网易人工智能 网易云 163yun.com IT大咖说 Boxue 博学教育

网易博学实践日
大数据与人工智能技术大会

2017.08.12

tensorflow 在艺术创作方面的应用

黄琨 @ 中国搜索

- tensorflow 简介
- 深度神经网络算法概述
- generative model 与艺术创作

tensorflow 简介

tensorflow 是 google 开源的通用计算框架

官网：https://www.tensorflow.org

github：https://github.com/tensorflow

TF 擅长机器学习，尤其是 tensor 运算

- why tensor ?

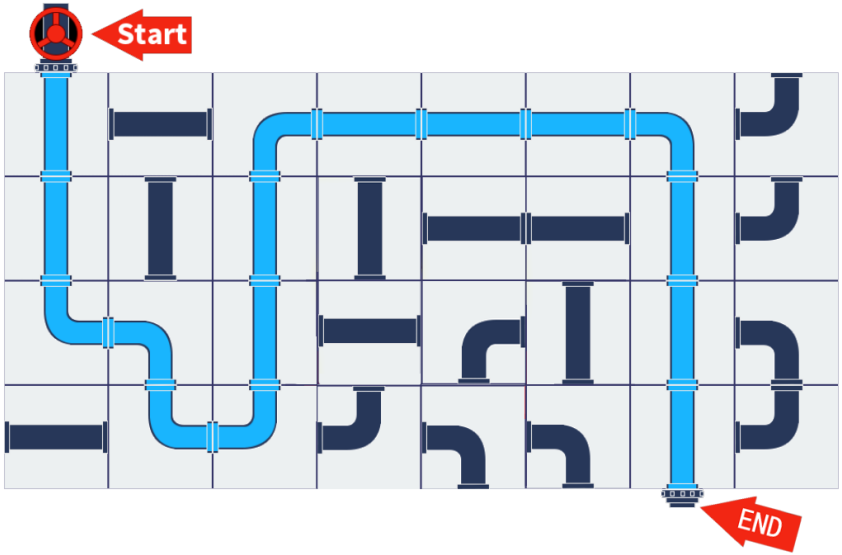
1. 元素操作（elementwise operations）
2. 样本批处理（samples batch）
3. 线性运算（linear algebra）

$$a = \text{sigmoid}(\overbrace{Wx + b}^{\text{linear algebra}})$$

elementwise operation　　samples batch

TF 编程模式

“tensor” + “flow”

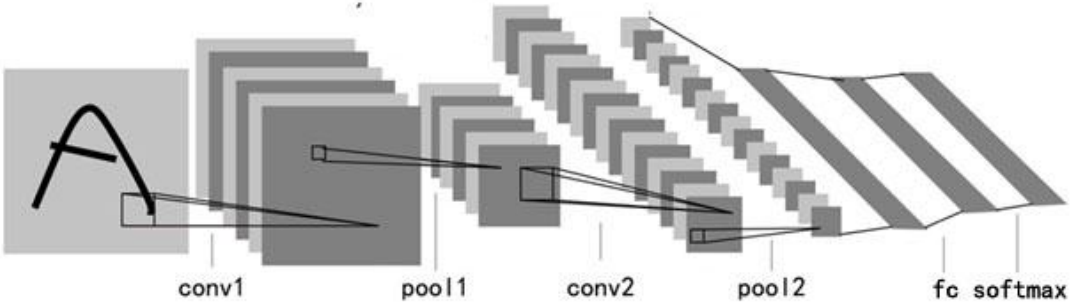


step-1: 连接“管道”
（building computational graph）

step-2: 打开“水闸”
（running computational graph）

代码示例

```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")
```

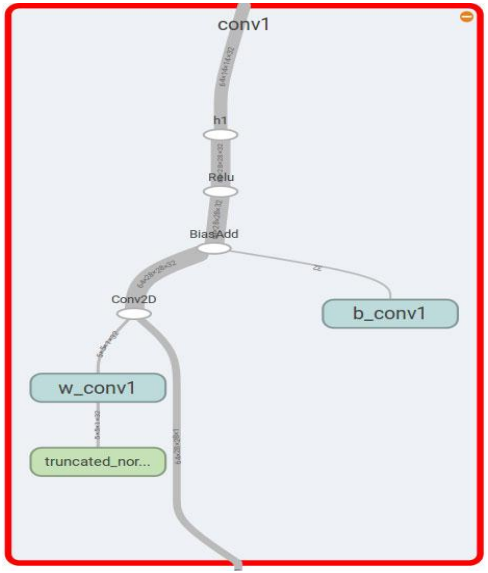


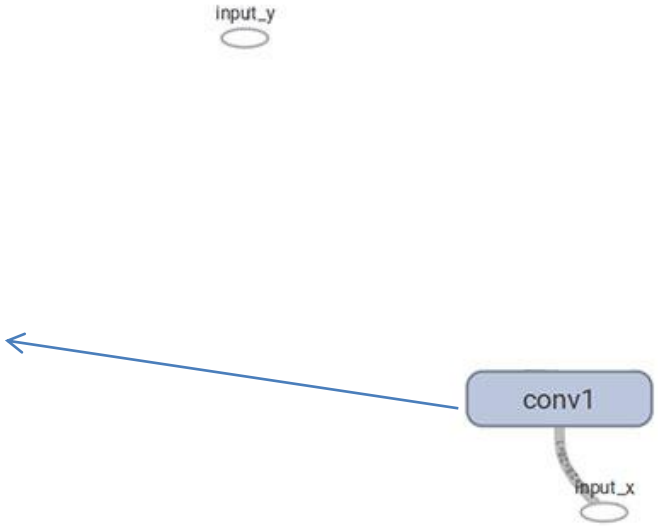
input_y

input_x

代码示例

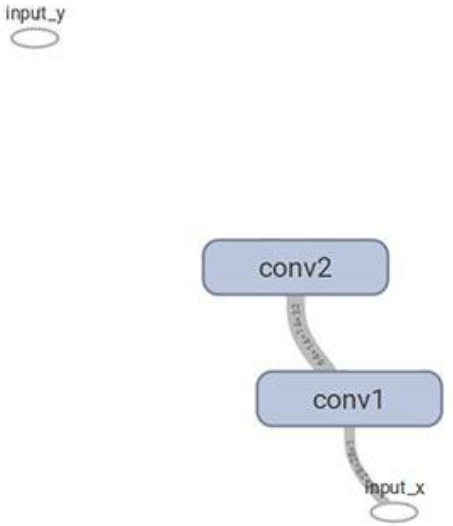
```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 2, 1], strides=[1, 2, 2, 2, 1], padding='SAME', name="h1")
```





