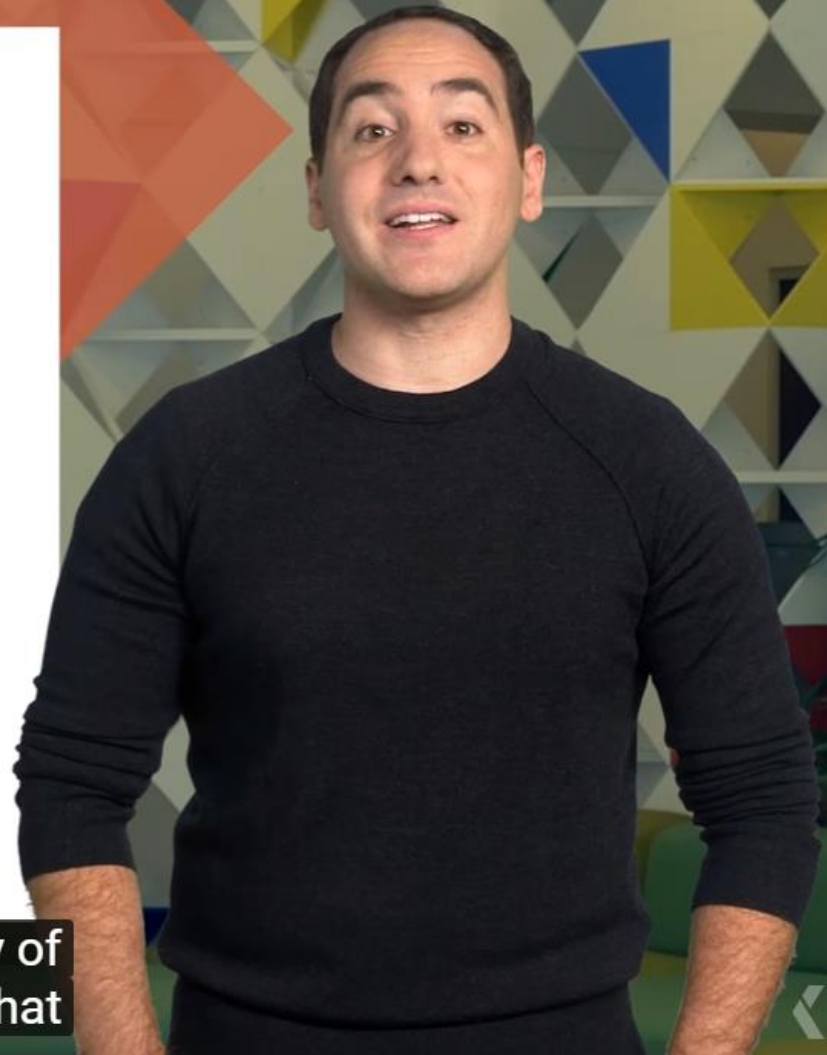
a car is parked at a stop light



Voice Recognition

*Learn from examples
and experience.*

It's the study of
algorithms that



Stock Prediction

- Stock Price Data from Yahoo
- Open, Highest, Lowest, Volume, Close
- 734 daily
- 70% for training, 30% for test

Stock Prediction

plt

at 0x1152171

lab-12-1-hello-rnn3.py	47
lab-12-2-char-seq-rnn2.py	48
lab-12-4-rnn_long_char2.py	49
lab-12-5-rnn_stock_prediction2.py	50
lab-12-6-rnn_softmax_stock_prediction2.py	51
lab-13-2-mnist_tensorboard2.py	52
lab-13-3-mnist_save_restore2.py	53
layer_test.py	54
my_rnn.py	55
README.md	56
rnn.py	57
rnn2.py	58
rnn_basic.py	59
rnn_basic-original.py	60
External Libraries	61

Run lab-12-6-rnn_softmax_stock_prediction2



```
self.preprocessing()
```

```
originalX = self.nati
```

```
originalY = self.nati
```

```
# print(len(self.x))
```

```
#cut_and_append, 시계열
```

```
dataX = []
```

```
dataY = []
```

```
# append 횟수는? 전체 줄
```

```
for i in range(0, len
```

```
    _x = originalX[i:
```

```
    _y = originalY[i
```

```
    #print(_x, "->",
```

