机器学习

Machine learning

第十章 神经网络与深度学习(4)

Neural Network and Deep Learning

授课人：周晓飞

zhouxiaofei@iie.ac.cn

2020-12-26

第十章 神经网络与深度学习

10.1 概述

10.2 多层感知机

10.3 卷积网络

10.4 Recurrent 网络

10.5 前沿概述

深度学习、生成对抗学习、强化学习、知识图谱

深层结构

神经网络 + 深层结构 + 优化 + 计算资源 + 人工智能应用

基本的深度神经网络结构

- Feedforward neural network, also called Multilayer Perceptron (MLP).
- Recurrent neural network.

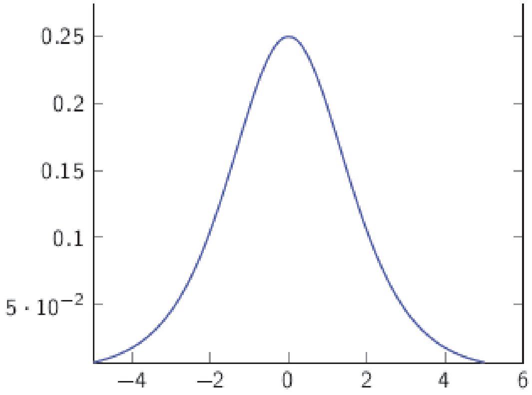
梯度消失

$$\delta^{(l)} = f_l'(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)},$$

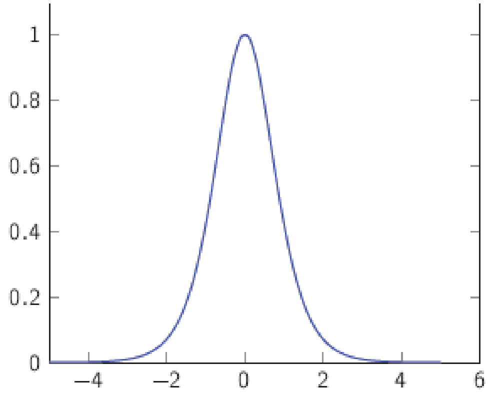
When we use sigmoid function, such as logistic $\sigma(x)$ and tanh,

