Lab 11
Tensorflow Basics

Song Chaoyang

Assistant Professor

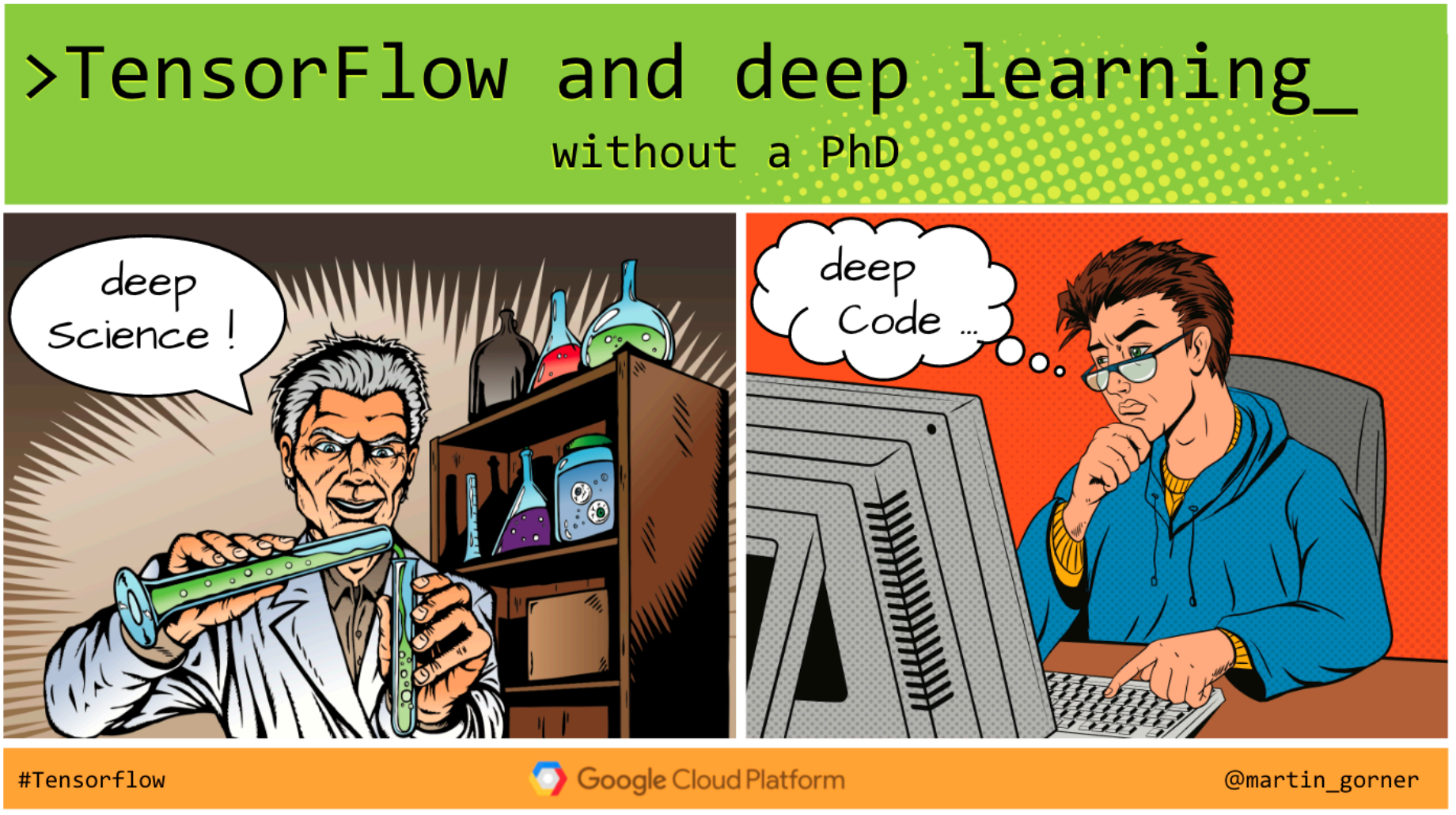
Department of Mechanical and Energy Engineering

songcy@sustech.edu.cn

[Slides adapted from Tensorflow and deep learning without a PhD series by Martin Gorner @ Google.]

Let's Try

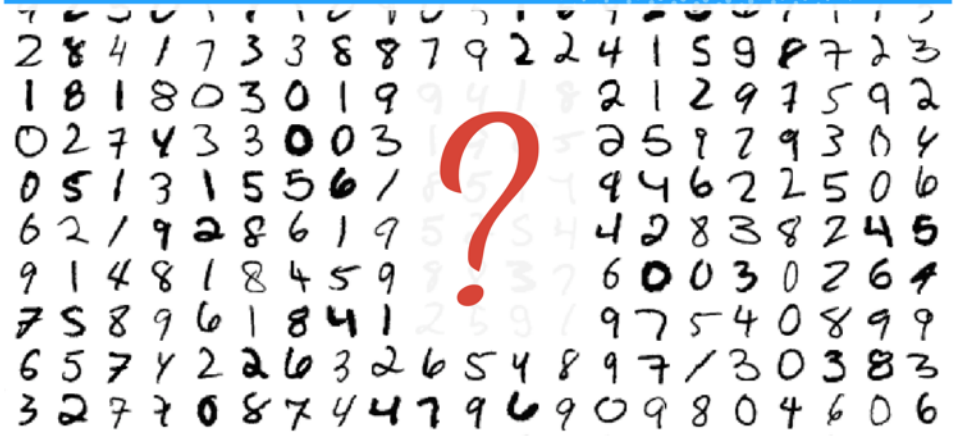
https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd



