OSC源创会 年终盛典 2016

Prisma核心算法理论讲解分析

&&

TensorFlow复现

解读CVPR2016 oral paper：

Image Style Transfer Using Convolutional Neural Networks

AG Group 万元芳

OSC源创会 年终盛典 2016

Image Style Transfer Using Convolutional Neural Networks

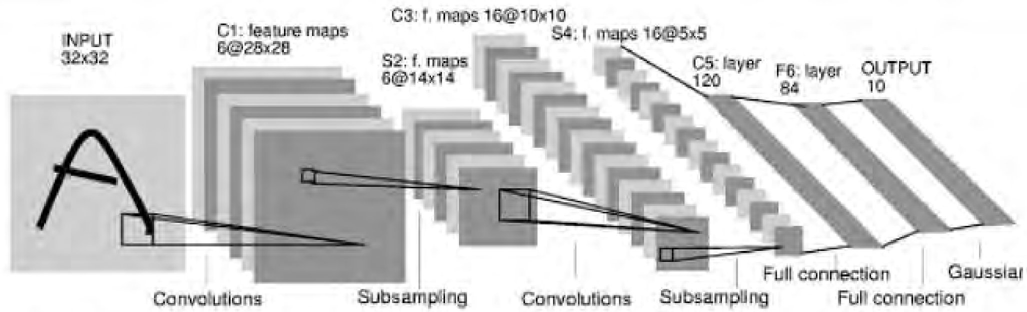
Leon A. Gatys
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Graduate School of Neural Information Processing, University of Tübingen, Germany
leon.gatys@bethgelab.org

Alexander S. Ecker
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany
Baylor College of Medicine, Houston, TX, USA

Matthias Bethge
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany

1998

LeCun et al.



# of transistors

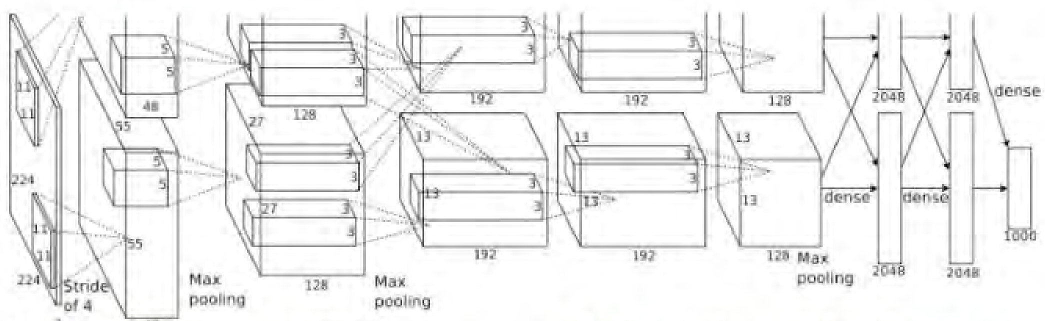
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

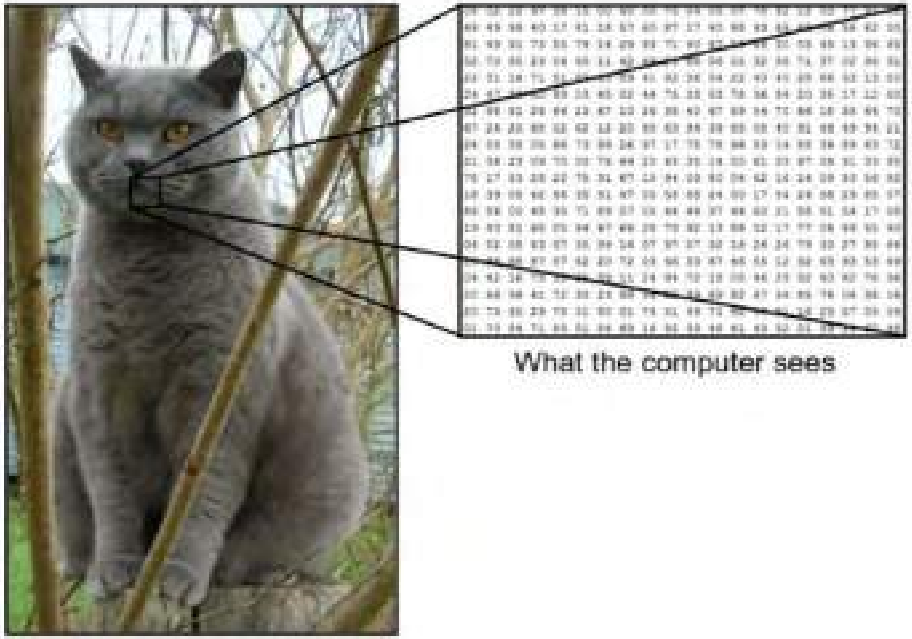
AG

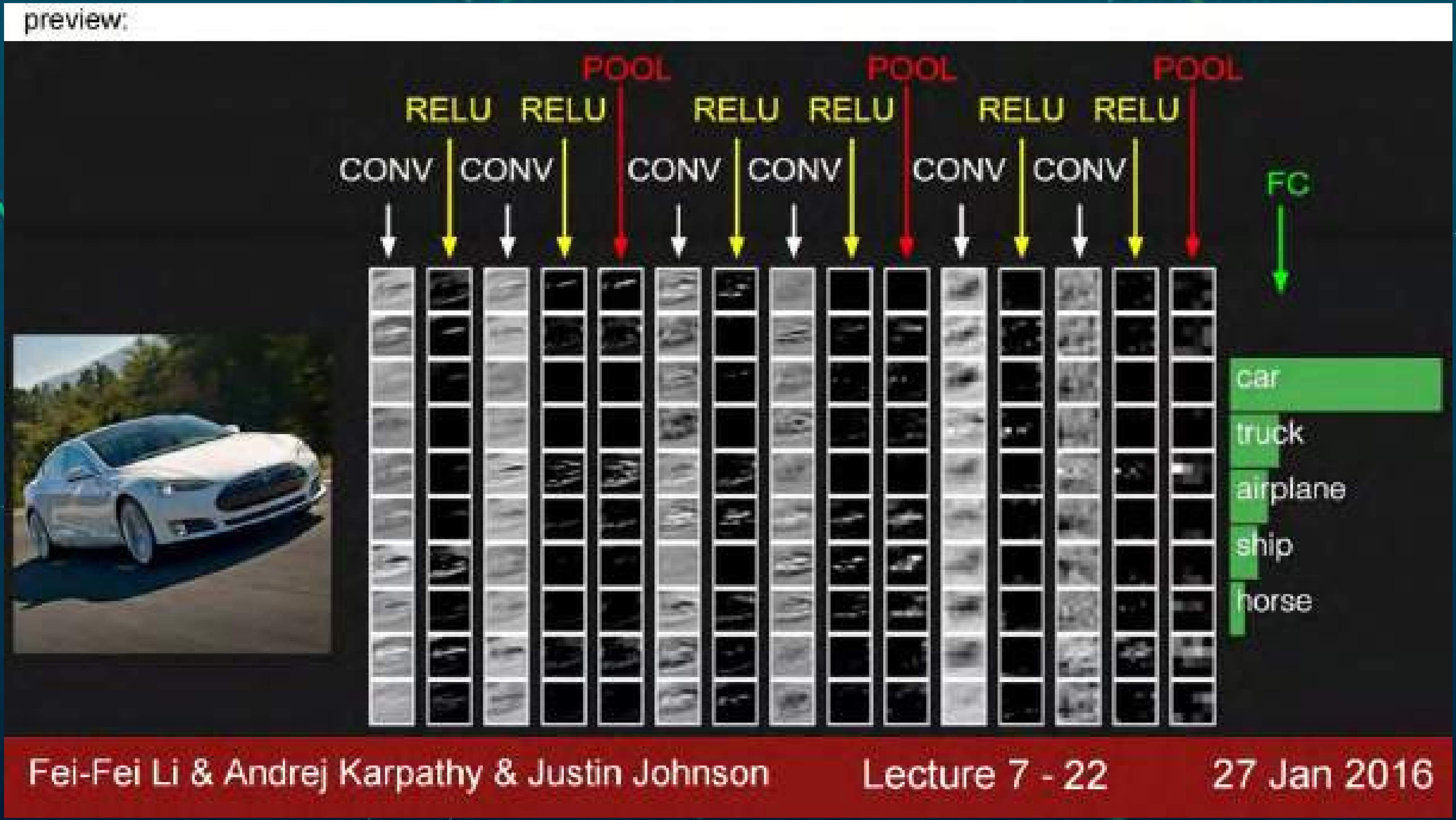
The problem:
semantic gap

Images are represented as 3D arrays of numbers, with integers between [0, 255].