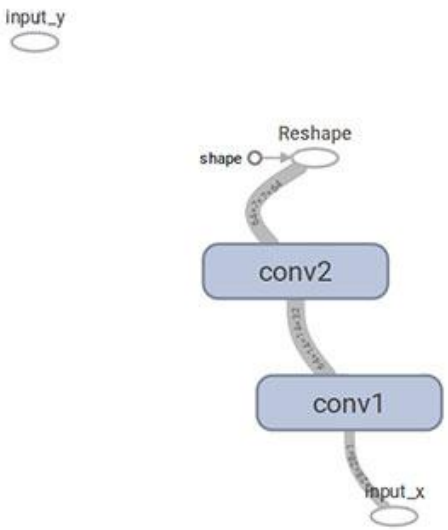
代码示例

```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h1")

with tf.name_scope('conv2'):
  conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv2")
  conv2_biases = tf.Variable(tf.constant(0.1, shape=[64], dtype=tf.float32), name="b_conv2")
  conv = tf.nn.conv2d(pool, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h2")
```



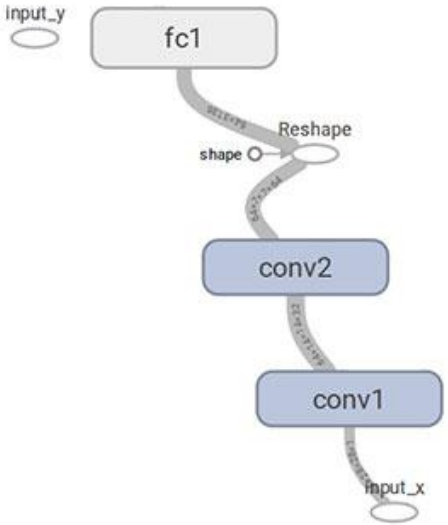
代码示例

```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h1")

with tf.name_scope('conv2'):
  conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv2")
  conv2_biases = tf.Variable(tf.constant(0.1, shape=[64], dtype=tf.float32), name="b_conv2")
  conv = tf.nn.conv2d(pool, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h2")

pool_shape = pool.get_shape().as_list()
reshape = tf.reshape(pool, [pool_shape[0], pool_shape[1] * pool_shape[2] * pool_shape[3]])
```



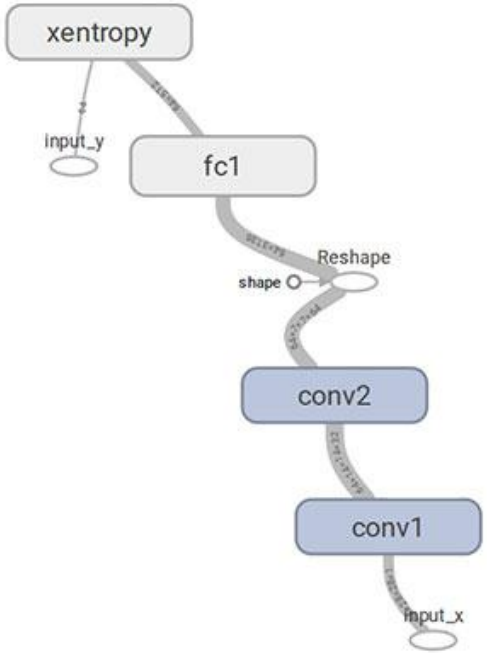
代码示例

```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h1")

with tf.name_scope('conv2'):
  conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv2")
  conv2_biases = tf.Variable(tf.constant(0.1, shape=[64], dtype=tf.float32), name="b_conv2")
  conv = tf.nn.conv2d(pool, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h2")

pool_shape = pool.get_shape().as_list()
reshape = tf.reshape(pool, [pool_shape[0], pool_shape[1] * pool_shape[2] * pool_shape[3]])

with tf.name_scope('fc1'):
  fc1_weights = tf.Variable(tf.truncated_normal([3136, 512], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc1")
  fc1_biases = tf.Variable(tf.constant(0.1, shape=[512], dtype=tf.float32), name="b_fc1")
  hidden = tf.nn.relu(tf.matmul(reshape, fc1_weights) + fc1_biases)
  hidden = tf.nn.dropout(hidden, 0.5, seed=SEED, name="h_3")
```



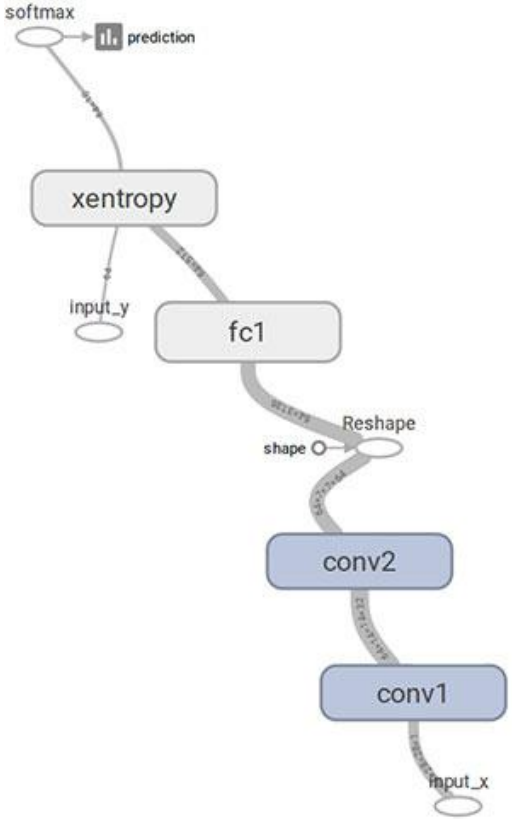
代码示例

```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h1")

with tf.name_scope('conv2'):
  conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv2")
  conv2_biases = tf.Variable(tf.constant(0.1, shape=[64], dtype=tf.float32), name="b_conv2")
  conv = tf.nn.conv2d(pool, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h2")

pool_shape = pool.get_shape().as_list()
reshape = tf.reshape(pool, [pool_shape[0], pool_shape[1] * pool_shape[2] * pool_shape[3]])

with tf.name_scope('fc1'):
  fc1_weights = tf.Variable(tf.truncated_normal([3136, 512], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc1")
  fc1_biases = tf.Variable(tf.constant(0.1, shape=[512], dtype=tf.float32), name="b_fc1")
  hidden = tf.nn.relu(tf.matmul(reshape, fc1_weights) + fc1_biases)
  hidden = tf.nn.dropout(hidden, 0.5, seed=SEED, name="h_3")

with tf.name_scope('xentropy'):
  fc2_weights = tf.Variable(tf.truncated_normal([512, NUM_LABELS], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc2")
  fc2_biases = tf.Variable(tf.constant(0.1, shape=[NUM_LABELS], dtype=tf.float32), name="b_fc2")
  logits = tf.matmul(hidden, fc2_weights) + fc2_biases
  loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(labels=input_y, logits=logits), name="loss")
```