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \in [0, 0.25]$$

$$\tanh'(x) = 1 - (\tanh(x))^2 \in [0, 1].$$



(e) logistic



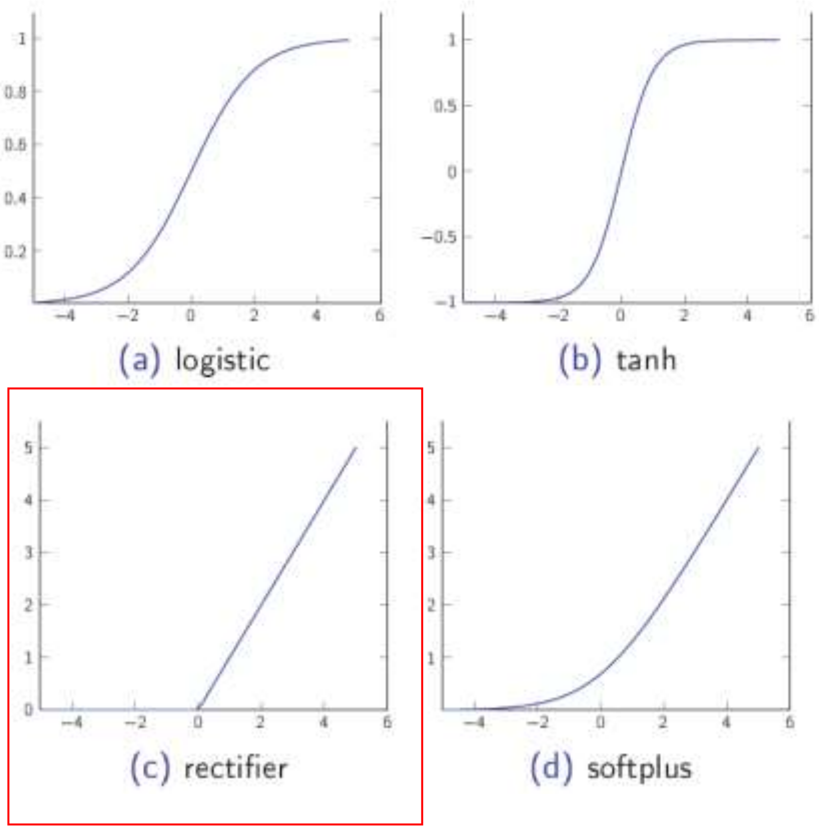
(f) tanh

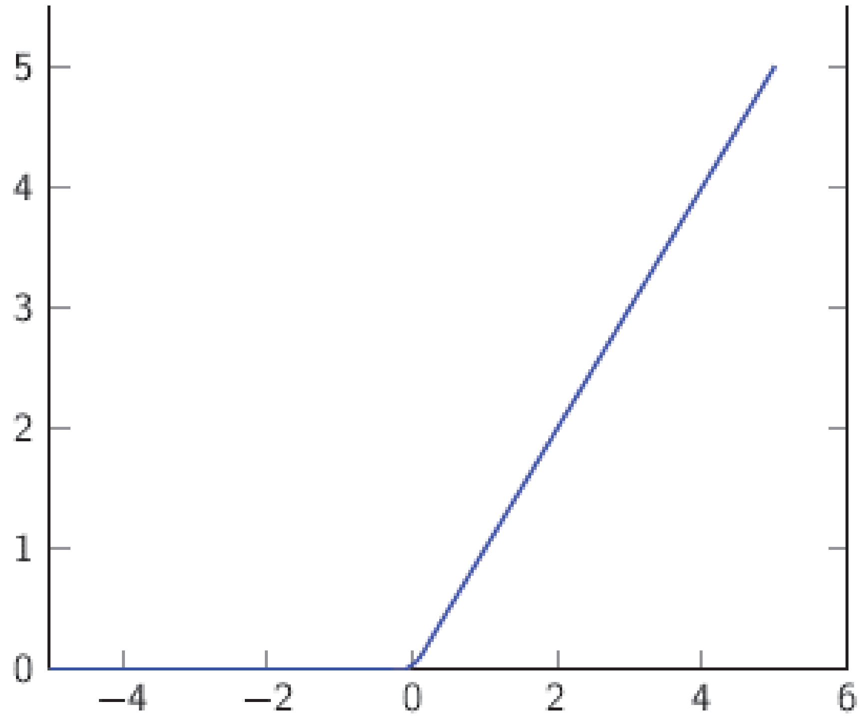
梯度消失

解决梯度消失

- **前馈网络：**
  自编码、ReLU 激活函数
- **Recurrent 网络：**
  二次优化、非线性逐次状态估计、ReLU 激活函数

ReLU 函数

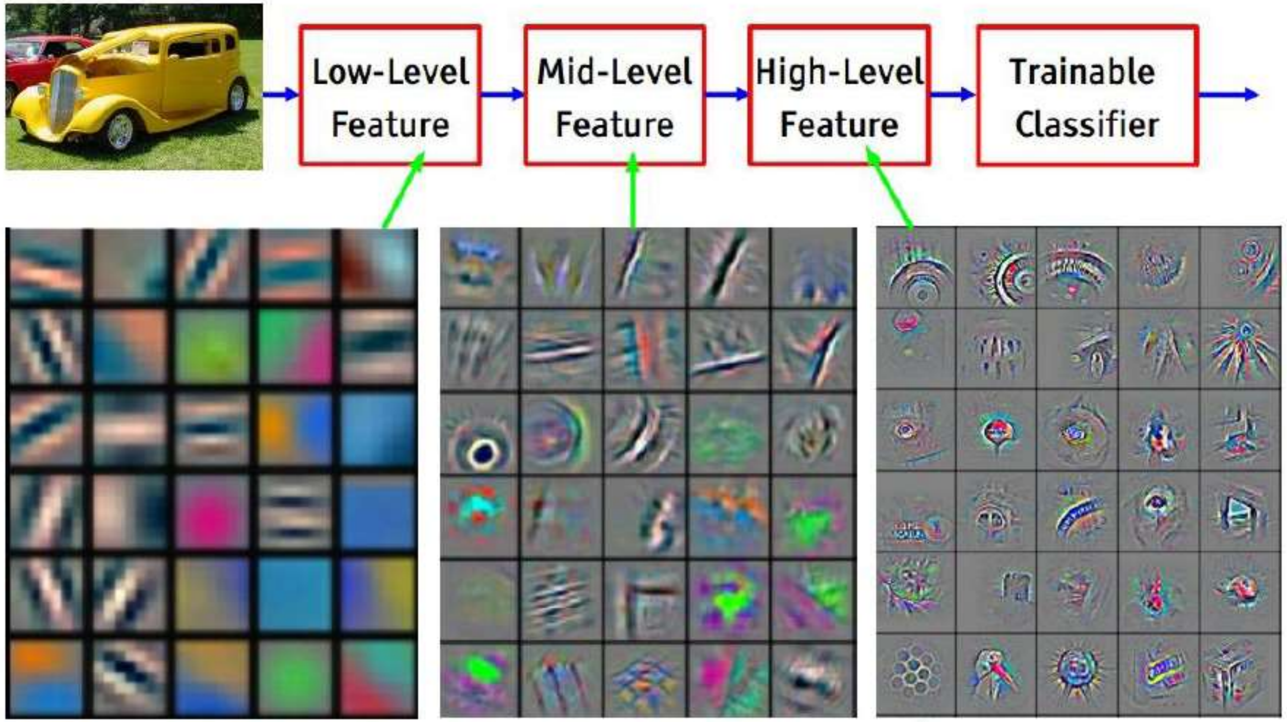




L= max(0, x)

视觉识别

Image Feature and Recognition

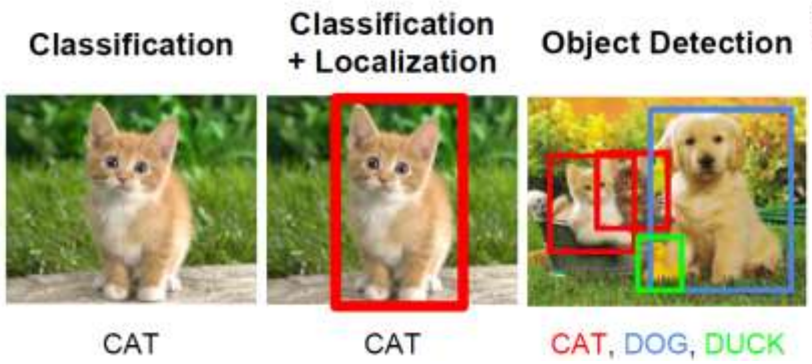


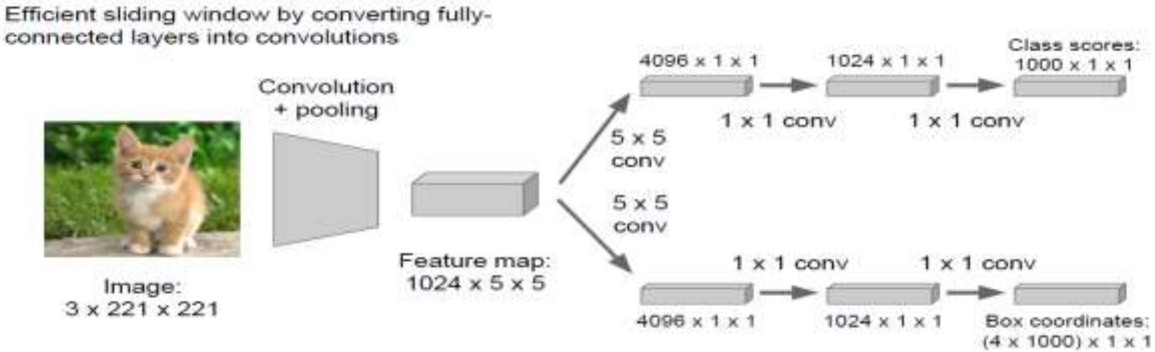
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