E.g.
300 x 100 x 3

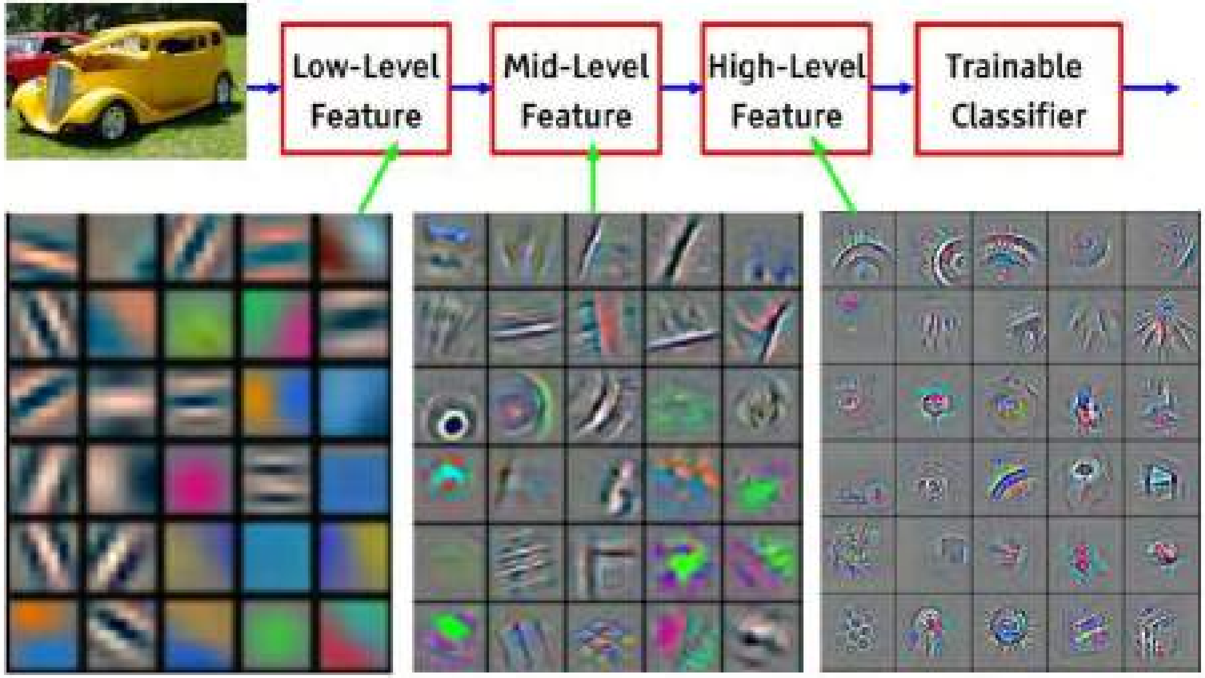
(3 for 3 color channels RGB)




preview:
POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse
Fei-Fei Li & Andrej Karpathy & Justin Johnson
Lecture 7 - 22
27 Jan 2016

Preview

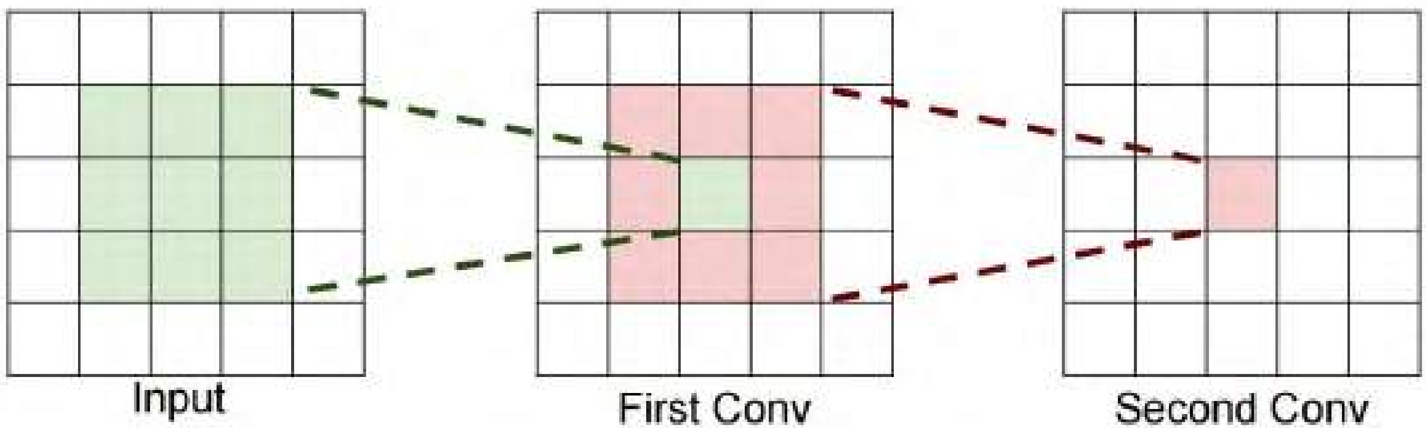
[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

The power of small filters

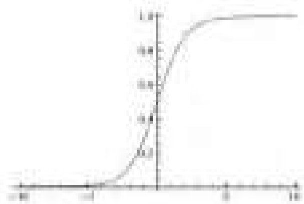
Suppose we stack two 3x3 conv layers (stride 1)
Each neuron sees 3x3 region of previous activation map

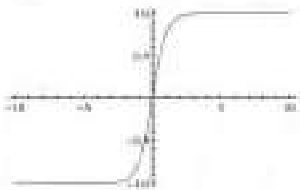


Activation Functions

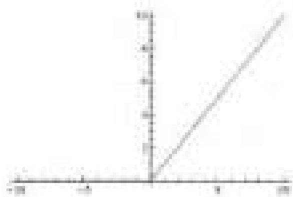
Sigmoid

$$\sigma(x) = 1/(1+e^{-x})$$



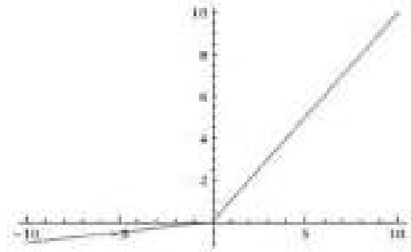


tanh tanh(x)



ReLU max(0,x)

Leaky ReLU
max(0.1x, x)