代码示例

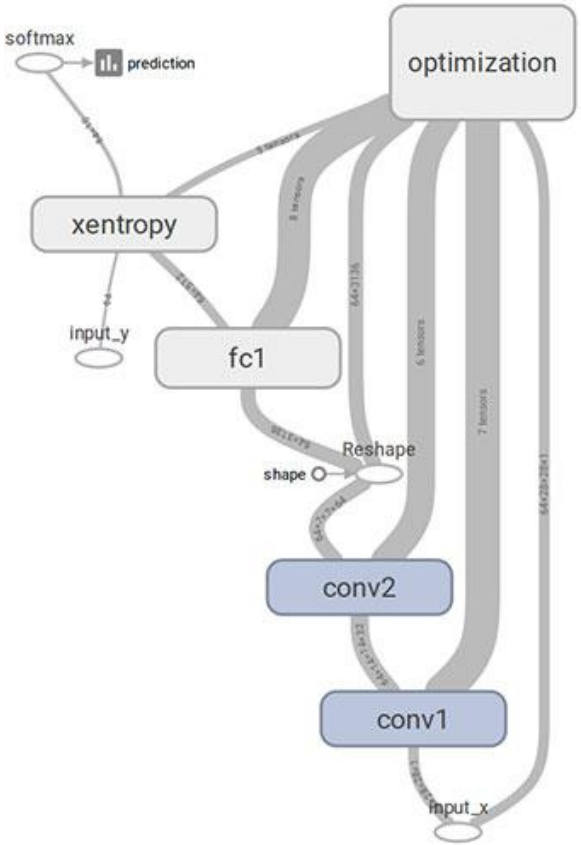
```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h1")

with tf.name_scope('conv2'):
  conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv2")
  conv2_biases = tf.Variable(tf.constant(0.1, shape=[64], dtype=tf.float32), name="b_conv2")
  conv = tf.nn.conv2d(pool, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h2")

pool_shape = pool.get_shape().as_list()
reshape = tf.reshape(pool, [pool_shape[0], pool_shape[1] * pool_shape[2] * pool_shape[3]])

with tf.name_scope('fc1'):
  fc1_weights = tf.Variable(tf.truncated_normal([3136, 512], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc1")
  fc1_biases = tf.Variable(tf.constant(0.1, shape=[512], dtype=tf.float32), name="b_fc1")
  hidden = tf.nn.relu(tf.matmul(reshape, fc1_weights) + fc1_biases)
  hidden = tf.nn.dropout(hidden, 0.5, seed=SEED, name="h_3")

with tf.name_scope('xentropy'):
  fc2_weights = tf.Variable(tf.truncated_normal([512, NUM_LABELS], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc2")
  fc2_biases = tf.Variable(tf.constant(0.1, shape=[NUM_LABELS], dtype=tf.float32), name="b_fc2")
  logits = tf.matmul(hidden, fc2_weights) + fc2_biases
  loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(labels=input_y, logits=logits), name="loss")

tf.summary.scalar('loss', loss)
prediction = tf.nn.softmax(logits, name="softmax")
tf.summary.histogram('prediction', prediction)
```



代码示例

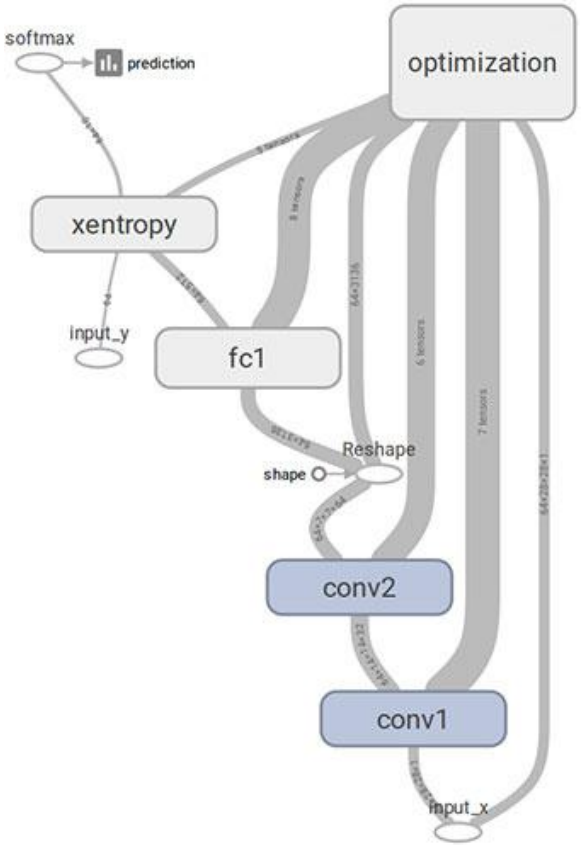
```
print("buiding graph")
input_x = tf.placeholder(tf.float32, shape=(64, 28, 28, 1), name="input_x")
input_y = tf.placeholder(tf.int64, shape=(64,), name="input_y")

with tf.name_scope('conv1'):
  conv1_weights = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv1")
  conv1_biases = tf.Variable(tf.zeros([32], dtype=tf.float32), name="b_conv1")
  conv = tf.nn.conv2d(input_x, conv1_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv1_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h1")

with tf.name_scope('conv2'):
  conv2_weights = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_conv2")
  conv2_biases = tf.Variable(tf.constant(0.1, shape=[64], dtype=tf.float32), name="b_conv2")
  conv = tf.nn.conv2d(pool, conv2_weights, strides=[1, 1, 1, 1], padding='SAME')
  relu = tf.nn.relu(tf.nn.bias_add(conv, conv2_biases))
  pool = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name="h2")

pool_shape = pool.get_shape().as_list()
reshape = tf.reshape(pool, [pool_shape[0], pool_shape[1] * pool_shape[2] * pool_shape[3]])

with tf.name_scope('fc1'):
  fc1_weights = tf.Variable(tf.truncated_normal([3136, 512], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc1")
  fc1_biases = tf.Variable(tf.constant(0.1, shape=[512], dtype=tf.float32), name="b_fc1")
  hidden = tf.nn.relu(tf.matmul(reshape, fc1_weights) + fc1_biases)
  hidden = tf.nn.dropout(hidden, 0.5, seed=SEED, name="h_3")

with tf.name_scope('xentropy'):
  fc2_weights = tf.Variable(tf.truncated_normal([512, NUM_LABELS], stddev=0.1, seed=SEED, dtype=tf.float32), name="w_fc2")
  fc2_biases = tf.Variable(tf.constant(0.1, shape=[NUM_LABELS], dtype=tf.float32), name="b_fc2")
  logits = tf.matmul(hidden, fc2_weights) + fc2_biases
  loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(labels=input_y, logits=logits), name="loss")

tf.summary.scalar('loss', loss)
prediction = tf.nn.softmax(logits, name="softmax")
tf.summary.histogram('prediction', prediction)

with tf.name_scope("optimization"):
  optimizer = tf.train.GradientDescentOptimizer(0.01)
  global_step = tf.Variable(0, name='global_step', trainable=False)
  train_op = optimizer.minimize(loss, global_step=global_step)
```