A 1-layer NN

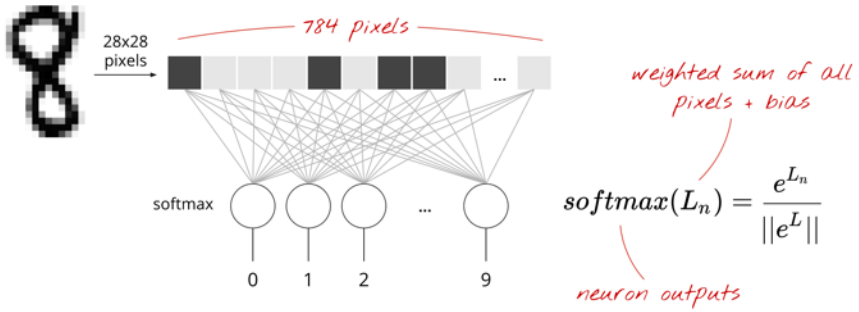
```
1 $ git clone https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd.git
2 $ cd tensorflow-without-a-phd/tensorflow-mnist-tutorial
```

Hello World: handwritten digits classification - MNIST



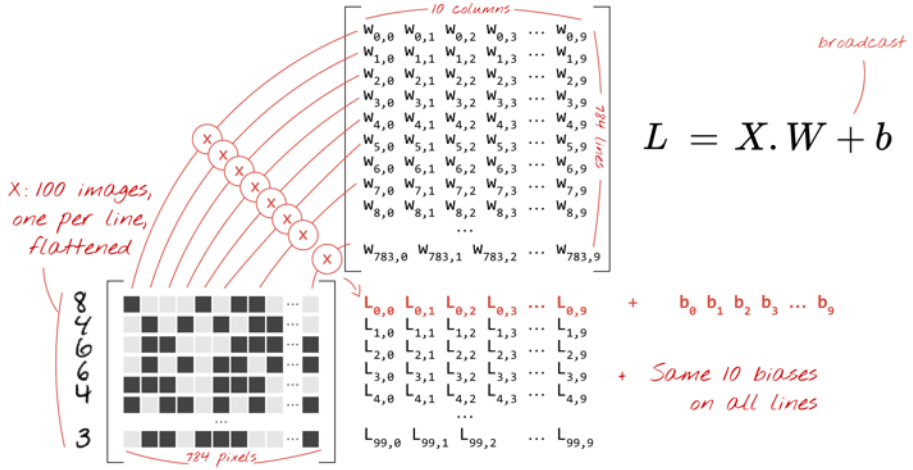
MNIST = Mixed National Institute of Standards and Technology - Download the dataset at http://yann.lecun.com/exdb/mnist/

Very simple model: softmax classification



DEVOXX @martin_gorner Google Cloud Platform

In matrix notation, 100 images at a time

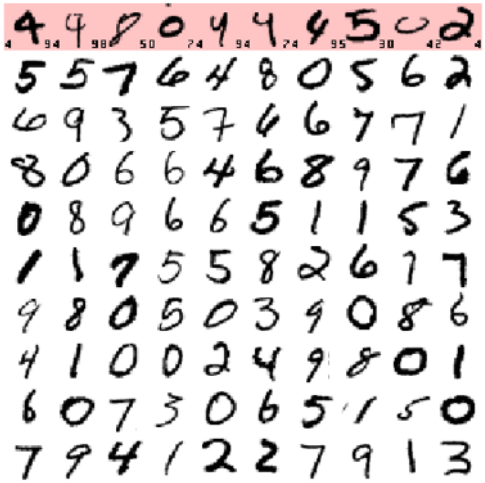


AncoraSIR.com

Training digits
=>
updates to weights and biases
=>
better recognition (loop)

Handwritten digits in the MNIST dataset are 28x28 pixel greyscale images.

Training digits



Gradient Descent

Training digits and labels =>
Loss function =>
Gradient (partial derivatives) =>
Steepest descent =>
Update weights and biases =>
Repeat with next mini-batch of training images and labels

Softmax, on a batch of images

Predictions Y[100, 10] — Images X[100, 784] — Weights W[784,10] — Biases b[10]

$$Y = softmax(X.W + b)$$

applied line by line — matrix multiply — broadcast on all lines

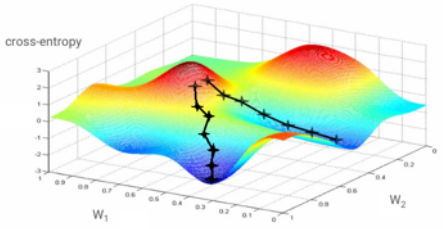
tensor shapes in [ ]

Now in TensorFlow (Python)

tensor shapes: X[100, 784] W[748,10] b[10]

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

matrix multiply — broadcast on all lines



Partial derivative of cross-entropy gives the "gradient", computed for a given image, label and present value of weights and biases

weighted sum of all pixels + bias

$$softmax(L_n) = \frac{e^{L_n}}{||e^{L}||}$$

neuron outputs

AncoraSIR.com

DEVOXX @martin_gorner Google Cloud Platform

Success ?