深度学习

视觉识别

Image Detection





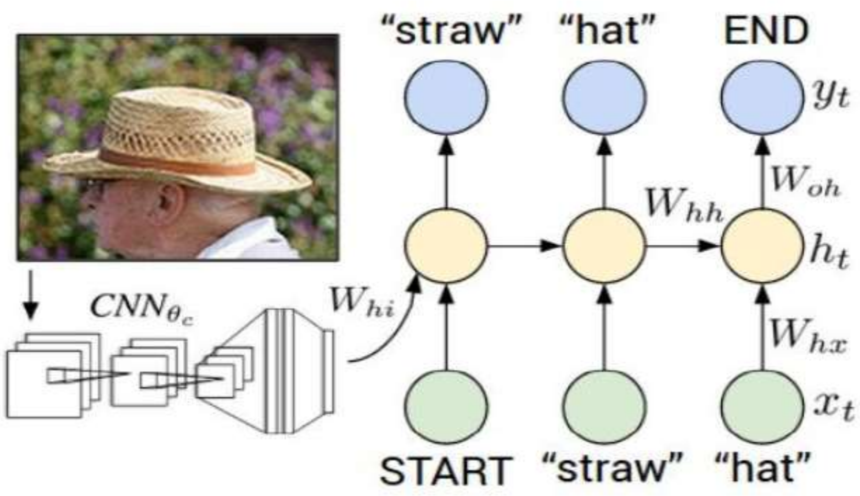
Idea #1: Localization as Regression

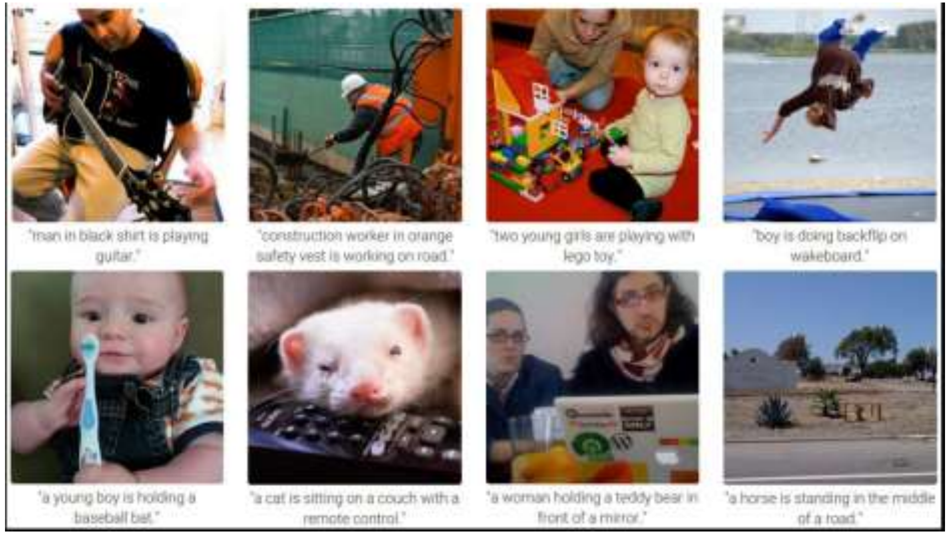
Idea #2: Sliding Window

深度学习

视觉识别

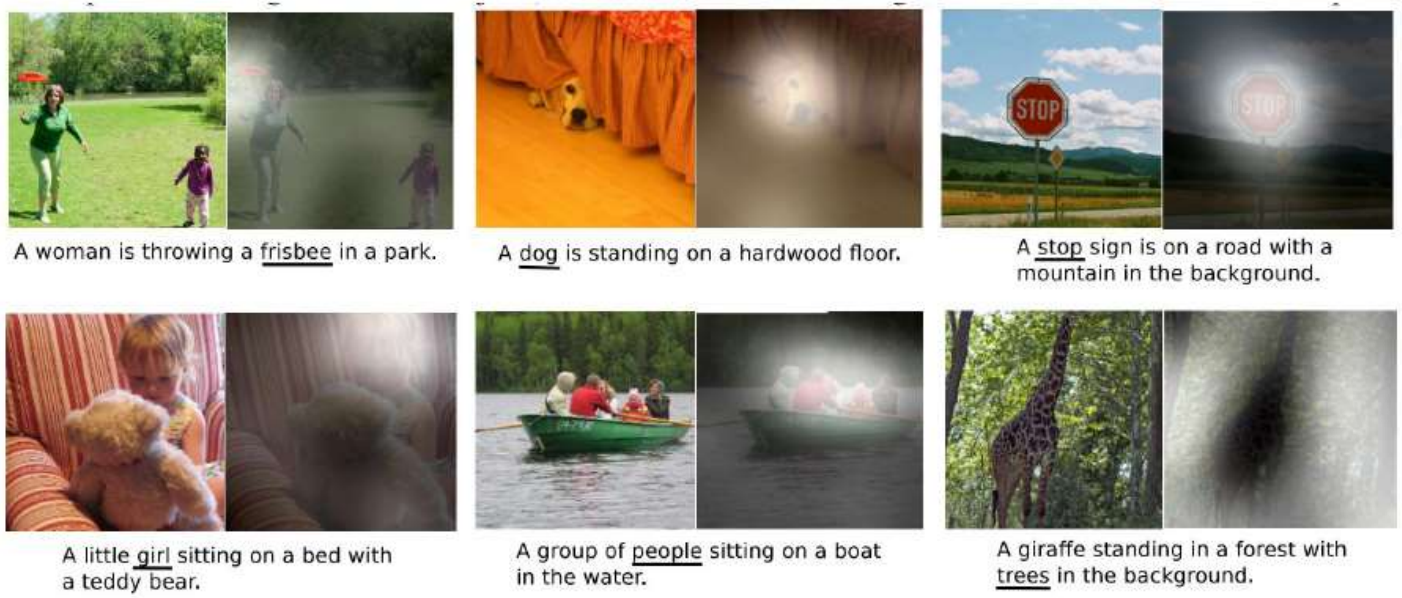
Image Captioning





视觉识别

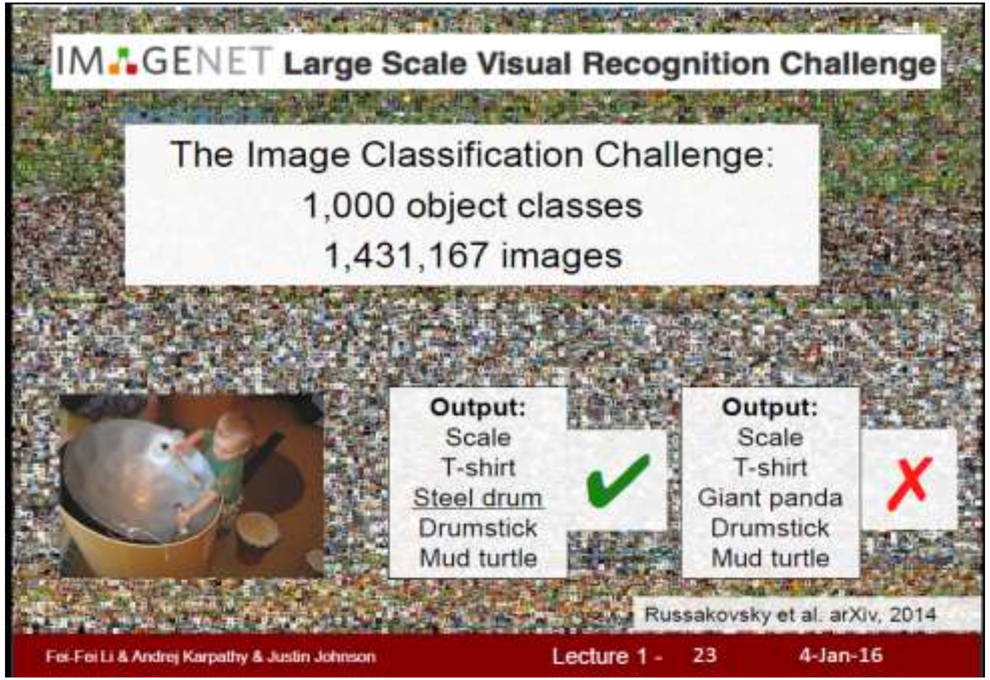
Attention for Captioning



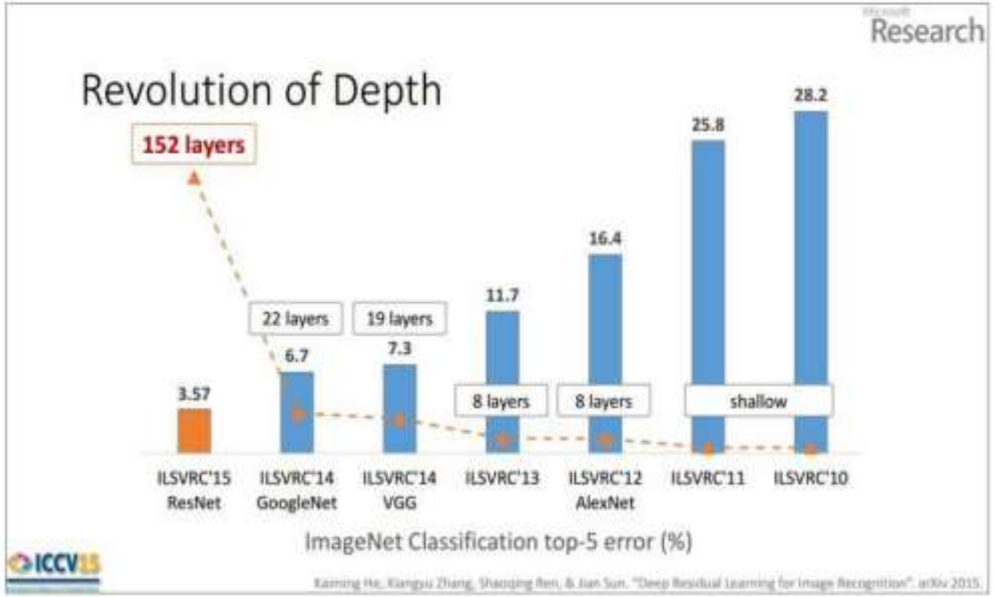
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