step-1: Building computational graph



代码示例

step-2: Running computational graph

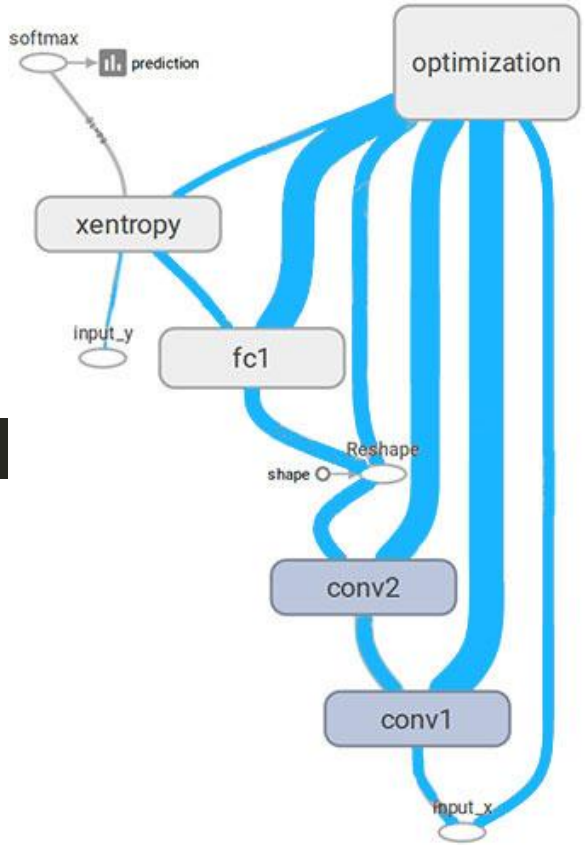


代码示例

step-2: Running computational graph

training 时：

```
sess.run(train_op, feed_dict={input_x: data, input_y: labels})
```

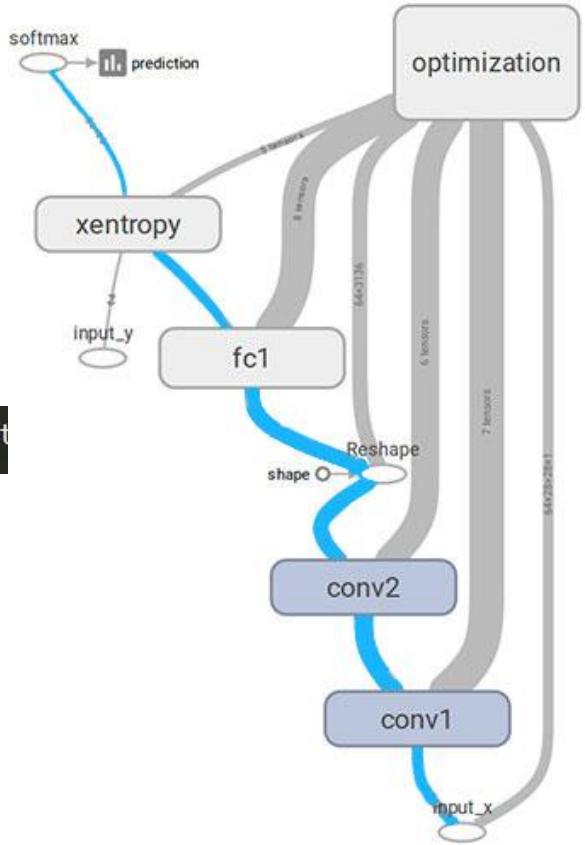


代码示例

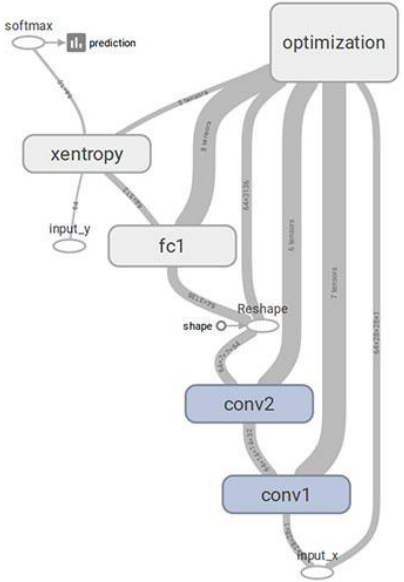
step-2: Running computational graph

inference 时：

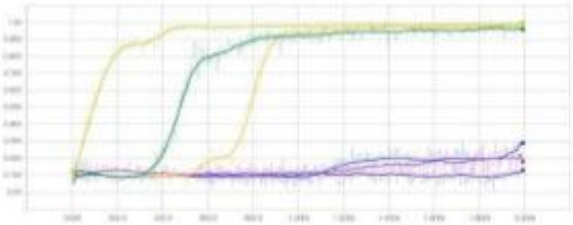
```
predictions = sess.run(prediction, feed_dict={input_x: data, input
```