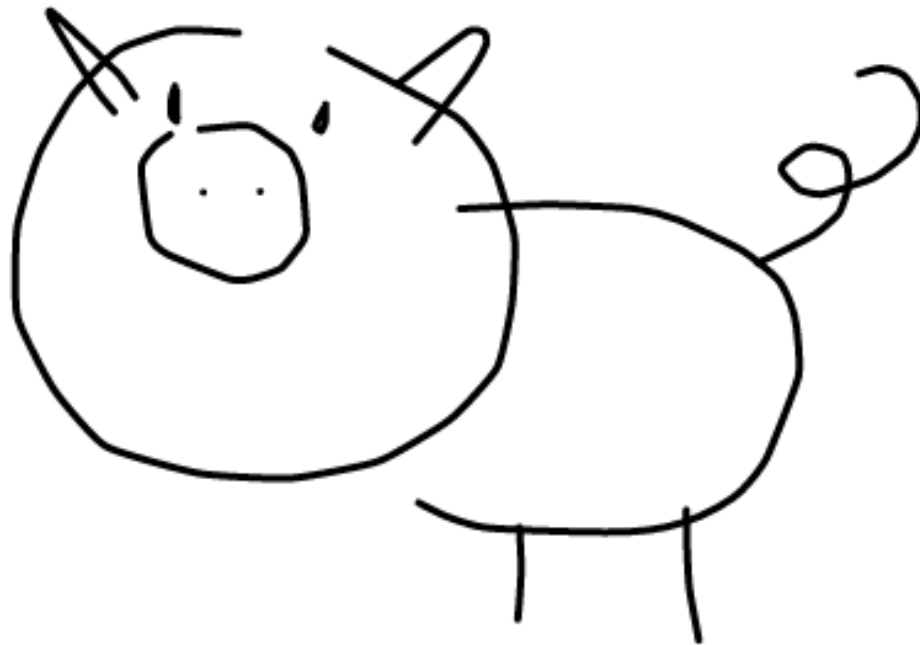
```
    #print(i)
```

Motion Prediction

Given an initial still frame,



Drawing by Machine



https://magenta.tensorflow.org/assets/sketch_rnn_demo/index.html

AI Poet



Music Composition

A.I. Duet

Trade melodies with a neural network.



PLAY

Music by Machine



Music by Machine

**Baidu is putting the *art*
in *artificial intelligence***



Music (Beatles' style)