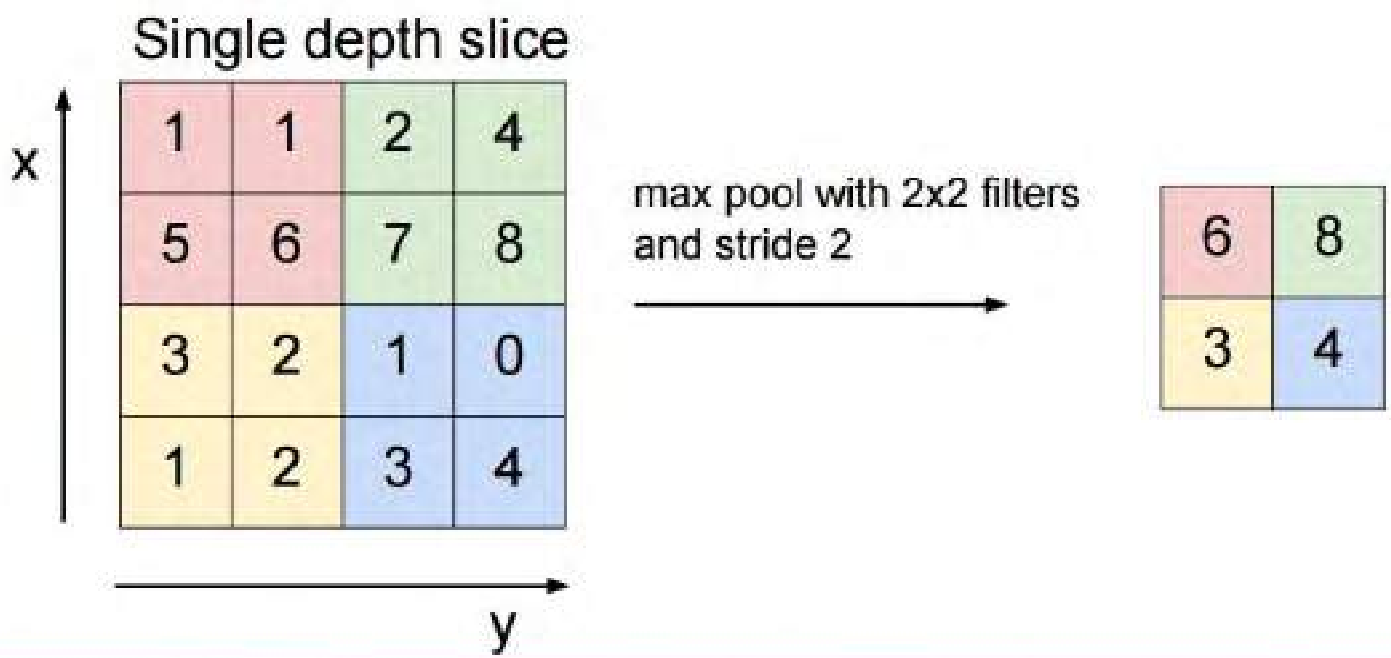


Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

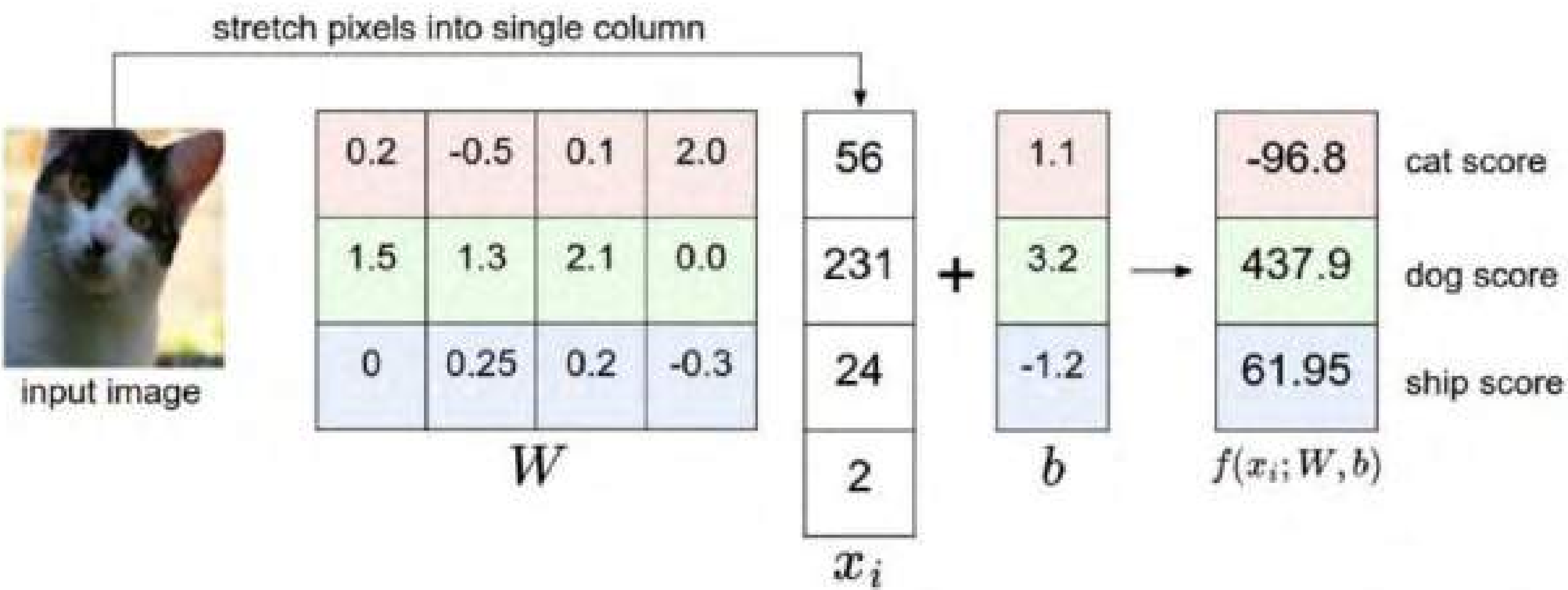
ELU

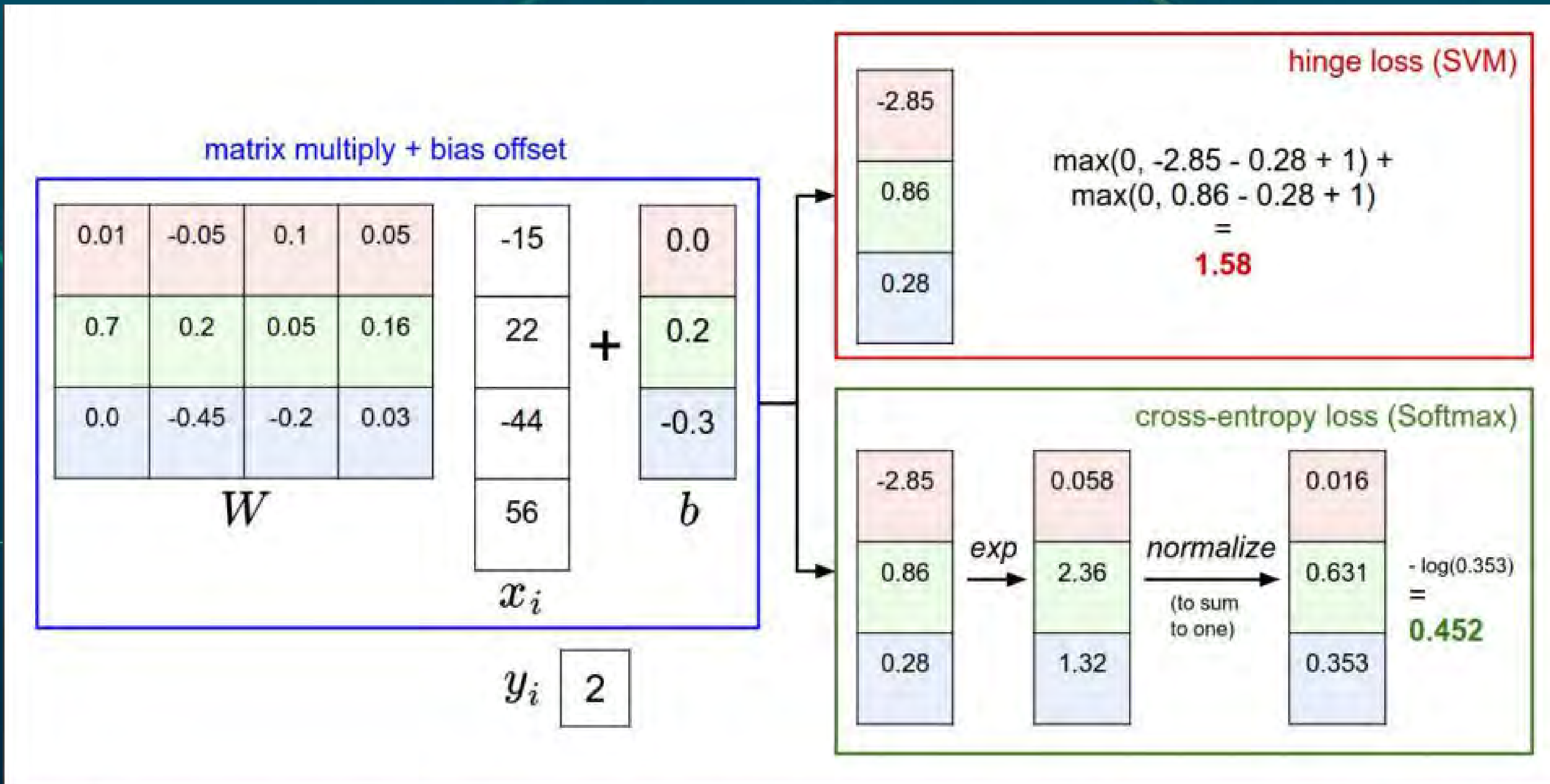
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$