可视化分析工具 tensorboard



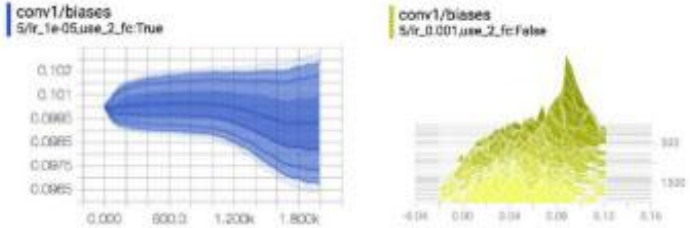
- **tf.summary.scalar**



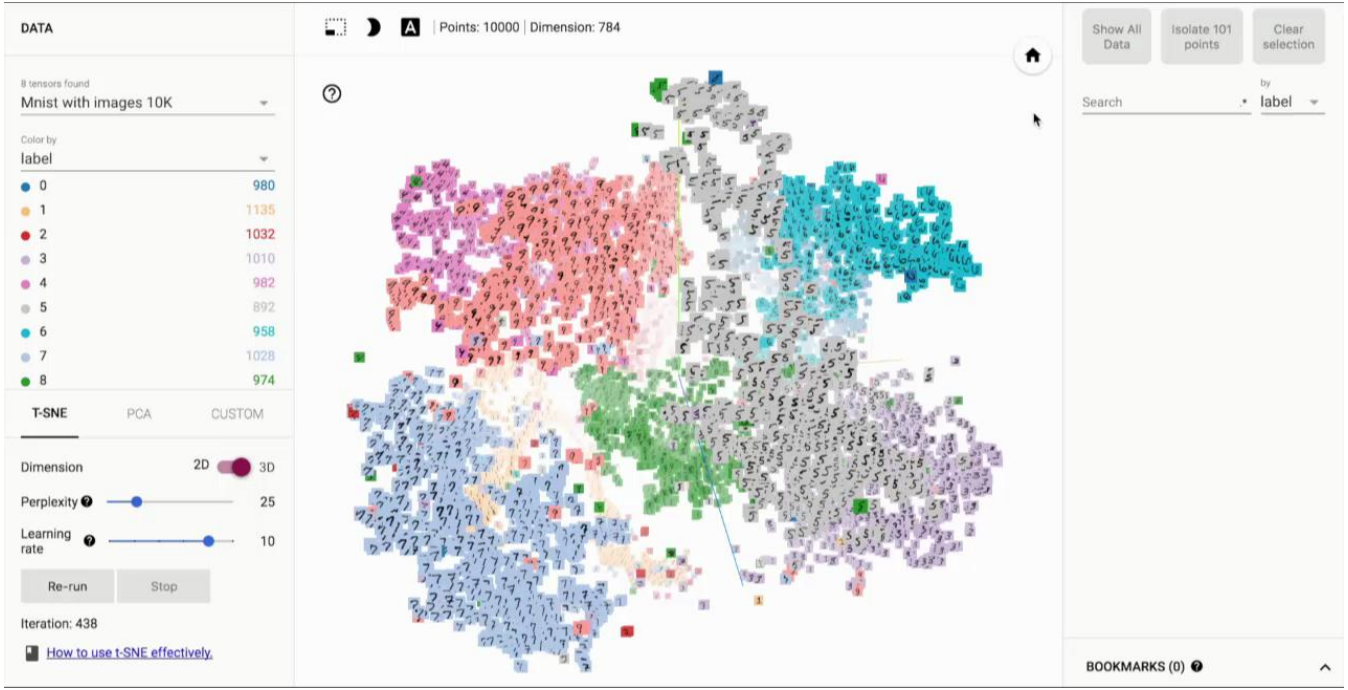
- **tf.summary.image**

- **tf.summary.audio**

- **tf.summary.histogram**



可视化分析工具 tensorboard

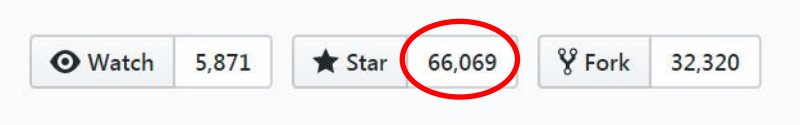


TF 其他特点

- 支持多核（cpu、gpu）计算
- 支持分布式计算（tensorflow + kubernetes）
- 易学性：和 numpy，theano 等有相似性
- 通用性：low/high-level function，跨平台，便于产品化
- 易用性：keras，tf.contrib.(learn, seq2seq 等)

TF 其他特点

- 社区活跃性



《神经网络和深度学习》第二章：“逆传播”算法的工作机制 原文链接：http://neu [...]

小结

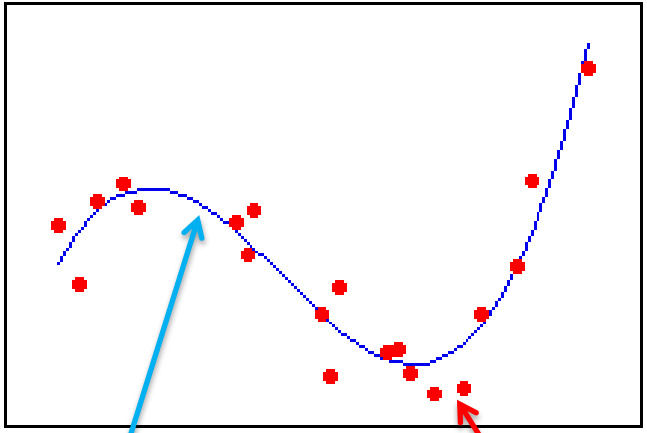
tensorflow，是足够好的平台，有足够多的“轮子”
让我们可以专注于研究具体应用、具体算法

Deep Learning：神经网络算法概述

基本思想

计算的目标： $P_{theory}(x|\theta) \xrightarrow{\theta^*} P_{real}(x)$

（最优化问题）



理论分布 P_{theory}

真实分布 P_{real}

例如：最小化 Kullback-Leibler Divergence $\mathrm{KL}(P_{theory}\|P_{real})$

$$S(\theta)=\mathrm{KL}(P_{theory}\|P_{real})=\int \mathrm{d}\,x\,\mathrm{P}_{\mathrm{theroy}}(x|\theta)(\log\mathrm{P}_{\mathrm{theroy}}(x|\theta)-\log\mathrm{P}_{\mathrm{real}}(x))$$

$$\theta^*=\arg\min_{\theta} S(\theta)$$

运算规模

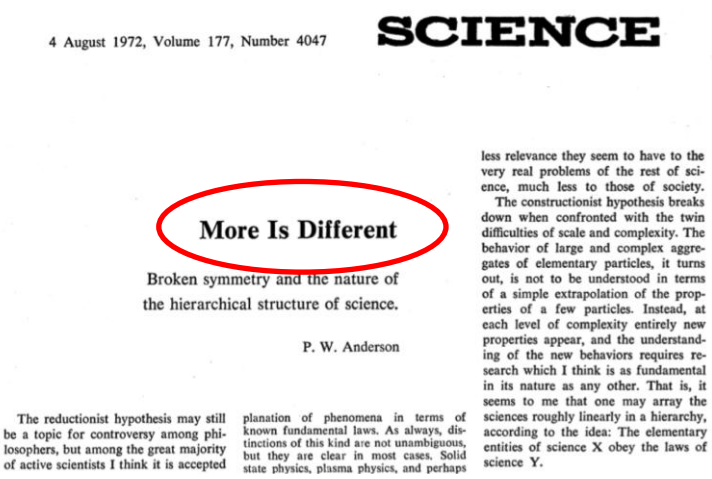
数据量级

Imagenet： 14,197,122 images

MNIST： 60,000 examples

参数量级

VGG model： 138,000,000 个



4 August 1972, Volume 177, Number 4047

SCIENCE

More Is Different

Broken symmetry and the nature of the hierarchical structure of science.