GEEK
TECH

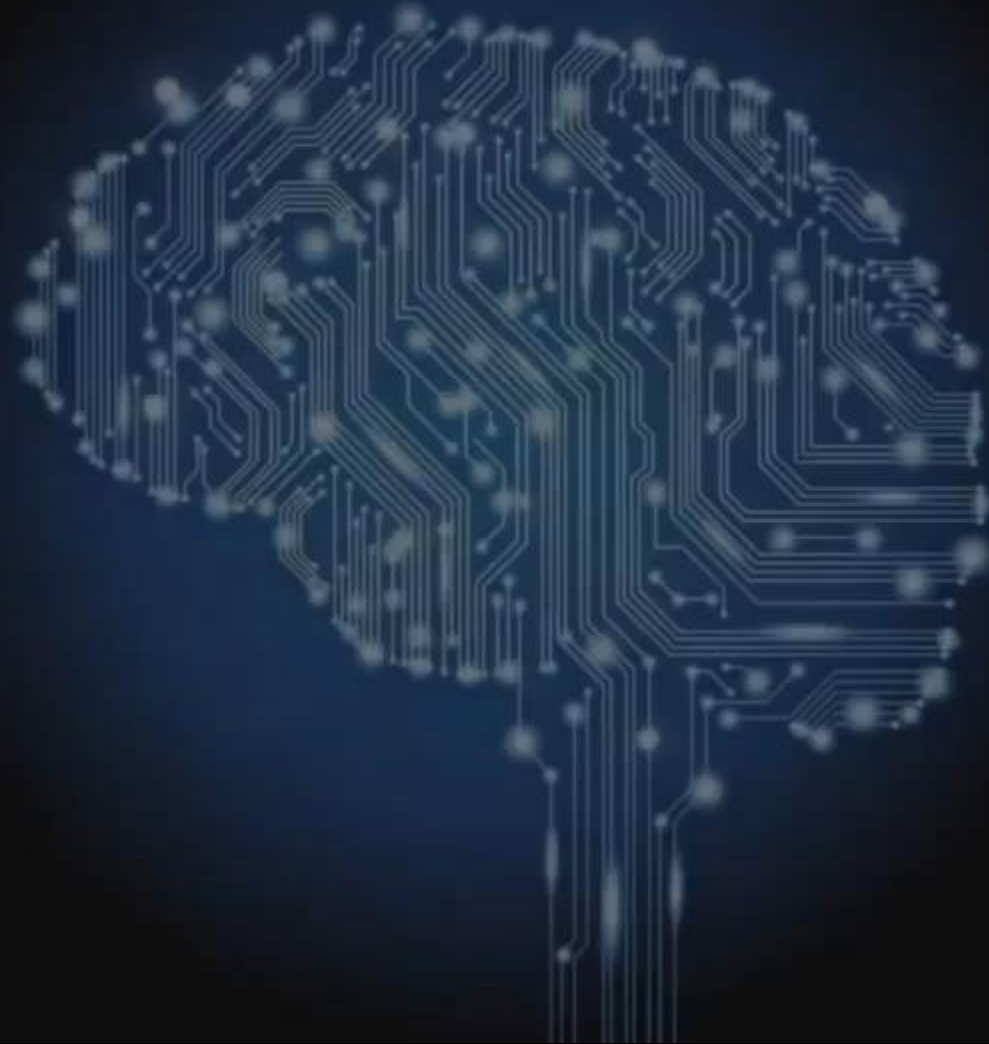


Song of birds



<https://aiexperiments.withgoogle.com/bird-sounds/view/>

Human an AI



Which ML Frameworks?

Caffe

No need to write code!

1. Convert data (run a script)
2. Define net (edit prototxt)
3. Define solver (edit prototxt)
4. Train (with pretrained weights)

TensorFlow

```
1 import tensorflow as tf
2 import numpy as np
3
4 N, D, H, C = 64, 1000, 100, 10
5
6 x = tf.placeholder(tf.float32, shape=[None, D])
7 y = tf.placeholder(tf.float32, shape=[None, C])
8
9 w1 = tf.Variable(1e-3 * np.random.randn(D, H).astype(np.float32))
10 w2 = tf.Variable(1e-3 * np.random.randn(H, C).astype(np.float32))
11
12 a = tf.matmul(x, w1)
13 a_relu = tf.nn.relu(a)
14 scores = tf.matmul(a_relu, w2)
15 probs = tf.nn.softmax(scores)
16 loss = -tf.reduce_sum(y * tf.log(probs))
17
18 learning_rate = 1e-2
19 train_step = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)
20
21 xx = np.random.randn(N, D).astype(np.float32)
22 yy = np.zeros((N, C)).astype(np.float32)
23 yy[np.arange(N), np.random.randint(C, size=N)] = 1
24
25 with tf.Session() as sess:
26     sess.run(tf.initialize_all_variables())
27
28     for t in xrange(100):
29         _, loss_value = sess.run([train_step, loss],
30                                 feed_dict={x: xx, y: yy})
31     print loss_value
32
```

Torch

```
1 require 'torch'
2 require 'nn'
3 require 'optim'
4
5 -- Batch size, input dim, hidden dim, num classes
6 local N, D, H, C = 100, 1000, 100, 10
7
8 -- Build a one-layer ReLU network
9 local net = nn.Sequential()
10 net:add(nn.Linear(D, H))
11 net:add(nn.ReLU())
12 net:add(nn.Linear(H, C))
13
14 -- Collect all weights and gradients in a single Tensor
15 local weights, grad_weights = net:getParameters()
16
17 -- Loss functions are called "criteria"
18 local crit = nn.CrossEntropyCriterion() -- Softmax loss
19
20 -- Callback to interface with optim methods
21 local function f(w)
22     assert(w == weights)
23
24     -- Generate some random input data
25     local x = torch.randn(N, D)
26     local y = torch.Tensor(N):random(C)
27
28     -- Forward pass: Compute scores and loss
29     local scores = net:forward(x)
30     local loss = crit:forward(scores, y)
31
32     -- Backward pass: compute gradients
33     grad_weights:zero()
34     local dscores = crit:backward(scores, y)
35     local dx = net:backward(x, dscores)
36
37     return loss, grad_weights
38 end
39
40 -- Make a step using Adam
41 local state = {learningRate=1e-3}
42 optim.adam(f, weights, state)
43
```

2,3

Theano

```
import theano
import theano.tensor as T

# Batch size, input dim, hidden dim, num classes
N, D, H, C = 64, 1000, 100, 10

x = T.matrix('x')
y = T.vector('y', dtype='int64')
w1 = T.matrix('w1')
w2 = T.matrix('w2')

# Forward pass: Compute scores
a = x.dot(w1)
a_relu = T.nnet.relu(a)
scores = a_relu.dot(w2)

# Forward pass: compute softmax loss
probs = T.nnet.softmax(scores)
loss = T.nnet.categorical_crossentropy(probs, y).mean()

# Backward pass: compute gradients
dw1, dw2 = T.grad(loss, [w1, w2])

f = theano.function(
    inputs=[x, y, w1, w2],
    outputs=[loss, scores, dw1, dw2],
)
```