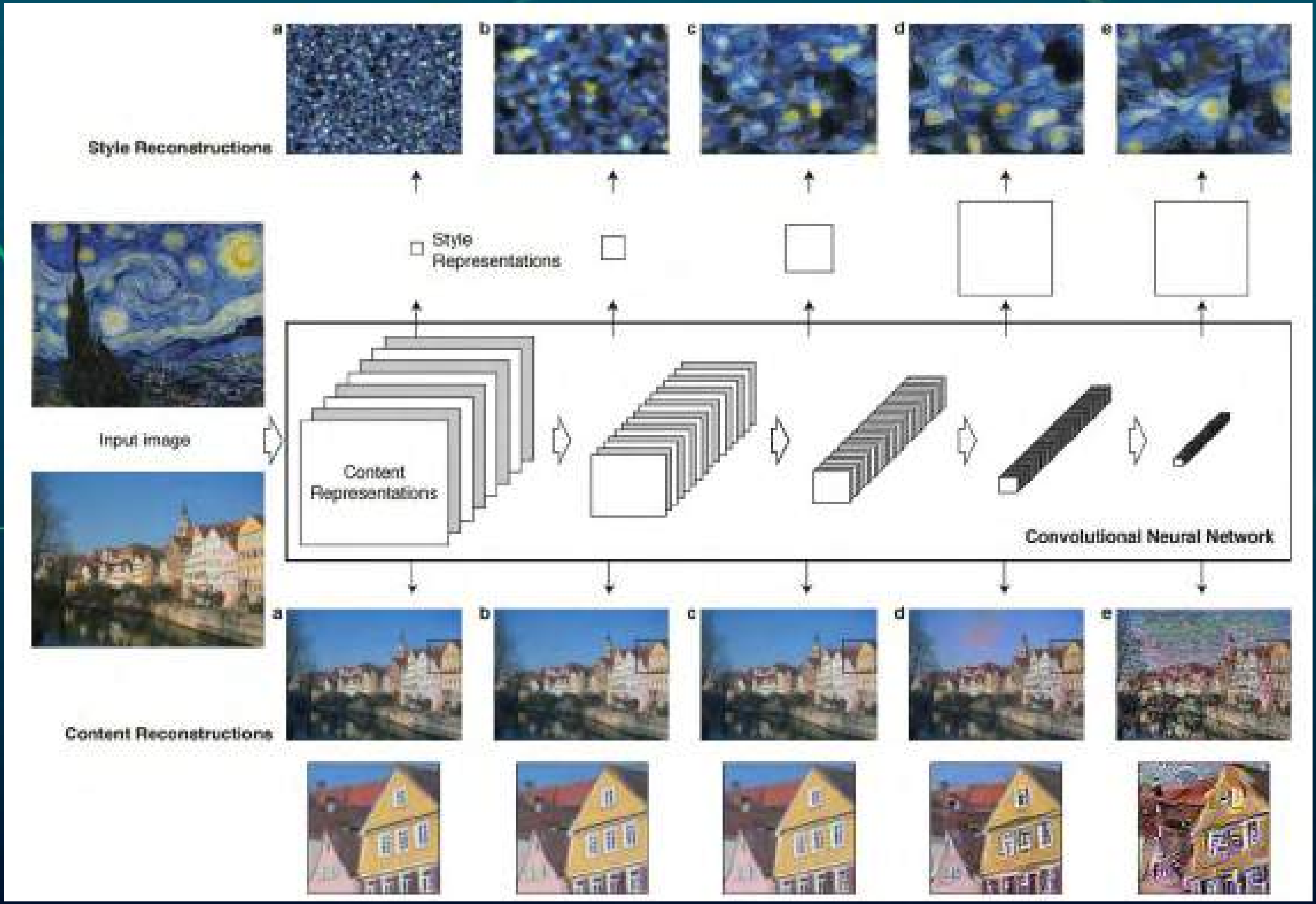
MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters and stride 2
6 8
3 4

Together, we've defined Score Functions...




matrix multiply + bias offset
0.01 -0.05 0.1 0.05
0.7 0.2 0.05 0.16
0.0 -0.45 -0.2 0.03
W
-15
22
-44
56
x_i
+
0.0
0.2
-0.3
b
y_i 2
hinge loss (SVM)
-2.85
0.86
0.28
max(0, -2.85 - 0.28 + 1) +
max(0, 0.86 - 0.28 + 1)
=
1.58
cross-entropy loss (Softmax)
-2.85
0.86
0.28
exp
0.058
2.36
1.32
normalize
(to sum to one)
0.016
0.631
0.353
- log(0.353)
=
0.452



Style Reconstructions
a
b
c
d
e
Style
Representations
Input image
Content
Representations
Convolutional Neural Network
Content Reconstructions

OSC源创会 年终盛典 2016

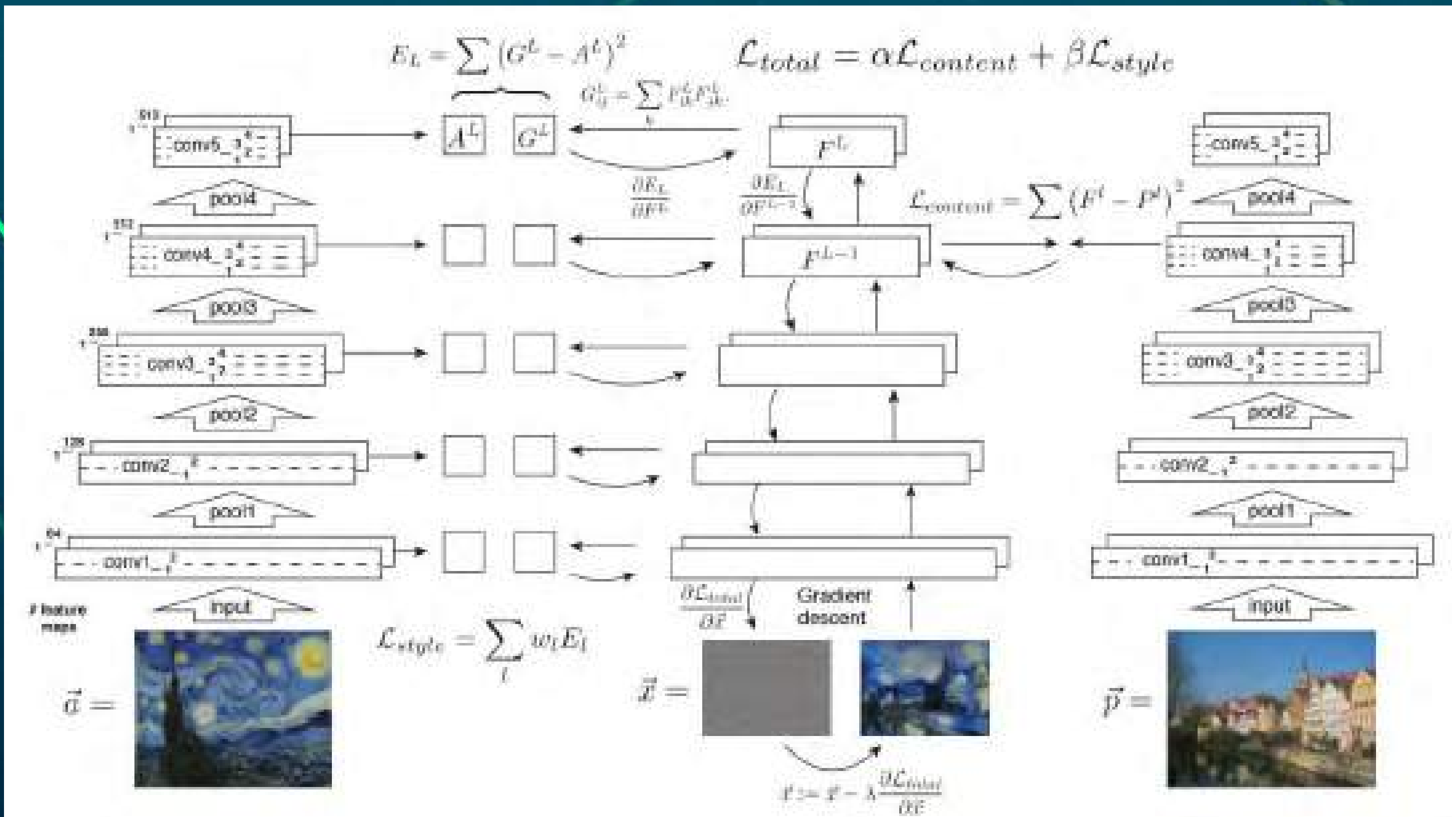
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	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					