0 1 2 3 4 5 6 7 8 9

0 0 0 0 0 0 1 0 0 0

actual probabilities, "one-hot" encoded

Cross entropy: $-\sum Y'_i . log(Y_i)$

computed probabilities

this is a "6"

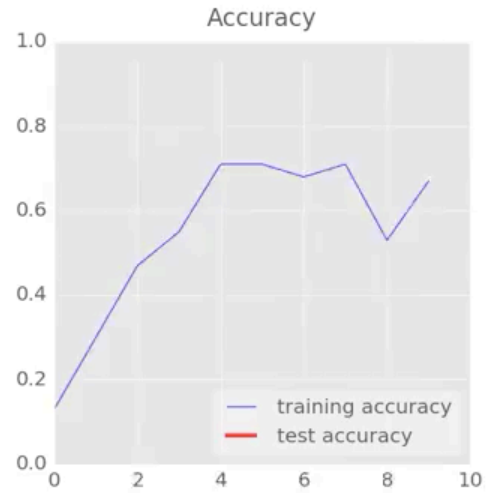
0.1 0.2 0.1 0.3 0.2 0.1 0.9 0.2 0.1 0.1

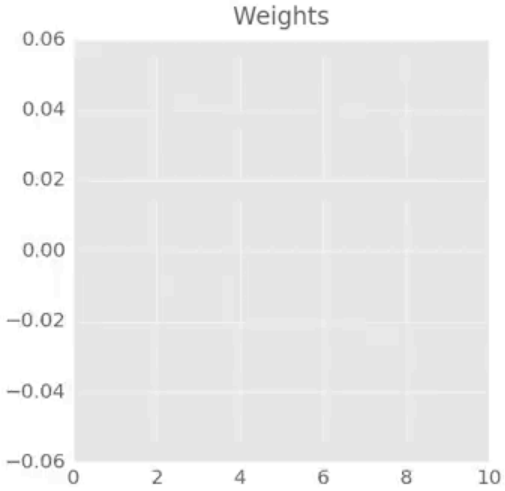
0 1 2 3 4 5 6 7 8 9

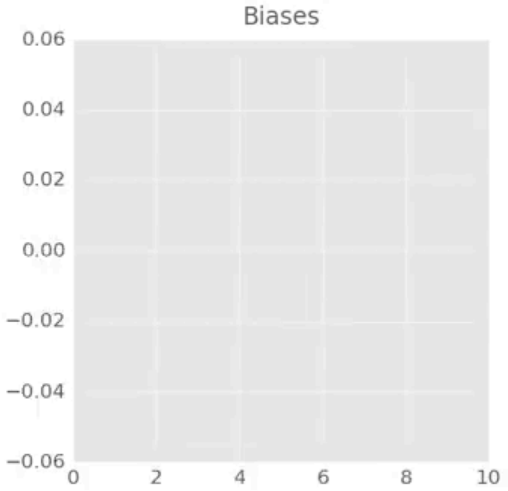
DEVOXX @martin_gorner Google Cloud Platform

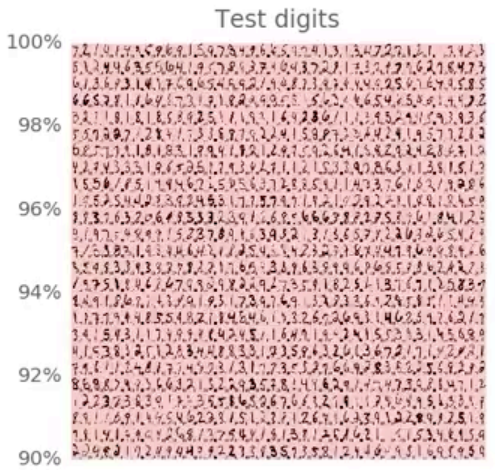
Southern University of Science and Technology

python mnist_1.0_softmax.py => 92%









AncoraSIR.com

Lab: let's jump into the code

TensorFlow - full python code

```python
import tensorflow as tf

X = tf.placeholder(tf.float32, [None, 28, 28, 1])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
init = tf.initialize_all_variables()

# model
Y=tf.nn.softmax(tf.matmul(tf.reshape(X,[-1, 784]), W) + b)

# placeholder for correct answers
Y_ = tf.placeholder(tf.float32, [None, 10])

# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))

# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y,1), tf.argmax(Y_,1))
accuracy = tf.reduce_mean(tf.cast(is_correct,tf.float32))
```