Mxnet
chainer
Nervana's Neon
Microsoft's CNTK

Lasagne

```
1 import numpy as np
2 import theano
3 import theano.tensor as T
4 import lasagne
5
6 N, D, H, C = 64, 1000, 100, 10
7
8 x = T.matrix('x')
9 y = T.vector('y', dtype='int64')
10
11 relu = lasagne.nonlinearities.rectify
12 softmax = lasagne.nonlinearities.softmax
13 net = lasagne.layers.InputLayer(shape=(None, D), input_var=x)
14 net = lasagne.layers.DenseLayer(net, H, nonlinearity=relu)
15 net = lasagne.layers.DenseLayer(net, C, nonlinearity=softmax)
16
17 probs = lasagne.layers.get_output(net)
18 loss = lasagne.objectives.categorical_crossentropy(probs, y).mean()
19
20 params = lasagne.layers.get_all_params(net, trainable=True)
21 updates = lasagne.updates.nesterov_momentum(loss, params,
22                                             learning_rate=1e-2, momentum=0.9)
23
24 train_fn = theano.function([x, y], loss, updates=updates)
25
26 xx = np.random.randn(N, D)
27 yy = np.random.randint(C, size=N).astype(np.int64)
28
29 for t in xrange(100):
30     loss_val = train_fn(xx, yy)
31     print loss_val
32
```

Keras

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD
from keras.utils import np_utils

D, H, C = 1000, 100, 10

model = Sequential()
model.add(Dense(input_dim=D, output_dim=H))
model.add(Activation('relu'))
model.add(Dense(input_dim=H, output_dim=C))
model.add(Activation('softmax'))

sgd = SGD(lr=1e-3, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)

N, N_batch = 1000, 32
X = np.random.randn(N, D)
y = np.random.randint(C, size=N)
y = np_utils.to_categorical(y)