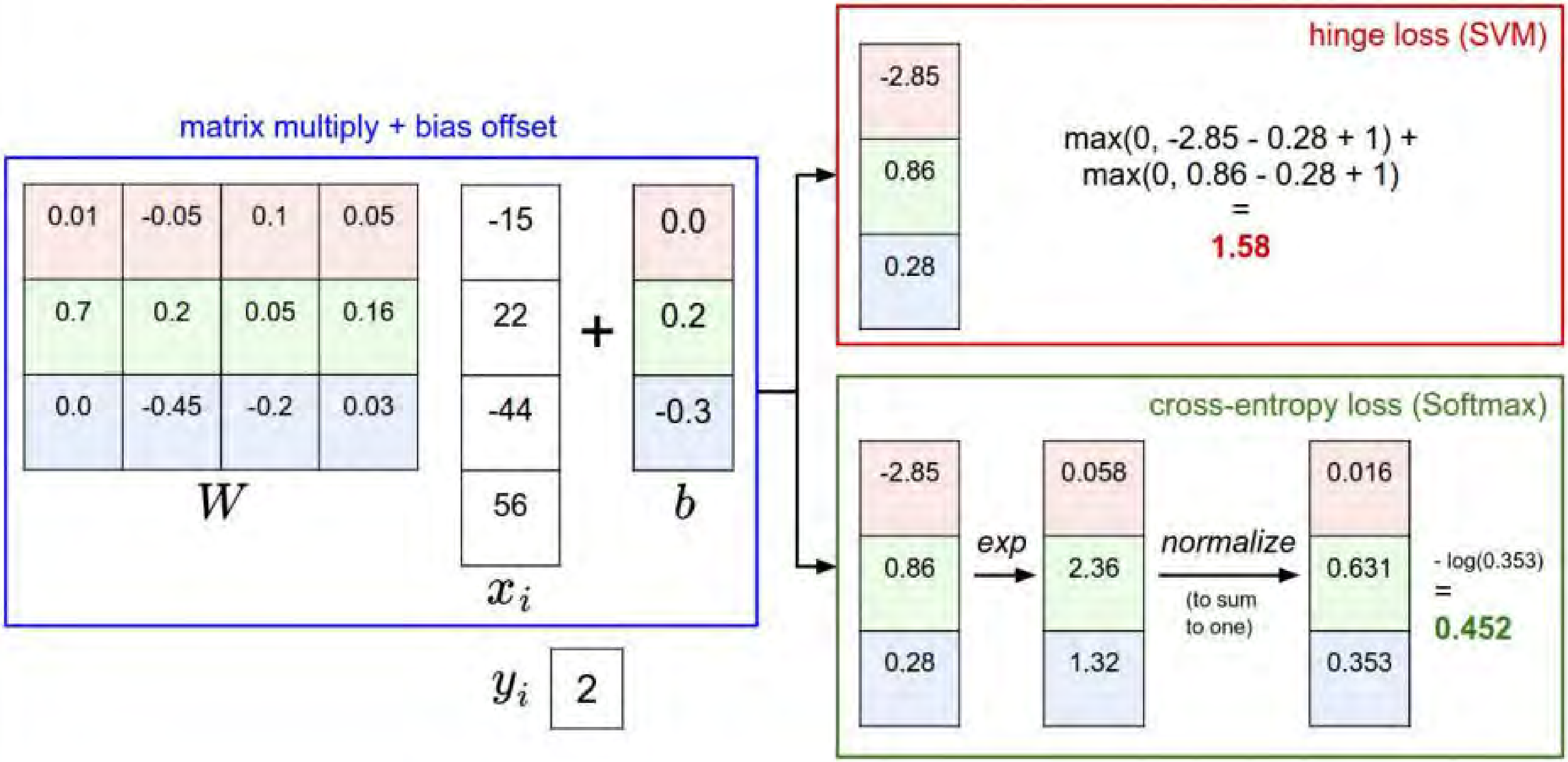
Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

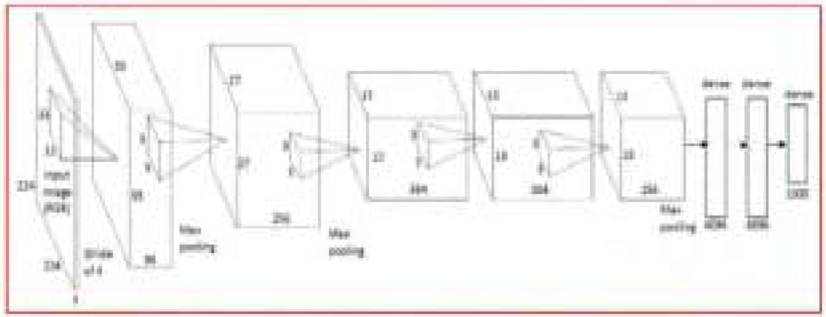
Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

OSC源创会 年终盛典 2016




matrix multiply + bias offset
0.01
-0.05
0.1
0.05
0.7
0.2
0.05
0.16
0.0
-0.45
-0.2
0.03
W
-15
22
-44
56
x_i
+
0.0
0.2
-0.3
b
y_i
2
hinge loss (SVM)
-2.85
0.86
0.28
max(0, -2.85 - 0.28 + 1) +
max(0, 0.86 - 0.28 + 1)
=
1.58
cross-entropy loss (Softmax)
-2.85
0.86
0.28
exp
0.058
2.36
1.32
normalize
(to sum
to one)
0.016
0.631
0.353
- log(0.353)
=
0.452

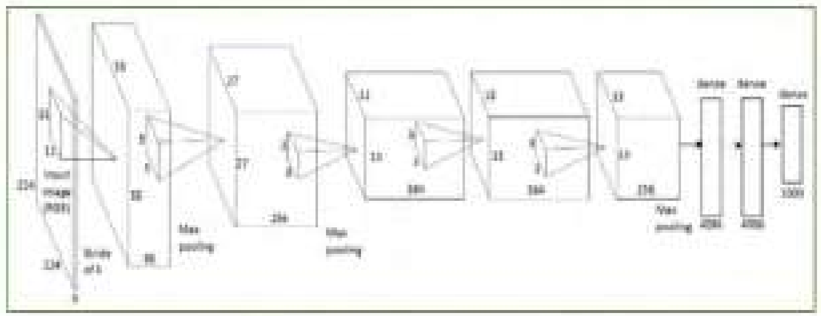
Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)



content activations

e.g.
at CONV5_1 layer we would have a [14x14x512] array of target activations

Step 2: Extract **style targets** (Gram matrices of ConvNet activations of all layers for the given style image)



style gram matrices

$$G = V^{T}V$$

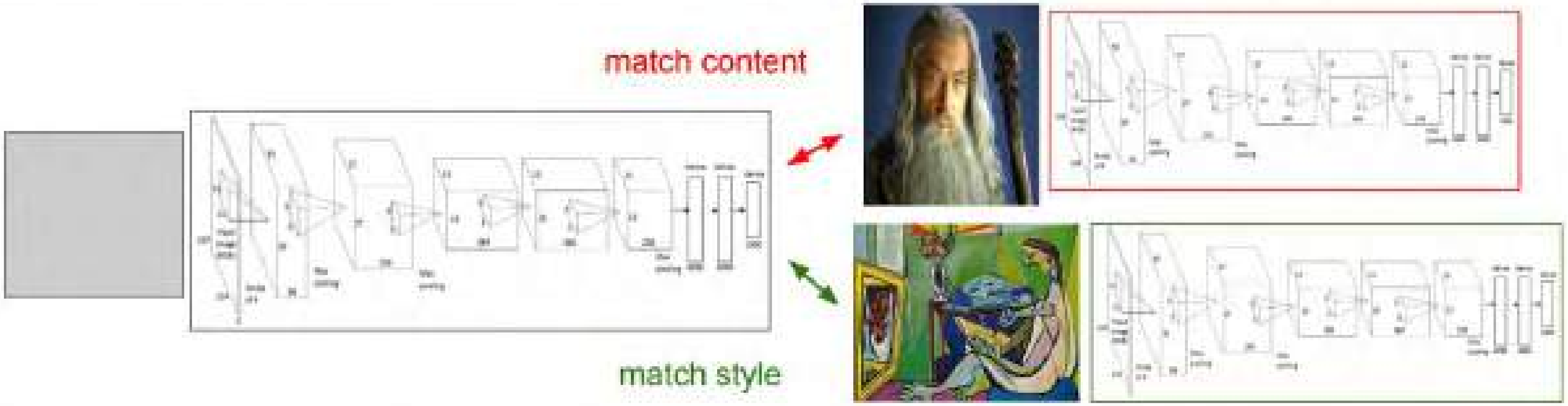
e.g.
at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

Step 3: Optimize over image to have:

- The **content** of the content image (activations match content)
- The **style** of the style image (Gram matrices of activations match style)

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

(+Total Variation regularization (maybe))

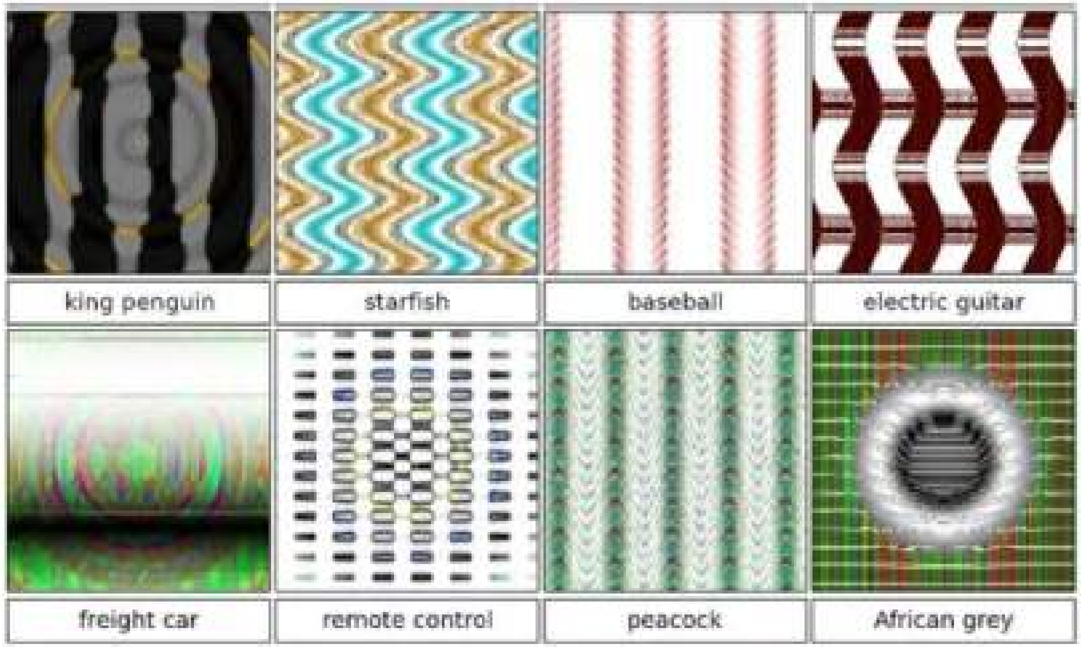


[Intriguing properties of neural networks, Szegedy et al., 2013]