视觉识别

ImageNet Completion



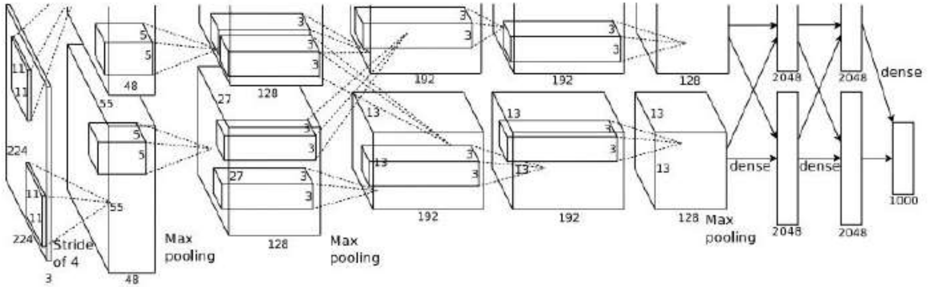
网络结构 ： CNN+全连接



深度学习

Case Study: AlexNet

[Krizhevsky et al. 2012]



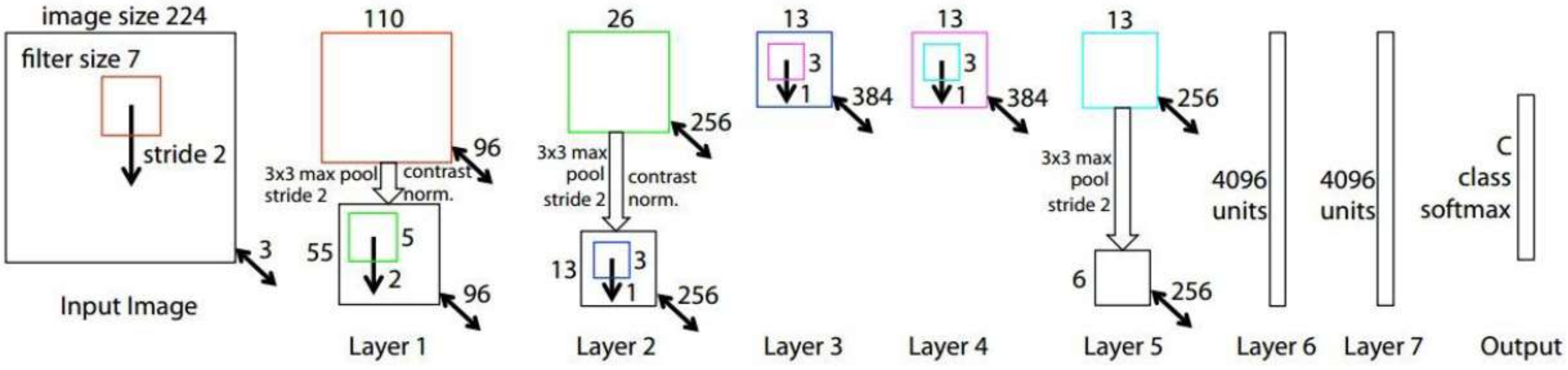
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

2012 ILSVRC winner （top 5 error of 16% compared to runner-up with 26% error）
共有 8 层，其中前 5 层卷积层，后边 3 层全连接层
最后的一层的输出是具有 1000 个输出的 softmax

Case Study: ZFNet

[Zeiler and Fergus, 2013]



AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 15.4% -> 14.8%

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

ConvNet Configuration			
B	C	D	
13 weight layers	16 weight layers	16 weight layers	19
put (224 × 224 RGB image			
conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Note:

Most memory is in early CONV

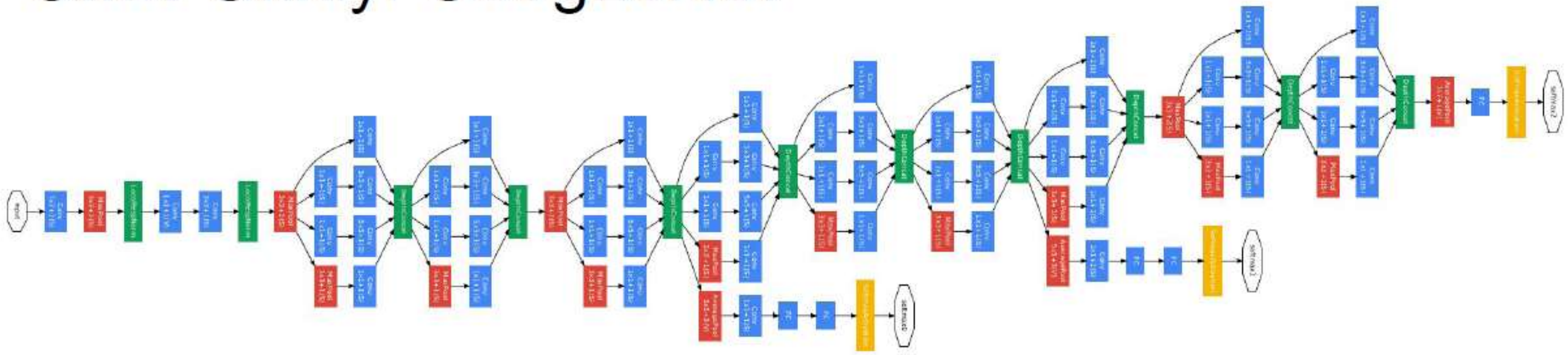
Most params are in late FC

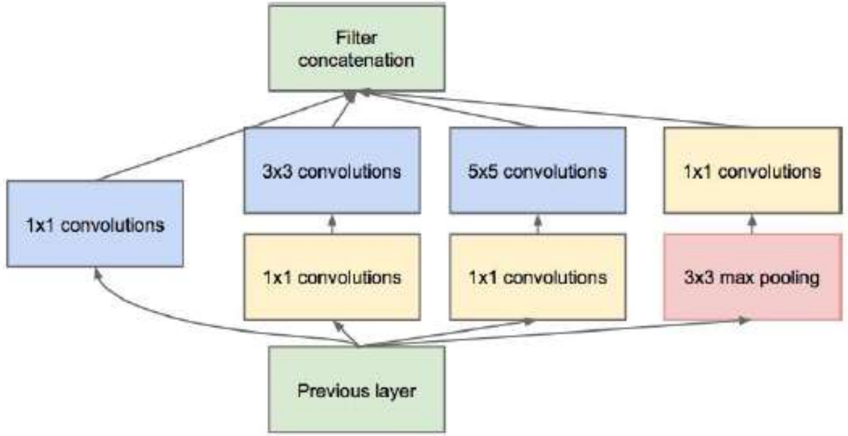
TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

深度学习

Case Study: GoogLeNet

[Szegedy et al., 2014]





Inception module

ILSVRC 2014 winner (6.7% top 5 error)

深度学习

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fun features:

- Only 5 million params!
(Removes FC layers completely)