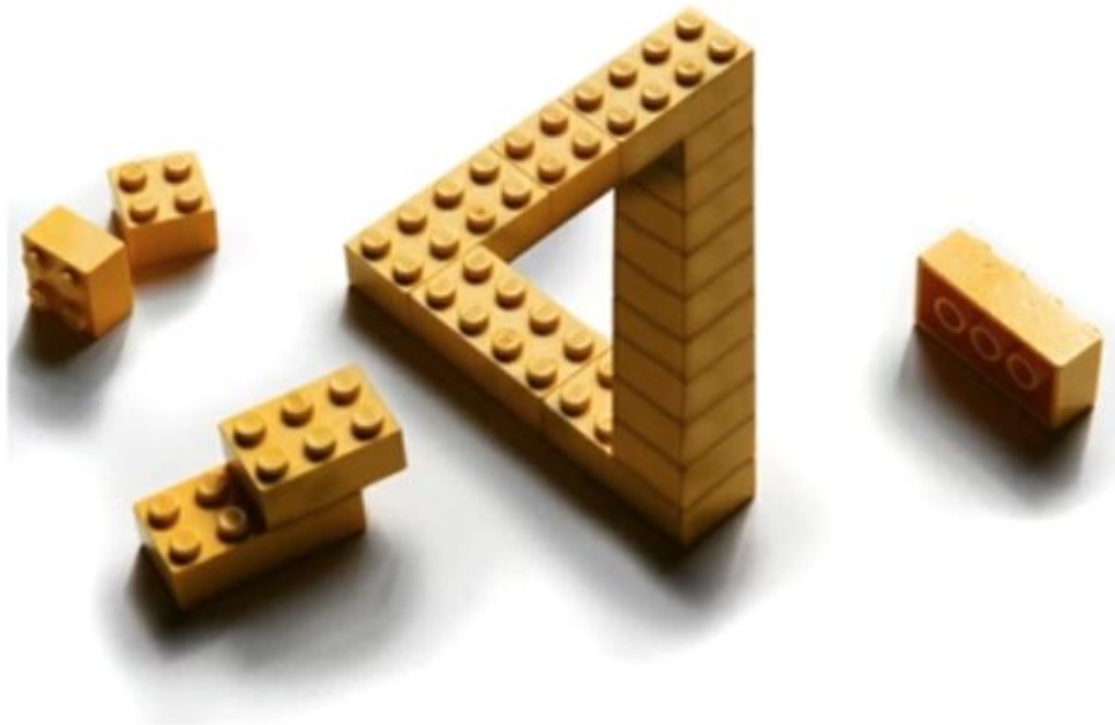
model.fit(X, y, nb_epoch=5, batch_size=N_batch, verbose=2)
```

1

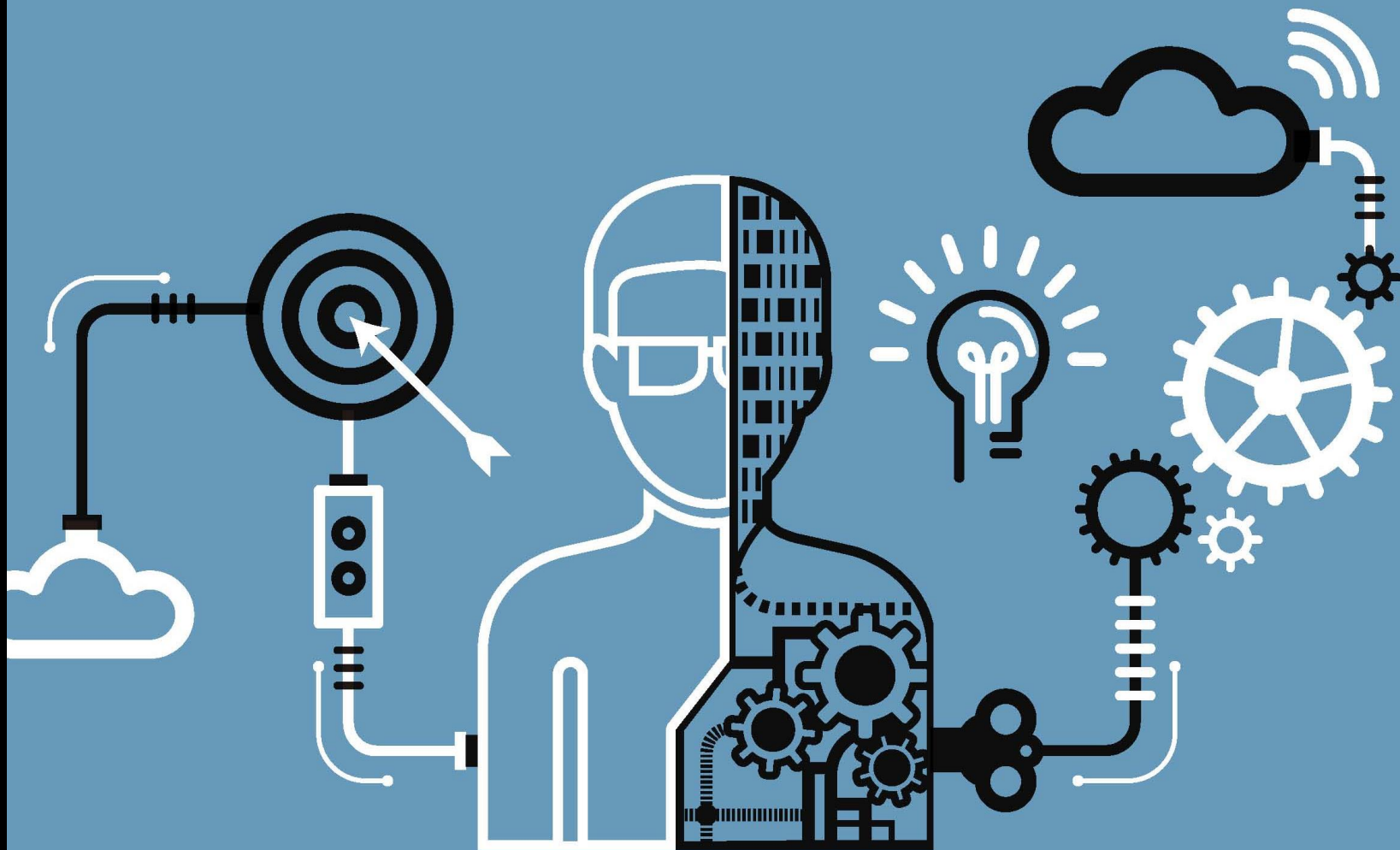
Do it now.

- Python
- TensorFlow / Keras
- GitHub

'The only limit is your imagination'



<http://itchyi.squarespace.com/thelatest/2012/5/17/the-only-limit-is-your-imagination.html>



MACHINE LEARNING
CAMP JEJU 2017





Thank you!

ycb@jejunu.ac.kr