correct +distort ostrich correct +distort ostrich

[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

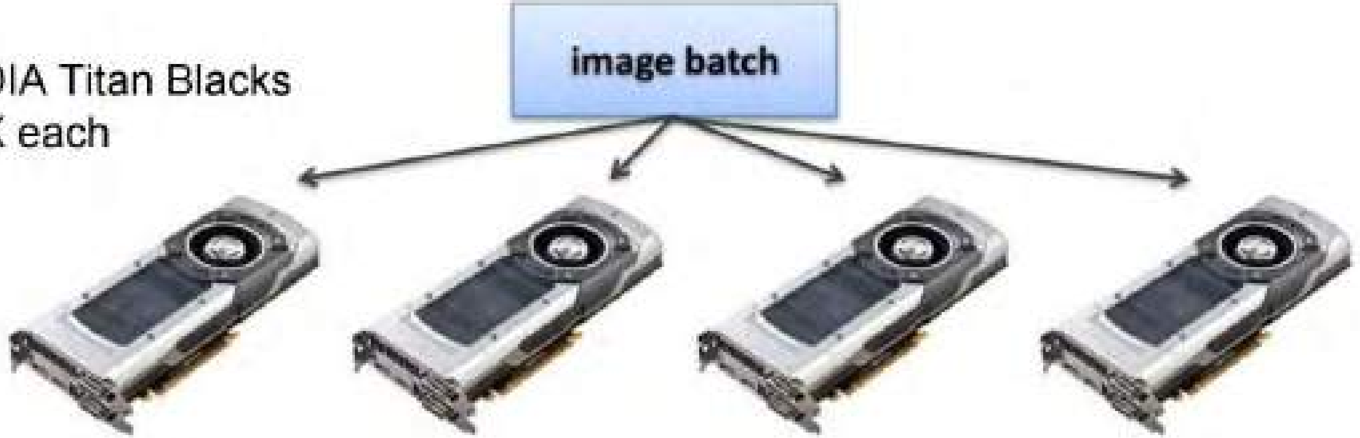
>99.6% confidences



Even with GPUs, training can be slow

VGG: ~2-3 weeks training with 4 GPUs
ResNet 101: 2-3 weeks with 4 GPUs

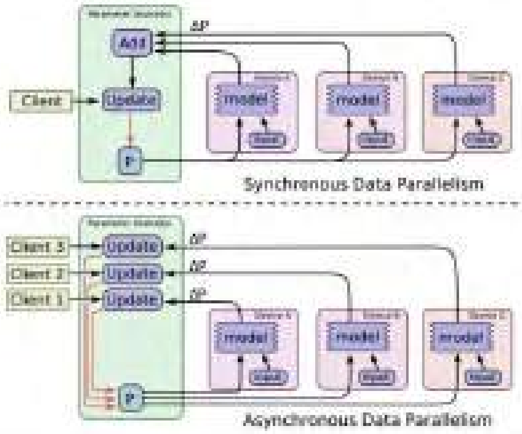
NVIDIA Titan Blacks
~$1K each



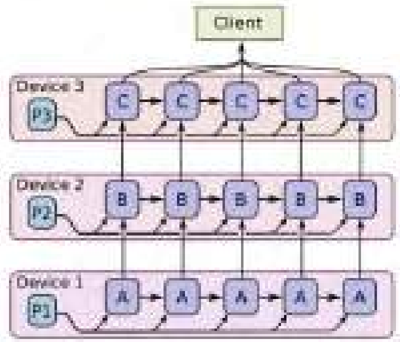
ResNet reimplemented in Torch: http://torch.ch/blog/2016/02/04/resnets.html

TensorFlow: Multi-GPU

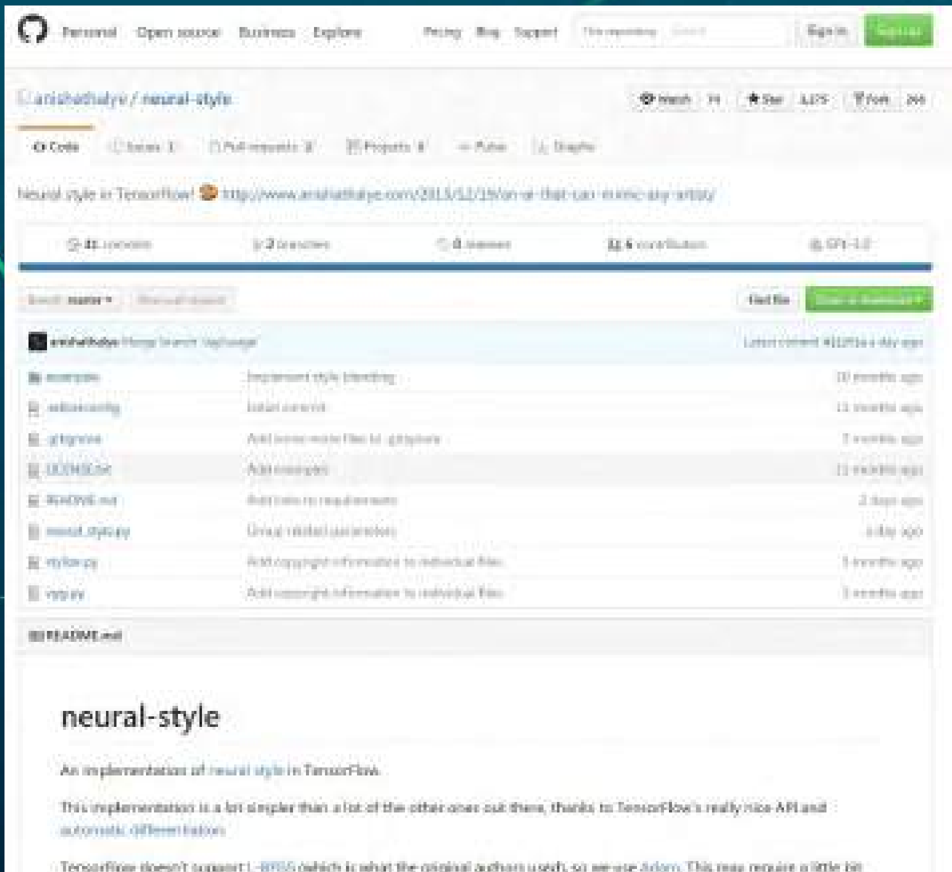
Data parallelism:
synchronous or asynchronous



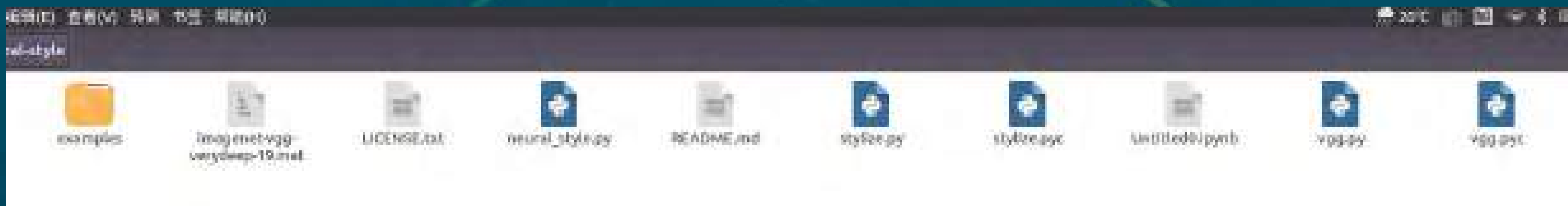
Model parallelism:
Split model across GPUs