Compared to AlexNet:
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

Microsoft Research

MSRA @ ILSVRC & COCO 2015 Competitions

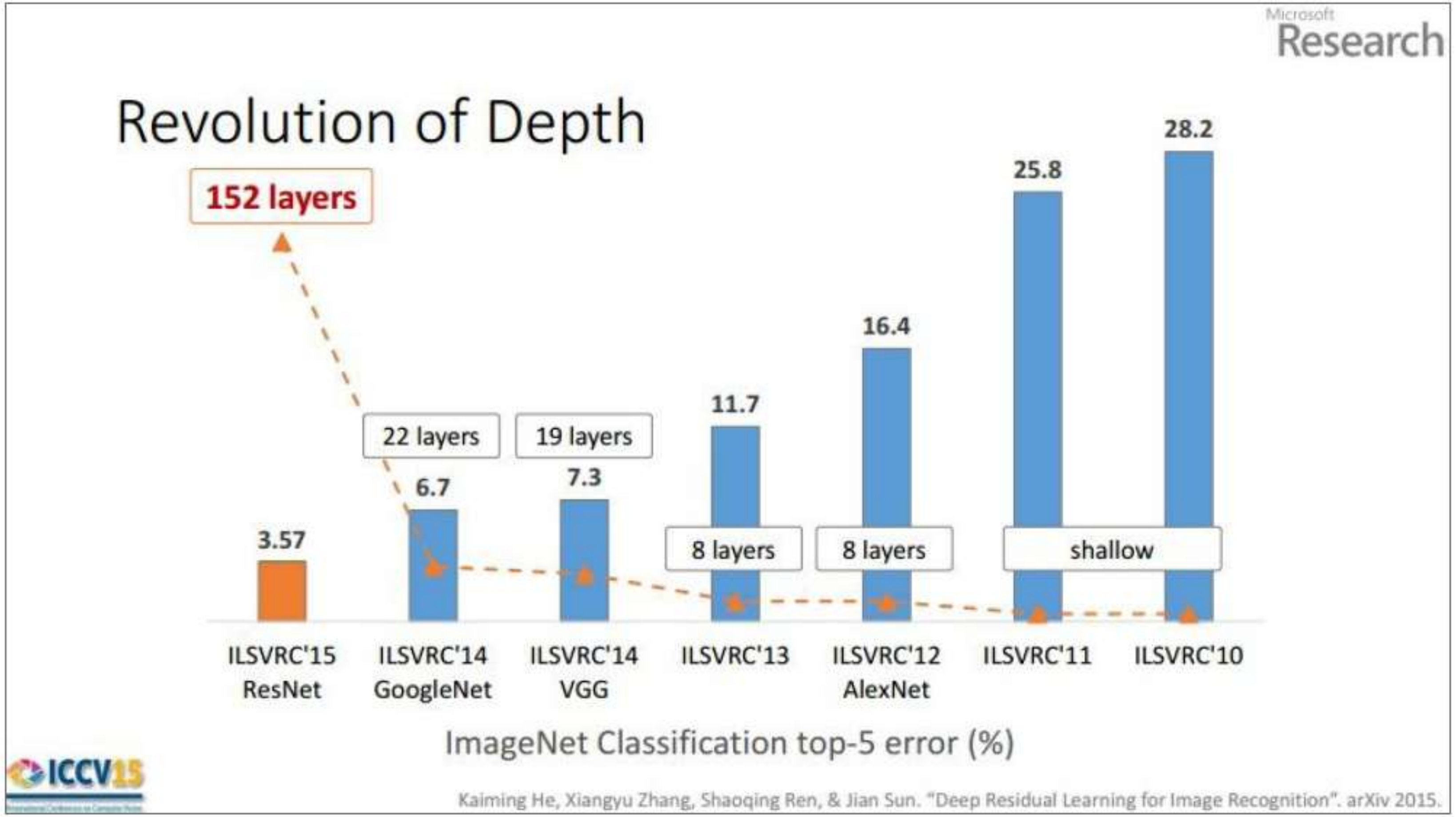
- **1st places in all five main tracks**
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

ICCV15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

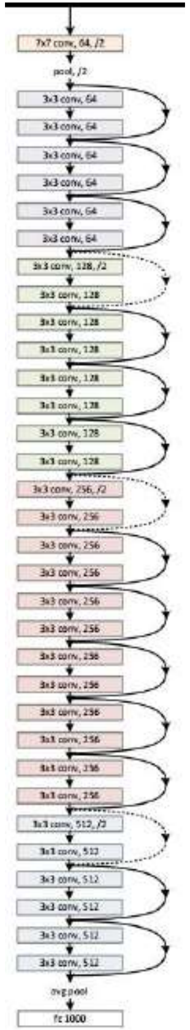
深度学习



Microsoft Research
Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)
ICCV15
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

深度学习

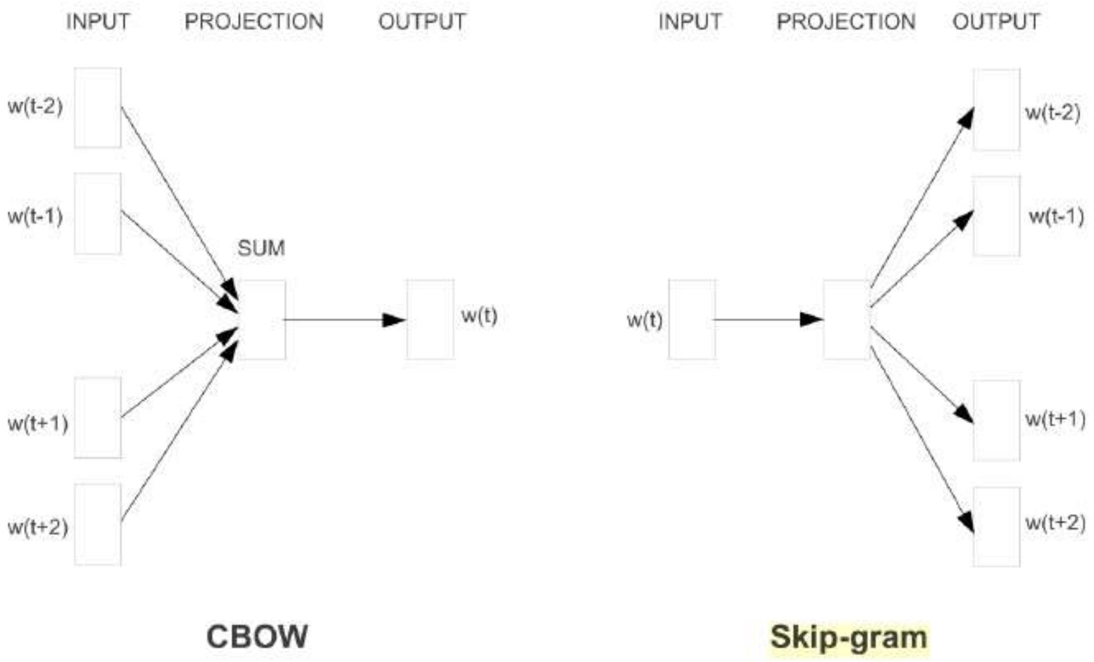
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