initialisation

this will become the batch size, 100

28 x 28 grayscale images

model

flattening images

Training = computing variables W and b

"one-hot" encoded

success metrics

"one-hot" decoding

```python
optimizer = tf.train.GradientDescentOptimizer(0.003)
train_step = optimizer.minimize(cross_entropy)

sess = tf.Session()
sess.run(init)

for i in range(10000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}

    # train
    sess.run(train_step, feed_dict=train_data)

    # success ? add code to print it
    a,c = sess.run([accuracy, cross_entropy], feed=train_data)

    # success on test data ?
    test_data={X:mnist.test.images, Y_:mnist.test.labels}
    a,c = sess.run([accuracy, cross_entropy], feed=test_data)
```

training step

learning rate

loss function

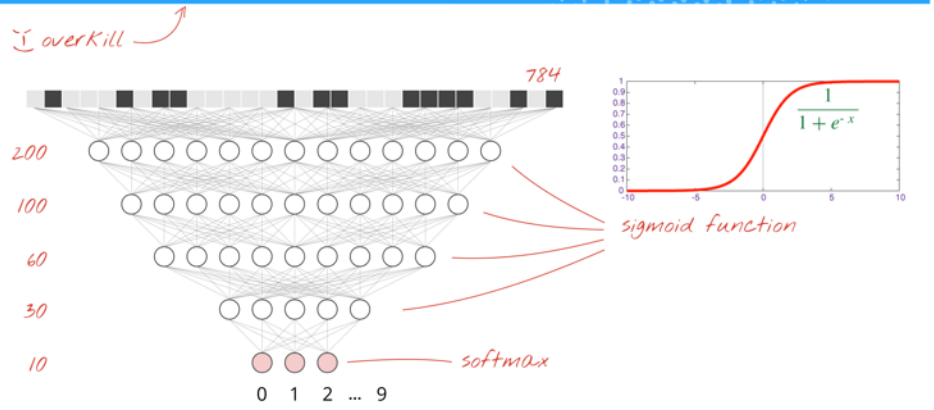
running a Tensorflow computation, feeding placeholders

Run

Tip: do this every 100 iterations

Lab: adding layers

Let's try 5 fully-connected layers !



TensorFlow - initialisation

```
K = 200
L = 100
M = 60
N = 30

W1 = tf.Variable(tf.truncated_normal([28*28, K] ,stddev=0.1))
B1 = tf.Variable(tf.zeros([K]))

W2 = tf.Variable(tf.truncated_normal([K, L], stddev=0.1))
B2 = tf.Variable(tf.zeros([L]))

W3 = tf.Variable(tf.truncated_normal([L, M], stddev=0.1))
B3 = tf.Variable(tf.zeros([M]))
W4 = tf.Variable(tf.truncated_normal([M, N], stddev=0.1))
B4 = tf.Variable(tf.zeros([N]))
W5 = tf.Variable(tf.truncated_normal([N, 10], stddev=0.1))
B5 = tf.Variable(tf.zeros([10]))
```