OSC源创会 年终盛典 2016



OSC源创会 年终盛典 2016



```python
def build_parser():
    parser = ArgumentParser()
    parser.add_argument('--content',
            dest='content', help='content image',
            metavar='CONTENT', required=True)
    parser.add_argument('--styles',
            dest='styles',
            nargs='+', help='one or more style images',
            metavar='STYLE', required=True)
    parser.add_argument('--output',
            dest='output', help='output path',
            metavar='OUTPUT', required=True)
    parser.add_argument('--checkpoint-output',
            dest='checkpoint_output', help='checkpoint output format',
            metavar='OUTPUT')
    parser.add_argument('--iterations', type=int,
            dest='iterations', help='iterations (default %(default)s)',
            metavar='ITERATIONS', default=ITERATIONS)
    parser.add_argument('--width', type=int,
            dest='width', help='output width',
            metavar='WIDTH')
    parser.add_argument('--style-scales', type=float,
            dest='style_scales',
            nargs='+', help='one or more style scales',
            metavar='STYLE_SCALE')
    parser.add_argument('--network',
            dest='network', help='path to network parameters (default %(default)s)',
            metavar='VGG_PATH', default=VGG_PATH)
    parser.add_argument('--content-weight', type=float,
            dest='content_weight', help='content weight (default %(default)s)',
            metavar='CONTENT_WEIGHT', default=CONTENT_WEIGHT)
```

```python
    parser.add_argument('--style-weight', type=float,
            dest='style_weight', help='style weight (default %(default)s)',
            metavar='STYLE_WEIGHT', default=STYLE_WEIGHT)
    parser.add_argument('--style-blend-weights', type=float,
            dest='style_blend_weights', help='style blending weights',
            nargs='+', metavar='STYLE_BLEND_WEIGHT')
    parser.add_argument('--tv-weight', type=float,
            dest='tv_weight', help='total variation regularization weight (default %(default)s)',
            metavar='TV_WEIGHT', default=TV_WEIGHT)
    parser.add_argument('--learning-rate', type=float,
            dest='learning_rate', help='learning rate (default %(default)s)',
            metavar='LEARNING_RATE', default=LEARNING_RATE)
    parser.add_argument('--initial',
            dest='initial', help='initial image',
            metavar='INITIAL')
    parser.add_argument('--print-iterations', type=int,
            dest='print_iterations', help='statistics printing frequency',
            metavar='PRINT_ITERATIONS')
    parser.add_argument('--checkpoint-iterations', type=int,
            dest='checkpoint_iterations', help='checkpoint frequency',
            metavar='CHECKPOINT_ITERATIONS')
    return parser
```

```
for iteration, image in stylize(
    network=options.network,
    initial=initial,
    content=content_image,
    styles=style_images,
    iterations=options.iterations,
    content_weight=options.content_weight,
    style_weight=options.style_weight,
    style_blend_weights=style_blend_weights,
    tv_weight=options.tv_weight,
    learning_rate=options.learning_rate,
    print_iterations=options.print_iterations,
    checkpoint_iterations=options.checkpoint_iterations
):
    output_file = None
    if iteration is not None:
        if options.checkpoint_output:
            output_file = options.checkpoint_output % iteration
    else:
        output_file = options.output
    if output_file:
        imsave(output_file, image)
```

```
def _conv_layer(input, weights, bias):
    conv = tf.nn.conv2d(input, tf.constant(weights), strides=(1, 1, 1, 1),
            padding='SAME')
    return tf.nn.bias_add(conv, bias)


def _pool_layer(input):
    return tf.nn.max_pool(input, ksize=(1, 2, 2, 1), strides=(1, 2, 2, 1),
            padding='SAME')


def preprocess(image, mean_pixel):
    return image - mean_pixel


def unprocess(image, mean_pixel):
    return image + mean_pixel
```

```
def net(data_path, input_image):
    layers = (
        'conv1_1', 'relu1_1', 'conv1_2', 'relu1_2', 'pool1',

        'conv2_1', 'relu2_1', 'conv2_2', 'relu2_2', 'pool2',

        'conv3_1', 'relu3_1', 'conv3_2', 'relu3_2', 'conv3_3',
        'relu3_3', 'conv3_4', 'relu3_4', 'pool3',

        'conv4_1', 'relu4_1', 'conv4_2', 'relu4_2', 'conv4_3',
        'relu4_3', 'conv4_4', 'relu4_4', 'pool4',

        'conv5_1', 'relu5_1', 'conv5_2', 'relu5_2', 'conv5_3',
        'relu5_3', 'conv5_4', 'relu5_4'
    )

    data = scipy.io.loadmat(data_path)
    mean = data['normalization'][0][0][0]
    mean_pixel = np.mean(mean, axis=(0, 1))
    weights = data['layers'][0]

    net = {}
    current = input_image
    for i, name in enumerate(layers):
        kind = name[:4]
        if kind == 'conv':
            kernels, bias = weights[i][0][0][0][0]
            # matconvnet: weights are [width, height, in_channels, out_channels]
            # tensorflow: weights are [height, width, in_channels, out_channels]
            kernels = np.transpose(kernels, (1, 0, 2, 3))
            bias = bias.reshape(-1)
            current = _conv_layer(current, kernels, bias)
        elif kind == 'relu':
            current = tf.nn.relu(current)
        elif kind == 'pool':
            current = _pool_layer(current)
        net[name] = current

    assert len(net) == len(layers)
    return net, mean_pixel
```

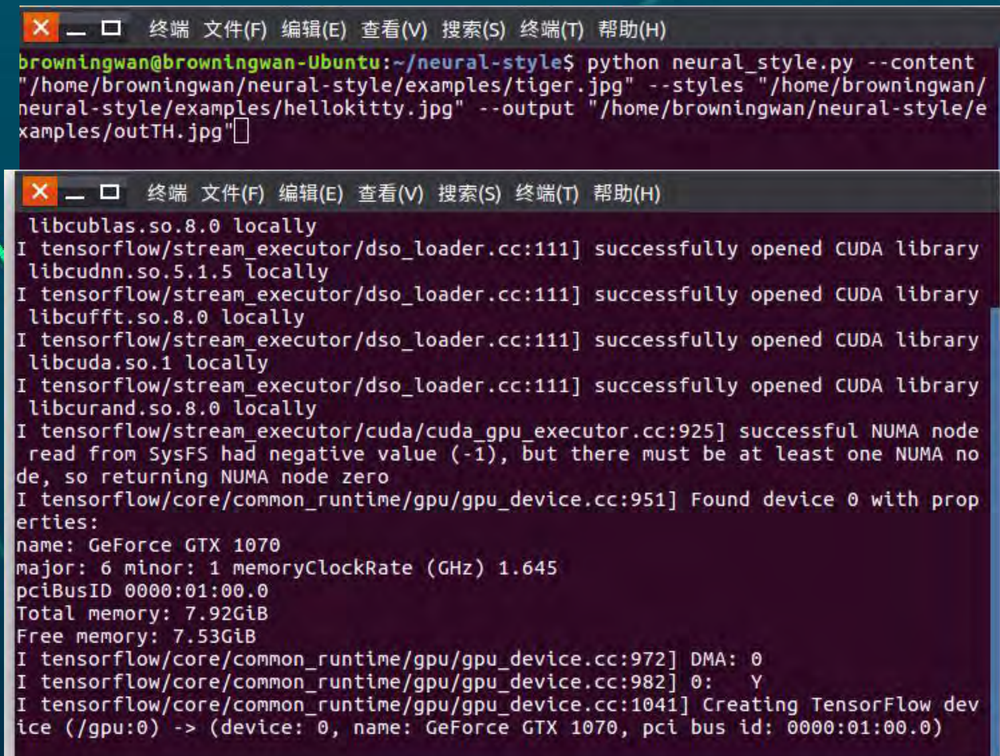
```python
# compute content features in feedforward mode
g = tf.Graph()
with g.as_default(), g.device('/cpu:0'), tf.Session() as sess:
    image = tf.placeholder('float', shape=shape)
    net, mean_pixel = vgg.net(network, image)
    content_pre = np.array([vgg.preprocess(content, mean_pixel)])
    content_features[CONTENT_LAYER] = net[CONTENT_LAYER].eval(
            feed_dict={image: content_pre})

# compute style features in feedforward mode
for i in range(len(styles)):
    g = tf.Graph()
    with g.as_default(), g.device('/cpu:0'), tf.Session() as sess:
        image = tf.placeholder('float', shape=style_shapes[i])
        net, _ = vgg.net(network, image)
        style_pre = np.array([vgg.preprocess(styles[i], mean_pixel)])
        for layer in STYLE_LAYERS:
            features = net[layer].eval(feed_dict={image: style_pre})
            features = np.reshape(features, (-1, features.shape[3]))
            gram = np.matmul(features.T, features) / features.size
            style_features[i][layer] = gram
```