自然语言处理

Representation

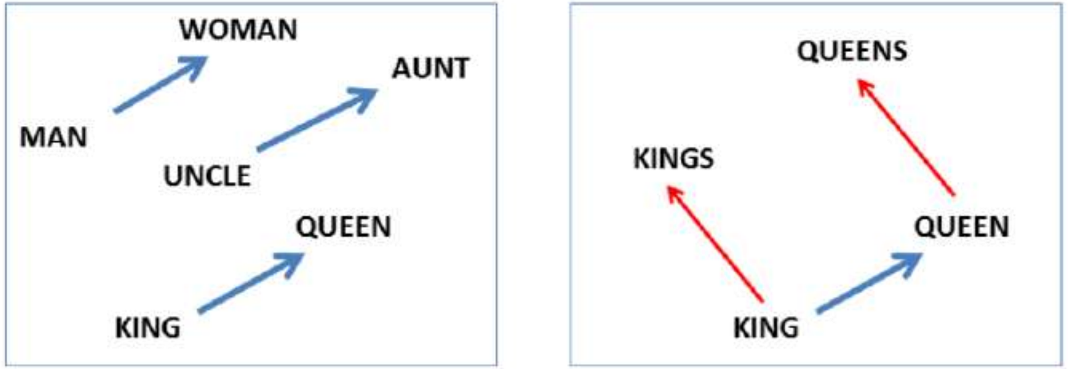
Word2Vector



自然语言处理

Representation

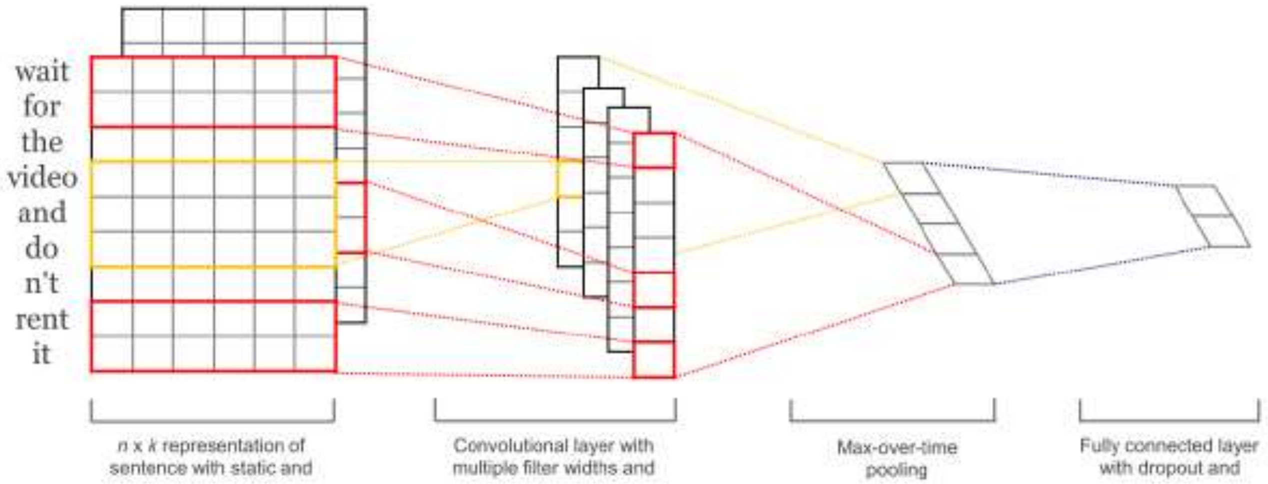
Knowledge Embedding



自然语言处理

Representation

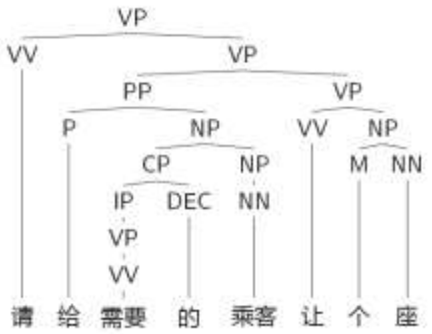
Sentence Modeling

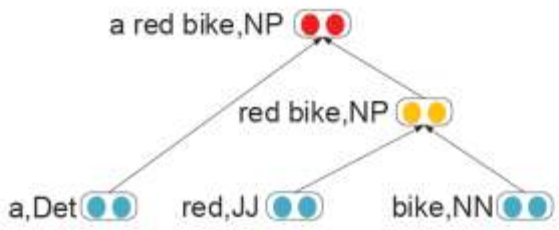


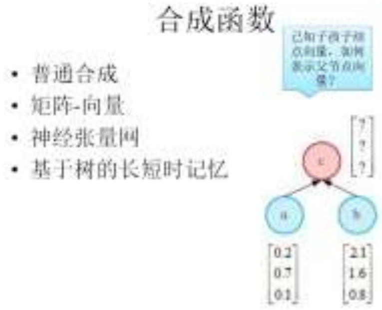
自然语言处理

Representation

Recursive Neural Network(结构的递归神经网络)