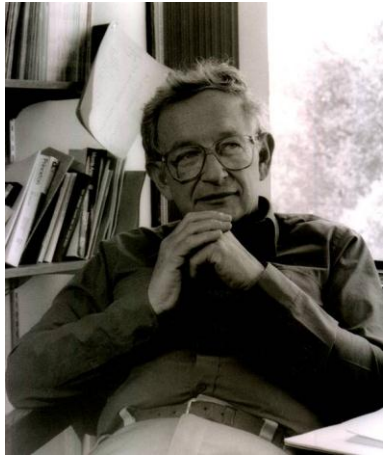
P. W. Anderson

The reductionist hypothesis may still be a topic for controversy among philosophers, but among the great majority of active scientists I think it is accepted

planation of phenomena in terms of known fundamental laws. As always, distinctions of this kind are not unambiguous, but they are clear in most cases. Solid state physics, plasma physics, and perhaps

less relevance they seem to have to the very real problems of the rest of science, much less to those of society.

The constructionist hypothesis breaks down when confronted with the twin difficulties of scale and complexity. The behavior of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead, at each level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other. That is, it seems to me that one may array the sciences roughly linearly in a hierarchy, according to the idea: The elementary entities of science X obey the laws of science Y.



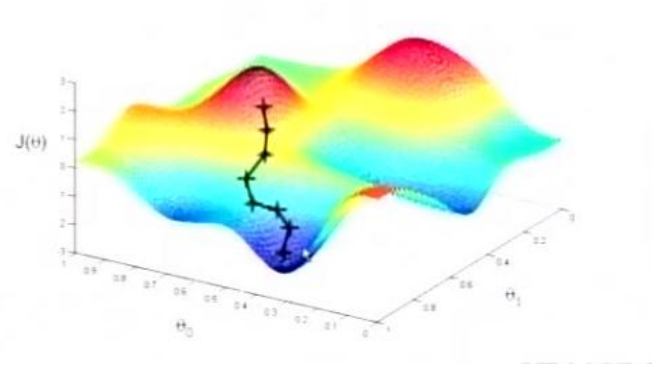
P. W. Anderson
Nobel prize Winner

怎样寻找最优解？

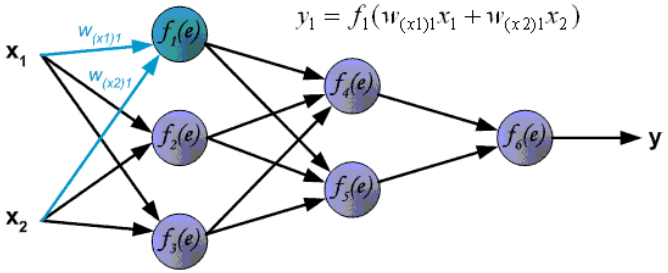
$$\theta^* = \arg\min_{\theta} S(\theta)$$

梯度下降法 （Stochastic Gradient Descent）

“逆传播”机制（Back Propagation）

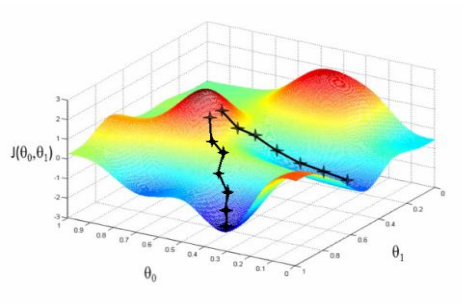


FP



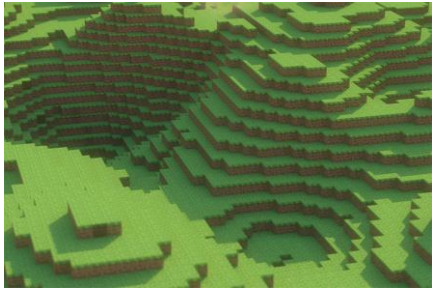
SGD的局限性

1. 局部极小值的“陷阱”