weights initialised with random values

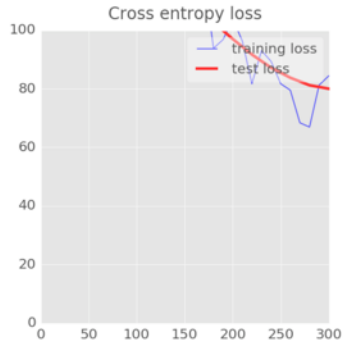
TensorFlow - the model

weights and biases

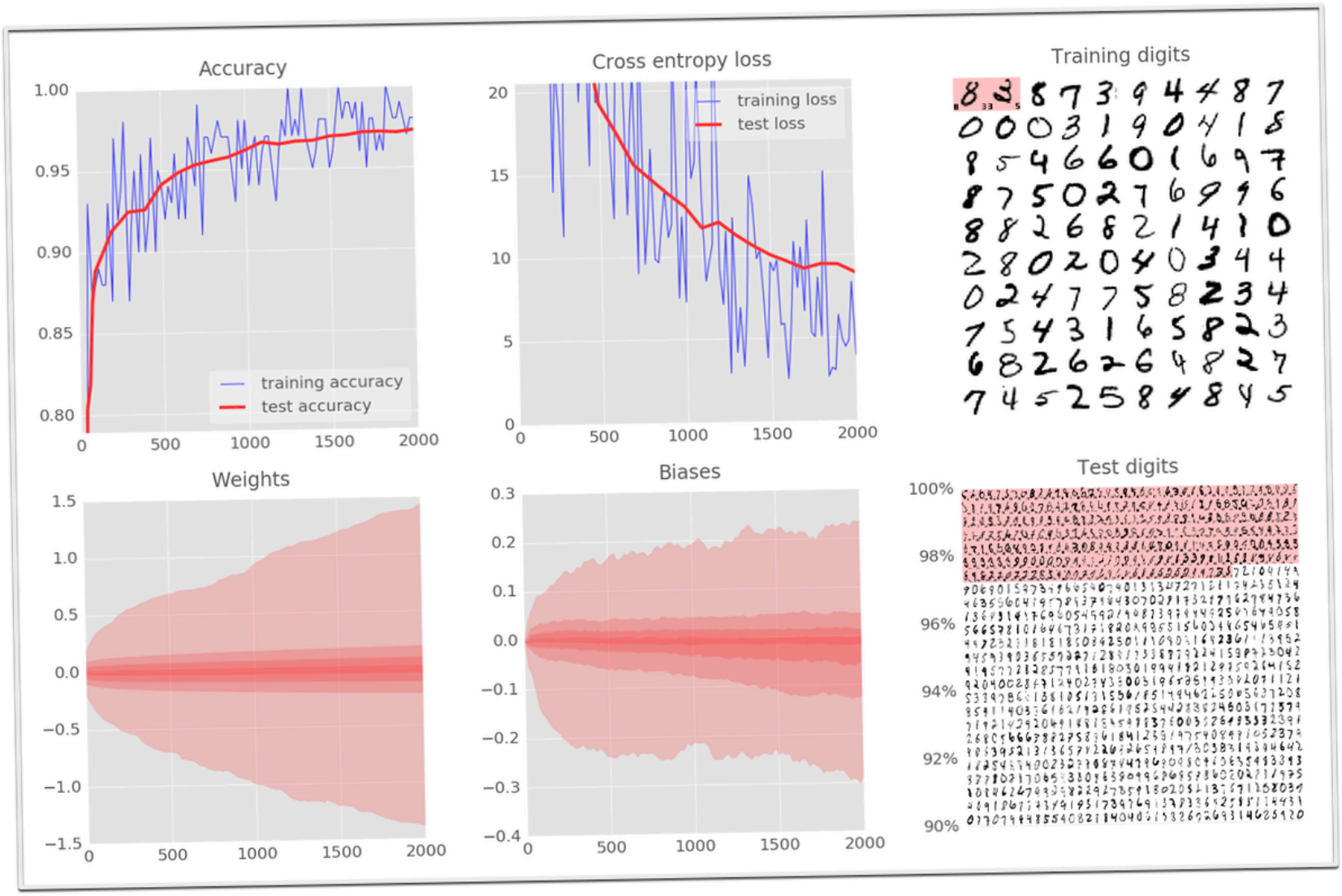
```
X = tf.reshape(X, [-1, 28*28])

Y1 = tf.nn.sigmoid(tf.matmul(X, W1) + B1)
Y2 = tf.nn.sigmoid(tf.matmul(Y1, W2) + B2)
Y3 = tf.nn.sigmoid(tf.matmul(Y2, W3) + B3)
Y4 = tf.nn.sigmoid(tf.matmul(Y3, W4) + B4)
Y = tf.nn.softmax(tf.matmul(Y4, W5) + B5)
```


Demo - slow start ?



python mnist_2.0_five_layers_sigmoid.py => 97%

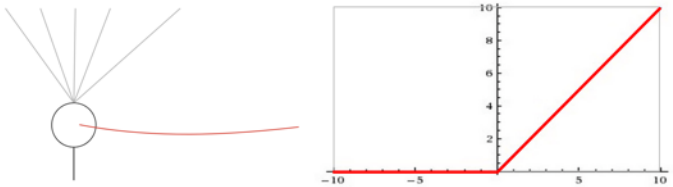


97%

Lab: special care for deep networks

RELU

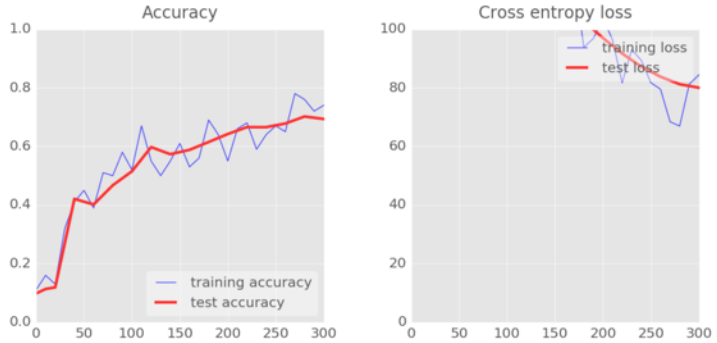
RELU = Rectified Linear Unit



```
Y = tf.nn.relu(tf.matmul(X, W) + b)
```

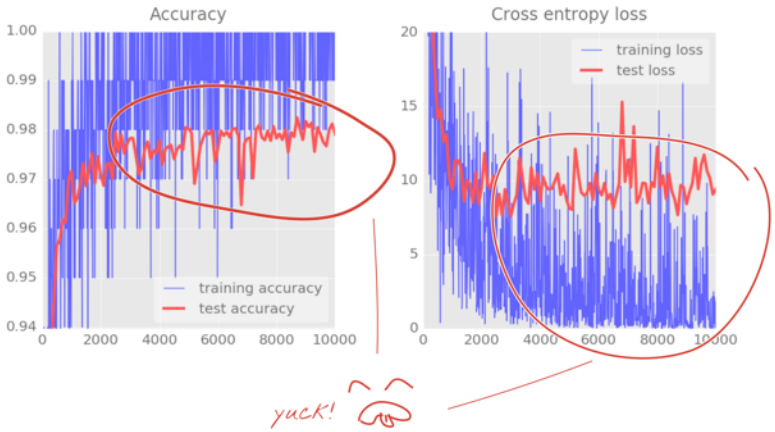
DEVOXX @martin_gorner Google Cloud Platform

RELU



DEVOXX @martin_gorner Google Cloud Platform

Demo - noisy accuracy curve ?