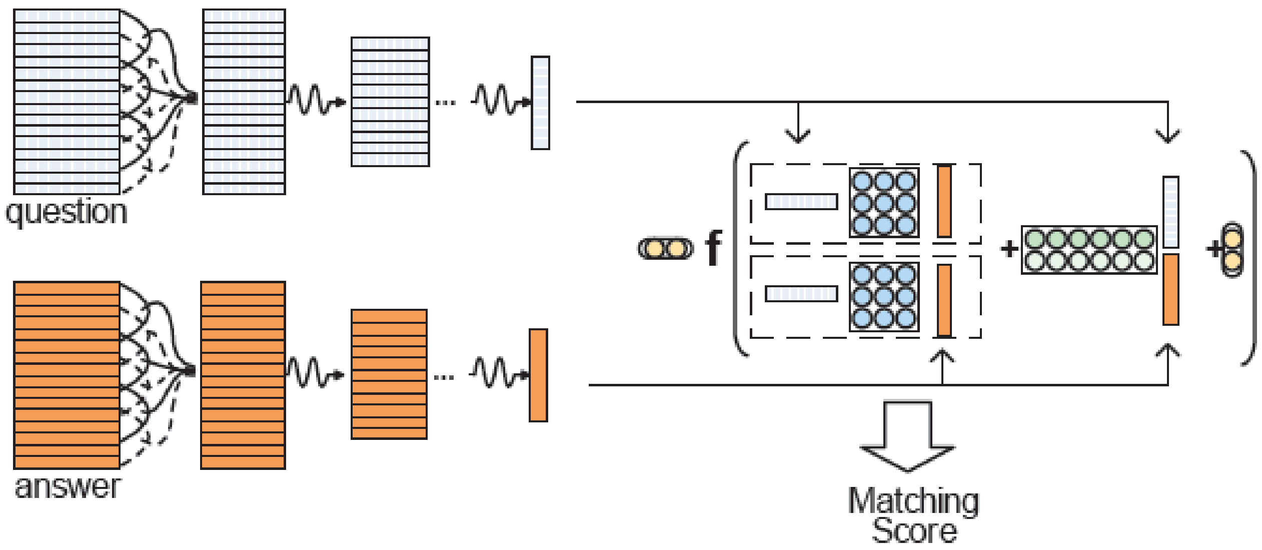






自然语言处理

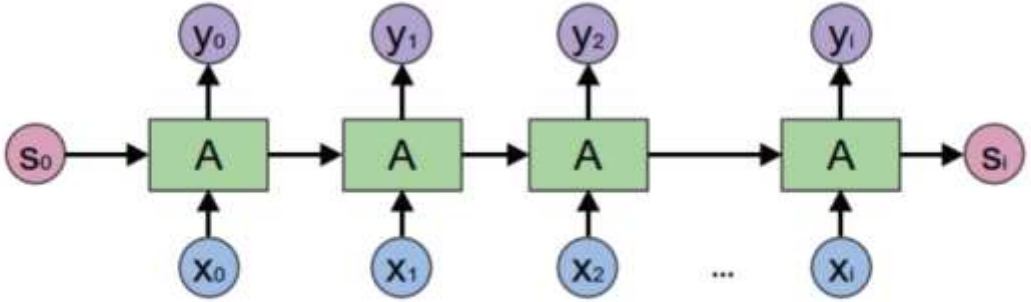
Question Answering



自然语言处理

Sequence to Sequence Modeling

- 输入：任意长度的序列
- 输出：任意长度的序列
- 应用：语音识别、中文分词、词性标注

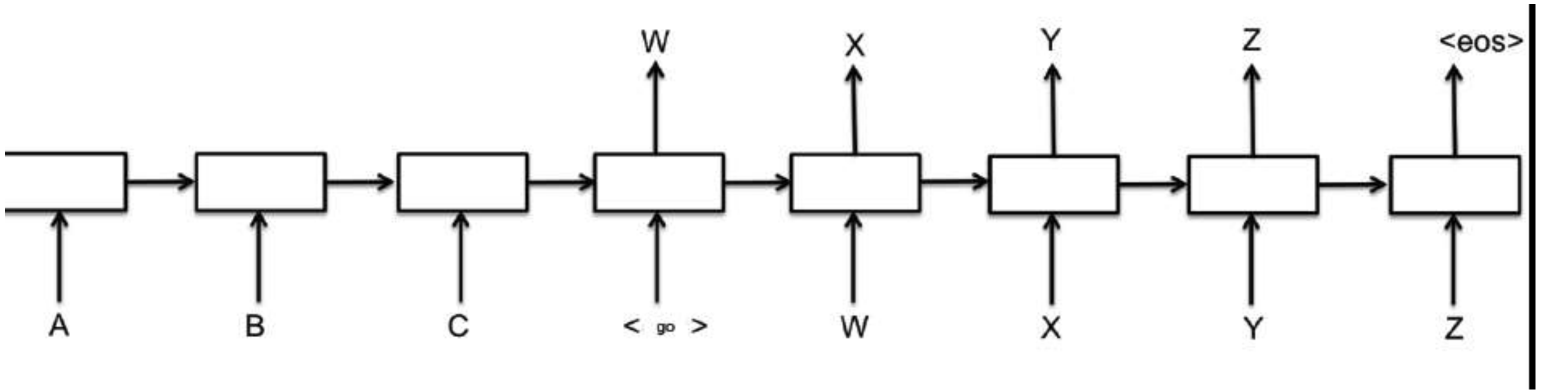


自然语言处理

Sequence to Sequence Modeling

- Encoder-Decoder

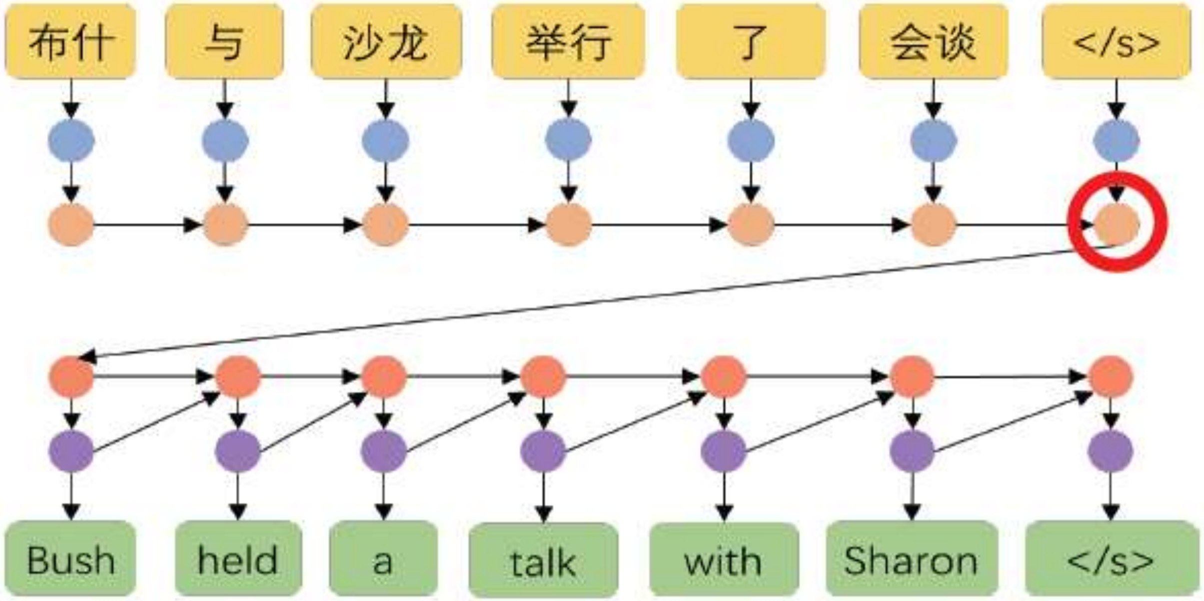
  自然语言中的典型应用：机器翻译、文档摘要、对话系统、自动问答。



自然语言处理

Sequence to Sequence Modeling

- 机器翻译例子



自然语言处理

Sequence to Sequence Modeling

- Attention Model

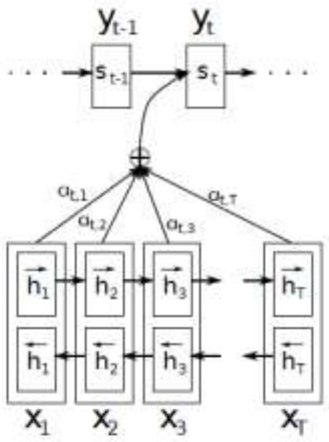


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j),$$

-

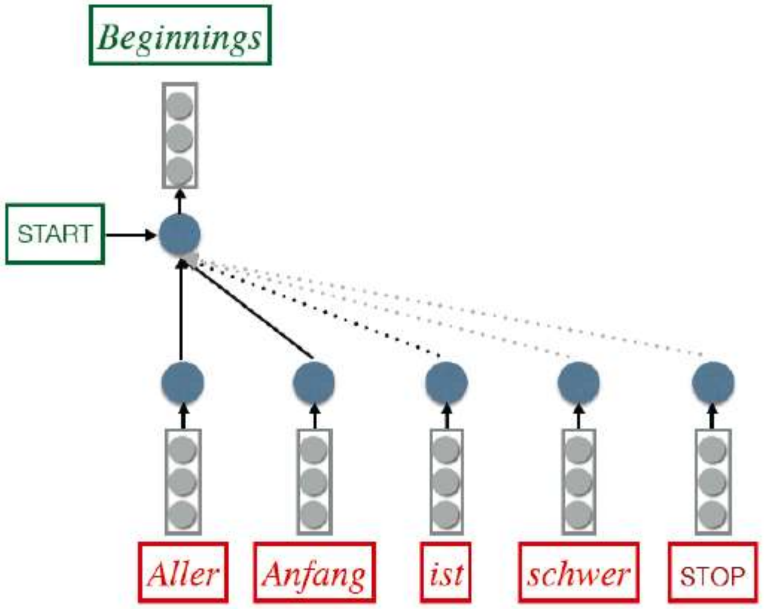
（Bahdanau，2014）

深度学习

自然语言处理

Sequence to Sequence Modeling

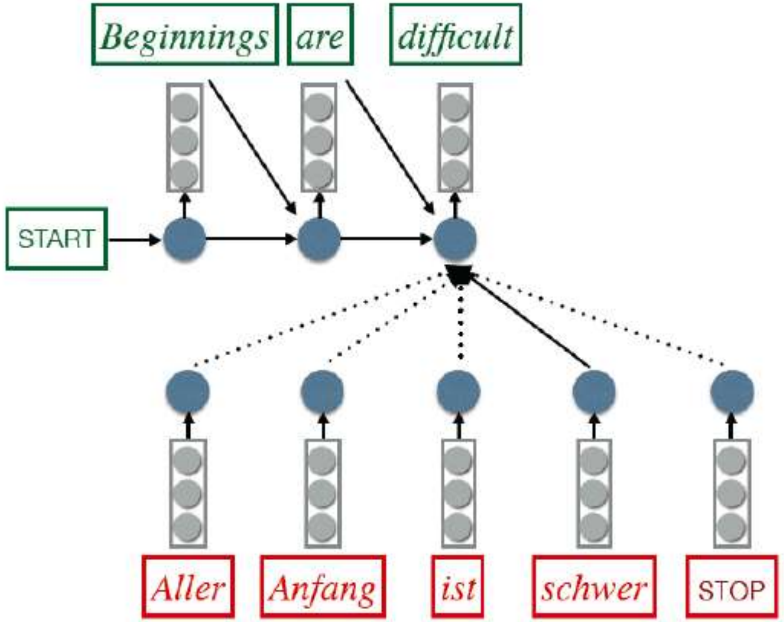
- It seems too much to expect one vector of input c can do everything.
- During decoding, dynamically pay attention to different parts of the input.



自然语言处理

Sequence to Sequence Modeling

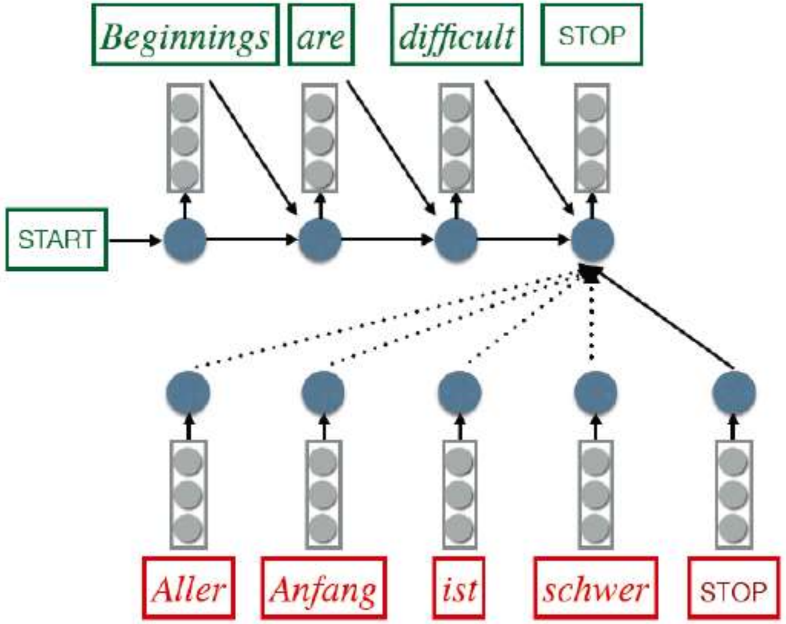
- It seems too much to expect one vector of input c can do everything.
- During decoding, dynamically pay attention to different parts of the input.