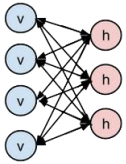
其他方案

2. 模型参数是离散值 （如 binary network）

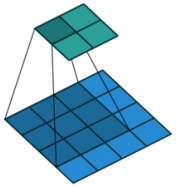


基本构件

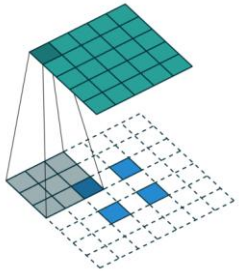
玻尔兹曼机（RBM）



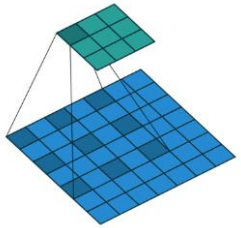
卷积神经网络（CNN）



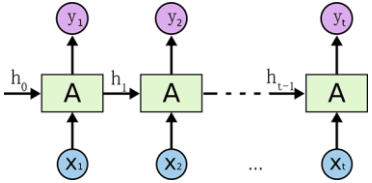
fractional cnn



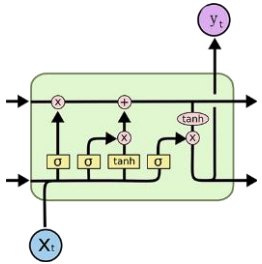
diated cnn



递归神经网络（RNN）

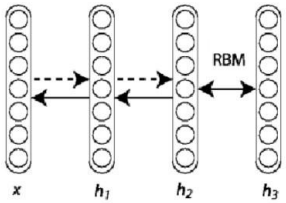


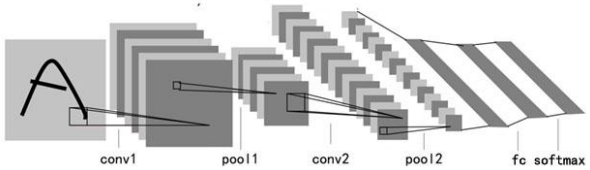
LSTM



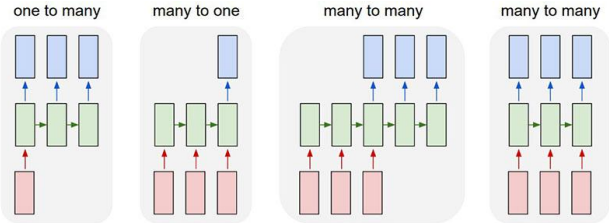
高阶模型

DBN （Deep Belief Network） / Stacked-CNN / ResNet

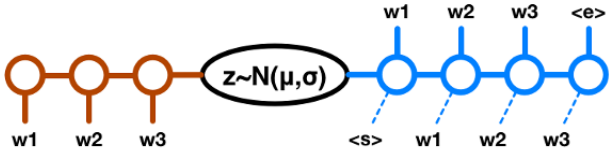




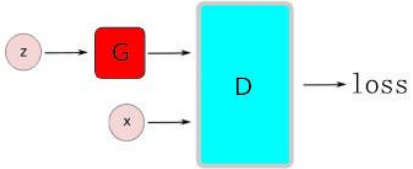
seq2seq



VAE（variational autoencoder）



GAN（generative adversarial networks）

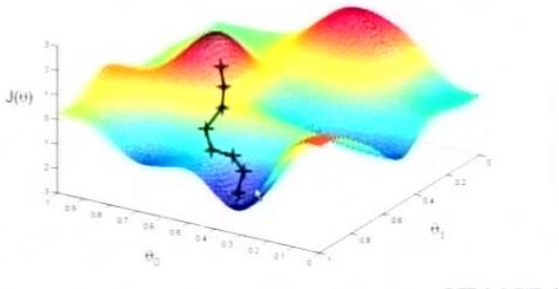


小结

深度神经网络算法的实现步骤

基本构件
如 cnn、rnn 等

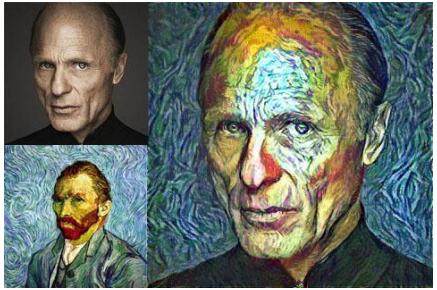
高阶模型
描述 $P_{theory}(x|\theta)$

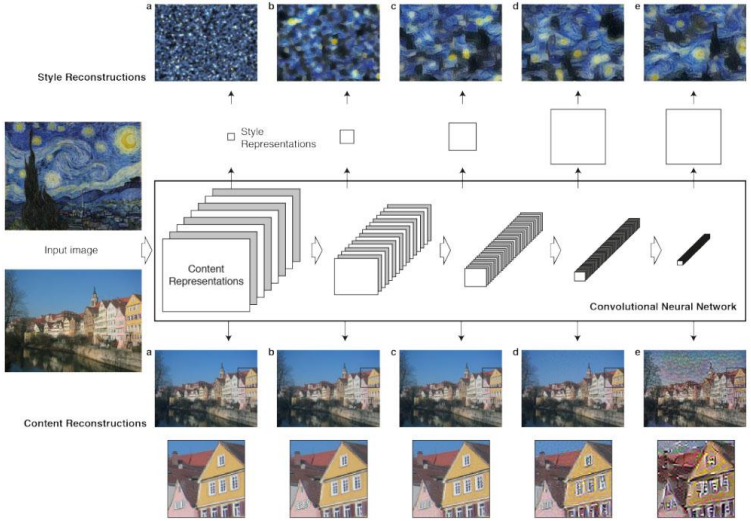


通过SGD等方法
找到模型最优解 $\theta^* = \arg\min_{\theta} S(\theta)$

generative model 与艺术创作

例子1： style transfer



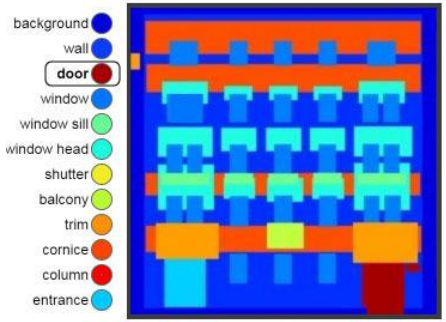


$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{content}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x})$$