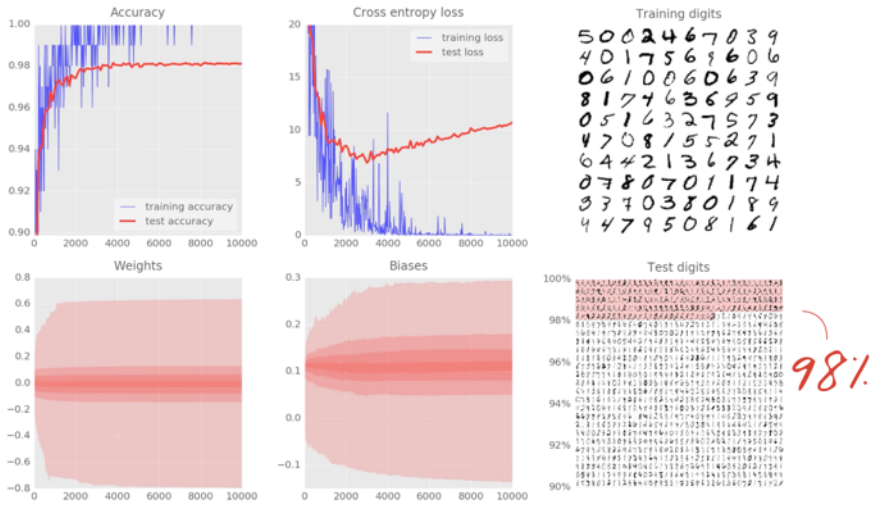
AncoraSIR.com

RELU

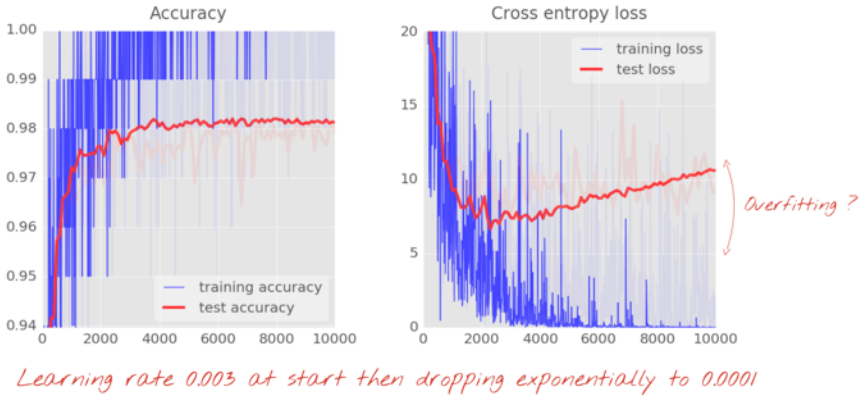
DEVOXX @martin_gorner Google Cloud Platform

Southern University of Science and Technology

python mnist_2.1_five_layers_relu_lrdecay.py => 98%

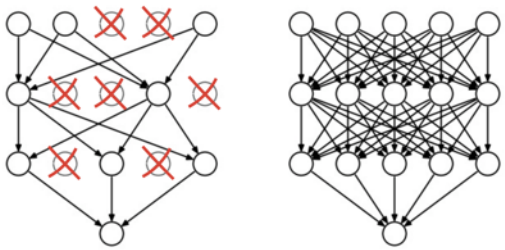


Learning rate decay



Learning rate 0.003 at start then dropping exponentially to 0.0001

Dropout




```
pkeep =
tf.placeholder(tf.float32)

Yf = tf.nn.relu(tf.matmul(X, W) + B)
Y  = tf.nn.dropout(Yf, pkeep)
```