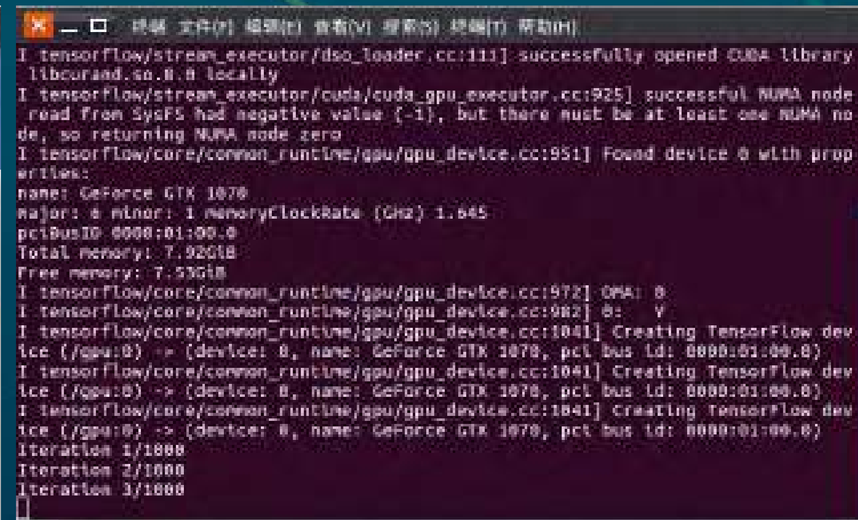
```python
# make stylized image using backpropogation
with tf.Graph().as_default():
    if initial is None:
        noise = np.random.normal(size=shape, scale=np.std(content) * 0.1)
        initial = tf.random_normal(shape) * 0.256
    else:
        initial = np.array([vgg.preprocess(initial, mean_pixel)])
        initial = initial.astype('float32')
    image = tf.Variable(initial)
    net, _ = vgg.net(network, image)
```

```python
# content loss
content_loss = content_weight * (2 * tf.nn.l2_loss(
        net[CONTENT_LAYER] - content_features[CONTENT_LAYER]) /
        content_features[CONTENT_LAYER].size)
# style loss
style_loss = 0
for i in range(len(styles)):
    style_losses = []
    for style_layer in STYLE_LAYERS:
        layer = net[style_layer]
        _, height, width, number = map(lambda i: i.value, layer.get_shape())
        size = height * width * number
        feats = tf.reshape(layer, (-1, number))
        gram = tf.matmul(tf.transpose(feats), feats) / size
        style_gram = style_features[i][style_layer]
        style_losses.append(2 * tf.nn.l2_loss(gram - style_gram) / style_gram.size)
    style_loss += style_weight * style_blend_weights[i] * reduce(tf.add, style_losses)
# total variation denoising
tv_y_size = _tensor_size(image[:,1:,:,:])
tv_x_size = _tensor_size(image[:,:,1:,:])
tv_loss = tv_weight * 2 * (
        (tf.nn.l2_loss(image[:,1:,:,:] - image[:,:shape[1]-1,:,:]) /
            tv_y_size) +
        (tf.nn.l2_loss(image[:,:,1:,:] - image[:,:,:shape[2]-1,:]) /
            tv_x_size))
# overall loss
loss = content_loss + style_loss + tv_loss

# optimizer setup
train_step = tf.train.AdamOptimizer(learning_rate).minimize(loss)
```

```python
with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    for i in range(iterations):
        last_step = (i == iterations - 1)
        print_progress(i, last=last_step)
        train_step.run()

        if (checkpoint_iterations and i % checkpoint_iterations == 0) or last_step:
            this_loss = loss.eval()
            if this_loss < best_loss:
                best_loss = this_loss
                best = image.eval()
            yield (
                (None if last_step else i),
                vgg.unprocess(best.reshape(shape[1:]), mean_pixel)
            )
```

OSC源创会 年终盛典 2016





```
终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Iteration 85/100
Iteration 86/100
Iteration 87/100
Iteration 88/100
Iteration 89/100
Iteration 90/100
Iteration 91/100
Iteration 92/100
Iteration 93/100
Iteration 94/100
Iteration 95/100
Iteration 96/100
Iteration 97/100
Iteration 98/100
Iteration 99/100
Iteration 100/100
  content loss: 1.64637e+06
    style loss: 2.04003e+06
       tv loss: 51504.3
    total loss: 3.73790e+06
browningwan@browningwan-Ubuntu:~/neural-style$ python neural_style.py --content "/home/browningwan/neural-style/examples/tiger.jpg" --styles "/home/browningwan/neural-style/examples/hellokitty.jpg" --output "/home/browningwan/neural-style/examples/outTH.jpg" --iterations 10
```


Hello kitty

OSC源创会 年终盛典 2016

Pip installation on Windows

TensorFlow supports only 64-bit Python 3.5 on Windows. We have tested the pip packages with the following distributions of Python:

- Python 3.5 from python.org
- Python 3.5 from Anaconda

Both distributions include pip. To install the CPU-only version of TensorFlow, enter the following command at a command prompt:

```
C:\> pip install --upgrade https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow-(
```

To install the GPU version of TensorFlow, enter the following command at a command prompt:

```
C:\> pip install --upgrade https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_(
```

You can now test your installation.

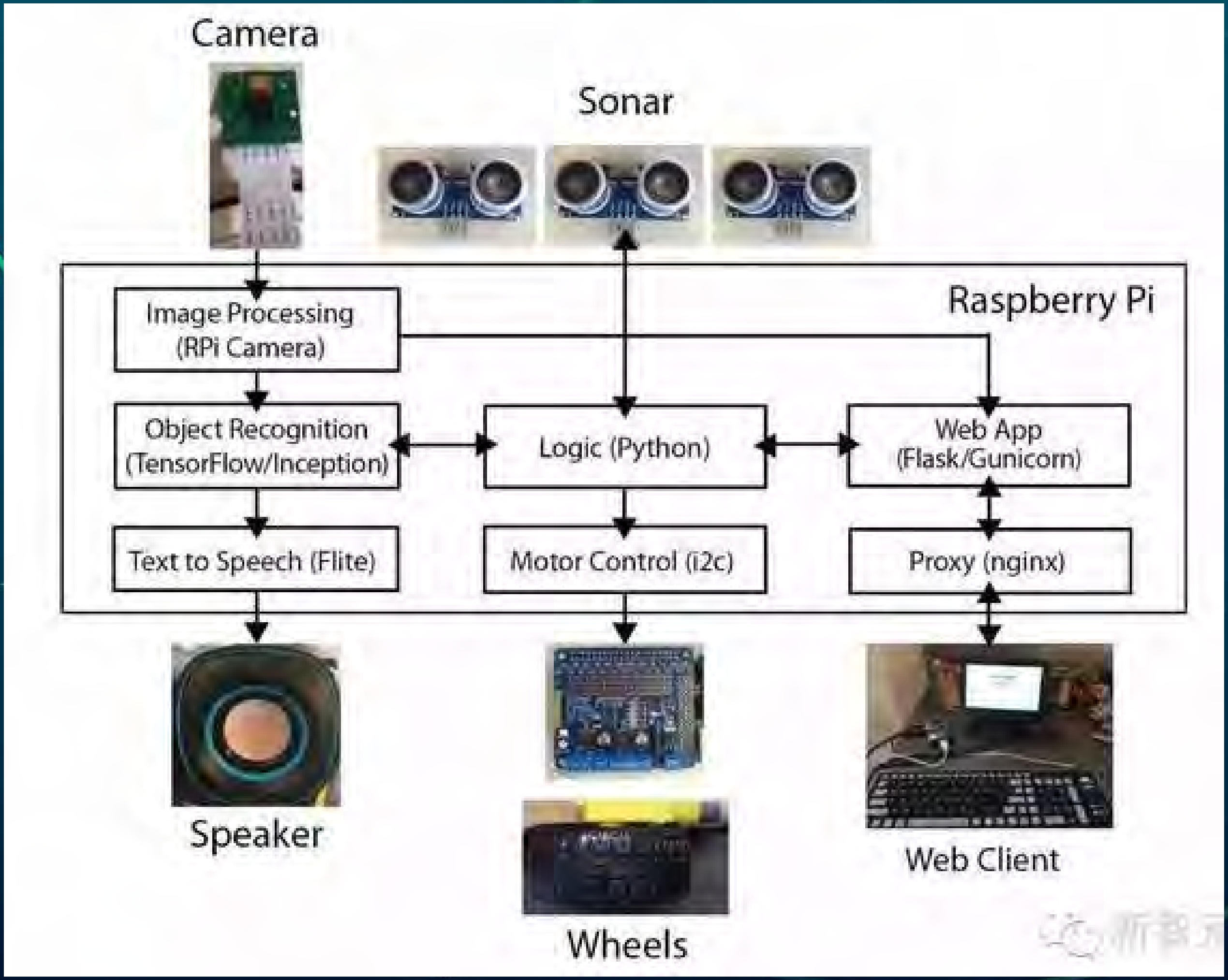
You can also use Virtualenv or Anaconda environments to manage your installation of TensorFlow on Windows.

4G


15% Tabby
14% Lynx
12% Tiger Cat
10% Egyptian Cat
Freeze Frame


notebook: 0.385
computer keyboard: 0.24
laptop: 0.165


desktop computer: 0.522
screen: 0.235


Camera
Sonar
Raspberry Pi
Image Processing (RPi Camera)
Object Recognition (TensorFlow/Inception)
Logic (Python)
Web App (Flask/Gunicorn)
Text to Speech (Flite)
Motor Control (i2c)
Proxy (nginx)
Speaker
Wheels
Web Client

- 感谢支持

AG Group 万元芳