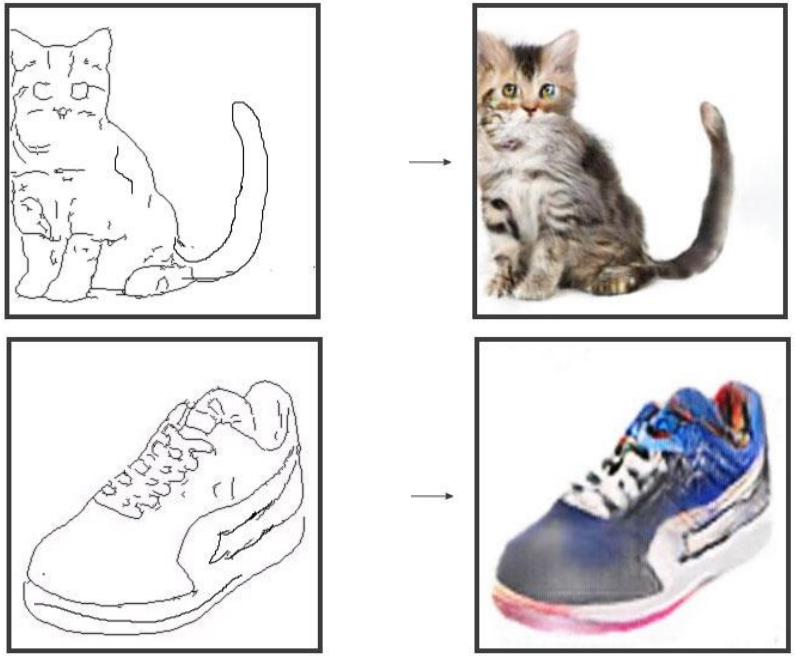
https://arxiv.org/abs/1508.06576

例子2： conditional-GAN

labels to image



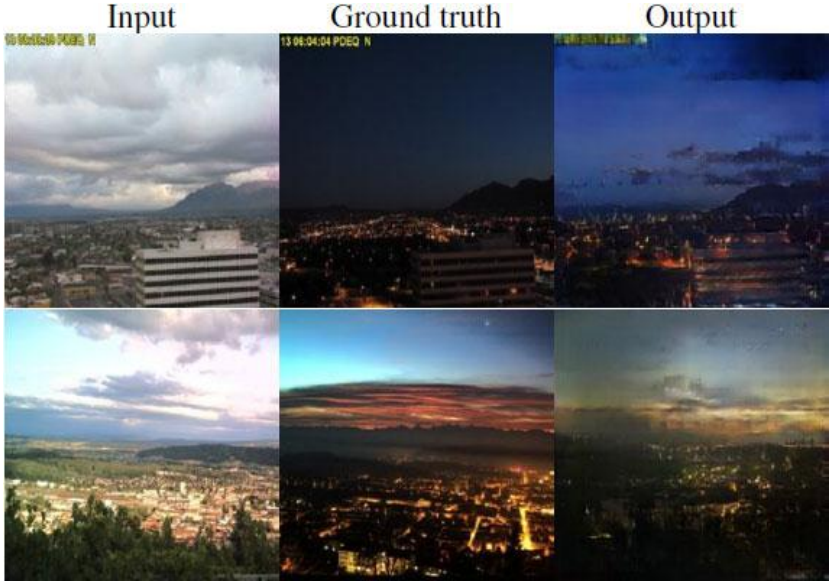
edges to image



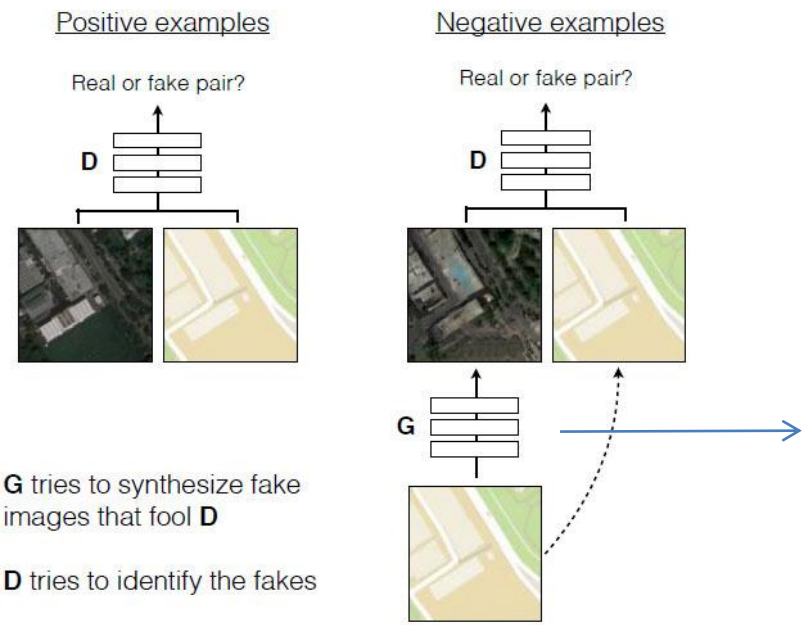
例子2： conditional-GAN

gray to color

day to night



例子2： conditional-GAN



$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{y\sim p_{data}(y)}[\log D(y)] + \mathbb{E}_{x\sim p_{data}(x), z\sim p_z(z)}[\log(1 - D(G(x,z))]$$

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G).$$

U-Net

https://arxiv.org/abs/1611.07004

例子4 写诗机器人

渺渺浮云外，孤灯次第休。
相思千里路，何处渡江流。

逶迤青嶂峻，山色夜窗悬。
寂寂潇湘水，迢迢月满川。

沧江千里外，渺渺白云高。
遥忆襄阳月，孤舟泛雪涛。

芙蓉池上曲，芳草满篮舆。
雨霁连天晓，清光映太虚。

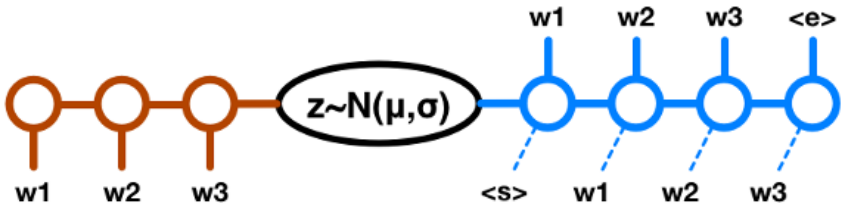
风起蒹葭苑，花开白帝城。
雪飘青峰下，月落洞庭春。

中原有弟子，国士自风流。
北阙南山外，京华万里游。

例子4 写诗机器人

数据：全唐诗

模型：VAE + beam search



例子4 写诗机器人

```
ray@Master:~/python/neural-poem-keras$ python sample.py
Using TensorFlow backend.
beam_size: 50,  drunk_level: 2.0000, sample from begin

tpl: ^我####，为####。秋####，香####。
candidates:
-----------------------------------------------------------------------
score: -95.715988: ^我去沧江客，为君湘水滨。秋风自日夕，香露多台风。
score: -95.728896: ^我去沧江客，为君湘水滨。秋风自鹭起，香露多坛春。
score: -95.750537: ^我去沧江客，为君无限情。秋风吹自发，香气向人间。
score: -97.683110: ^我住湘江路，为君心不知。秋风吹暮雨，香气照秋风。
score: -97.692954: ^我住湘江路，为君心不知。秋风吹暮雨，香气带秋风。
score: -97.786075: ^我住湘江路，为君心不知。秋风吹暮雨，香气向人间。
score: -98.031218: ^我为沧江客，为君湘水滨。秋风何处处，香气满衣裳。
```

```
score: -97.034842: ^我为沧江客，为君湘水滨。秋风吹万里，香气夕阳中。
score: -97.343641: ^我为沧江客，为君湘水滨。秋风吹不尽，香气夕阳中。
score: -97.451114: ^我为沧江客，为君湘水滨。秋风吹不尽，香气夕阳明。
score: -97.468288: ^我为沧江客，为君湘水滨。秋风吹万里，香气夕阳边。
```

```
score: -97.345348: ^我为沧江客，为君湘水滨。秋风吹万里，香气满尘埃。
score: -97.693209: ^我为沧江客，为君湘水滨。秋风吹万里，香气满洛城。
score: -97.701529: ^我为沧江客，为君湘水滨。秋风吹万里，香气满长安。
score: -97.768862: ^我为沧江客，为君湘水滨。秋风吹万里，香气满沧洲。
score: -97.919264: ^我为沧江客，为君湘水滨。秋风吹万里，香马夕阳中。
score: -97.952546: ^我为沧江客，为君湘水滨。秋风吹万里，香气满平阳。
```

我为沧江客，
为君湘水滨。
秋风吹万里，
香气满长安。

同样的模型，可以用于生成音乐

原曲

生成

一点感受

深度学习的很多效果出人意料（more is different）

适宜辅助，而非决策

适宜专用，而非通用

AI （Artificial Intelligence）

↓

IA（Intelligence Augment）

东南有坐网，至今心未易。
为报君王博，不负平生学。

落日洒蛛网，悠悠江水大。
浮生不容易，为君识白牛。

谢谢！

THANKS