All the party tricks

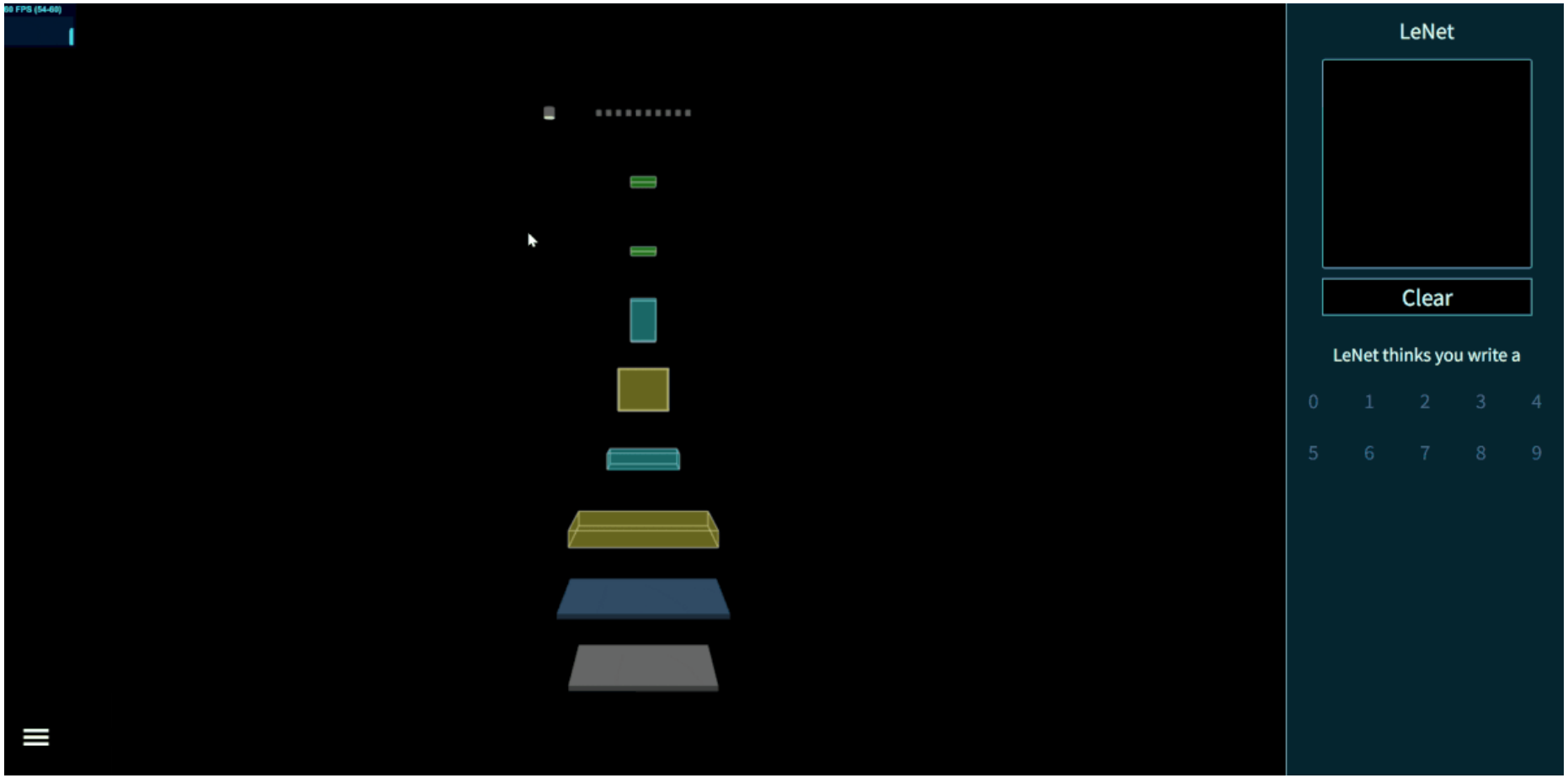
RELU, decaying learning rate 0.003 -> 0.0001 and dropout 0.75

mnist_2.2_five_layers_relu_lrdecay_dropout.py

AncoraSIR.com

TensorSpace.js

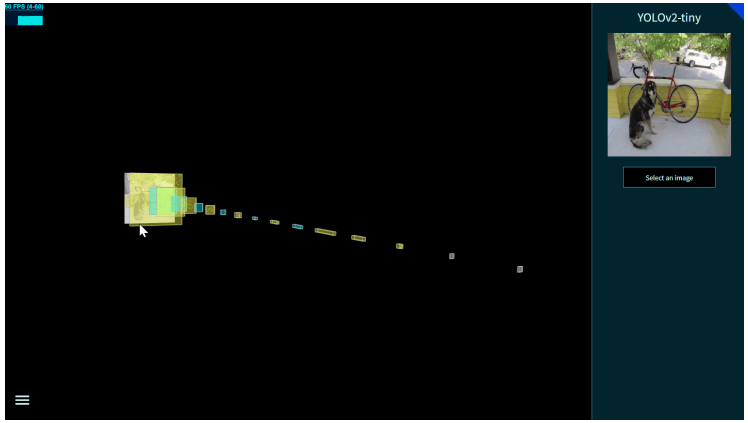
A neural network 3D visualization framework

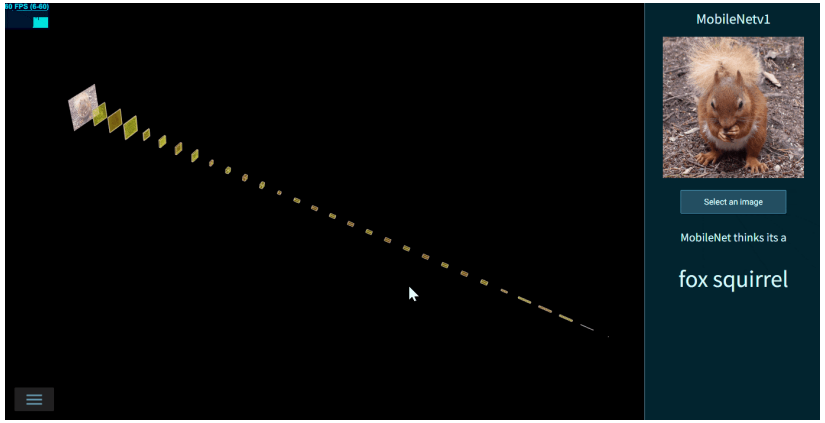


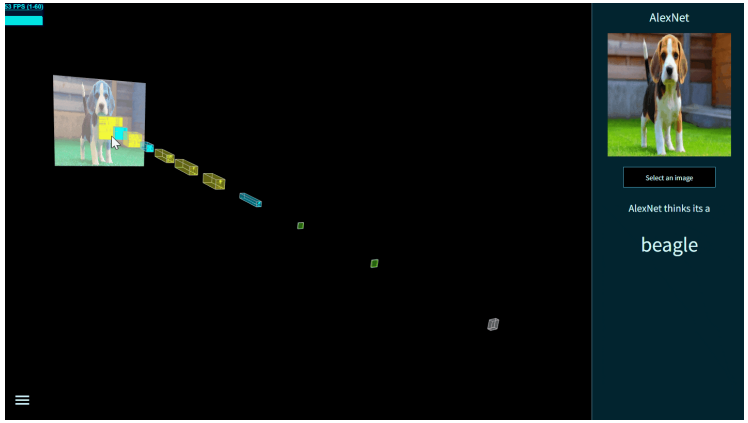
AncoraSIR.com

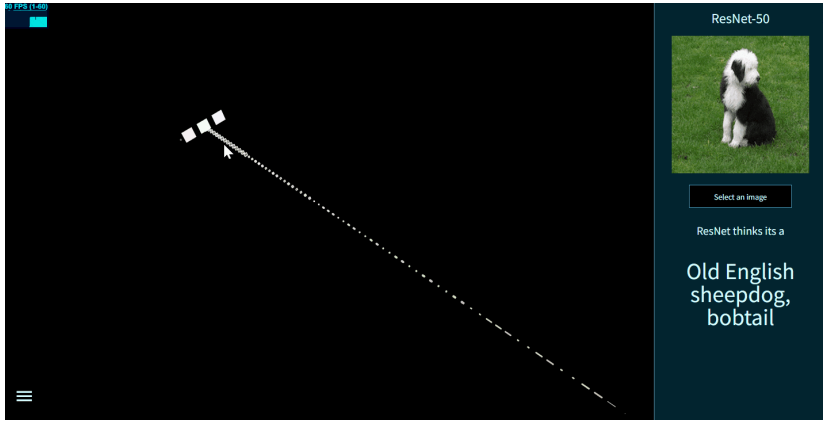
https://github.com/tensorspace-team/tensorspace

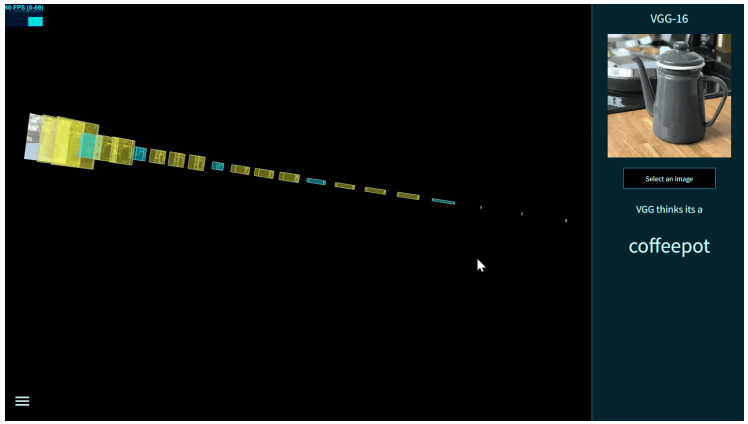
SUSTech
Southern University of Science and Technology

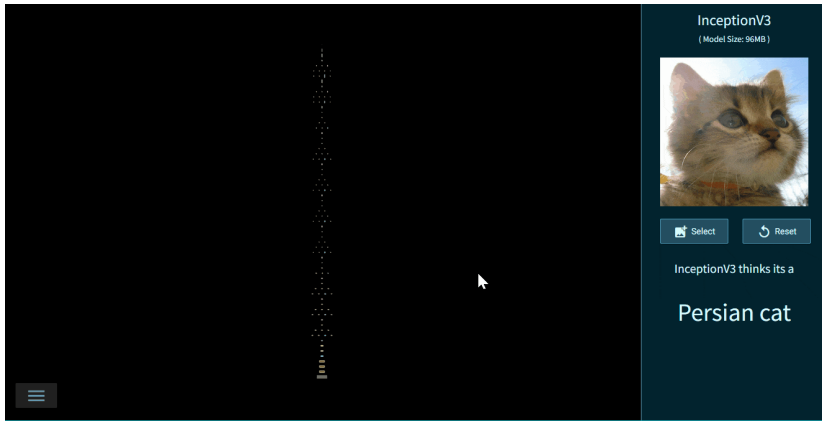












[Live demo]

AncoraSIR.com

SUSTech
Southern University of Science and Technology

Thank you!

Prof. Song Chaoyang

- Dr. Wan Fang (sophie.fwan@hotmail.com)

AncoraSIR.com

SUSTech
Southern University of Science and Technology