自然语言处理

Sequence to Sequence Modeling

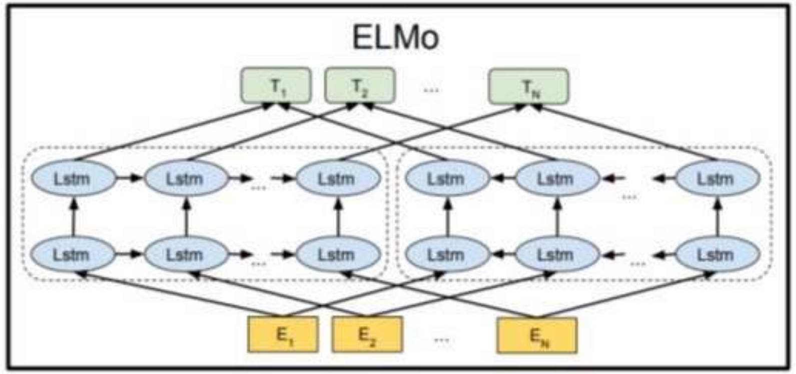
- It seems too much to expect one vector of input c can do everything.
- During decoding, dynamically pay attention to different parts of the input.



自然语言处理

ELMo

- **Pre-train bidirectional language models**



$$\sum_{k=1}^{N}\Big(\log p(t_k \mid t_1,\ldots,t_{k-1};\Theta_x,\overrightarrow{\Theta}_{LSTM},\Theta_s)$$
$$+\log p(t_k \mid t_{k+1},\ldots,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s)\Big)$$

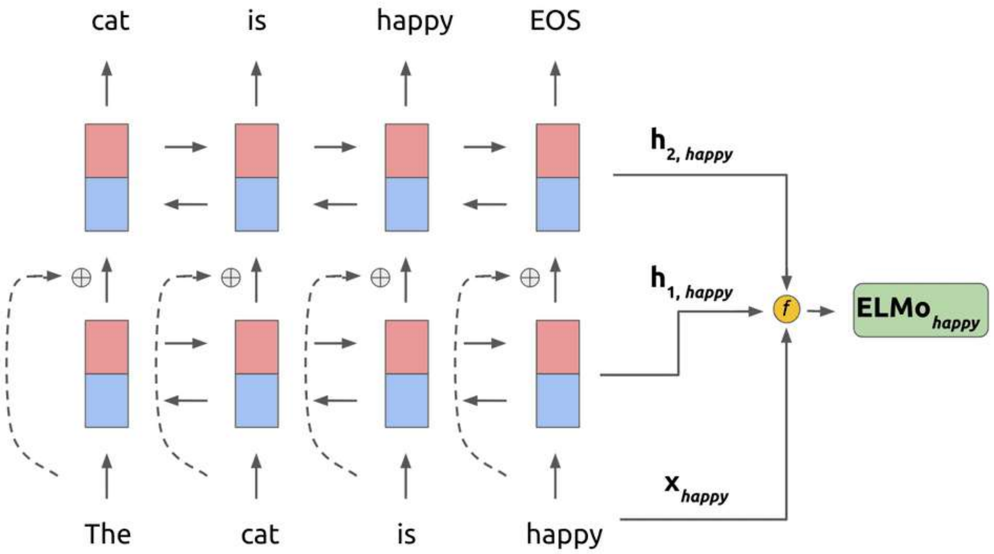
- **ELMo contextualized word representations**

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j=1,\ldots,L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j=0,\ldots,L\}, \end{aligned}$$

自然语言处理

ELMo

- **ELMo for supervised NLP tasks**



$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

自然语言处理

Transformer

- Positional Encoding
- Multi-Head Attention

  Encoder: Over all words (self-attention)

  Decoder: (1) Over previously generated words (self-attention)

  (2) Between outputs of encoder and (1)

  (not self-attention)

- Residual Connection

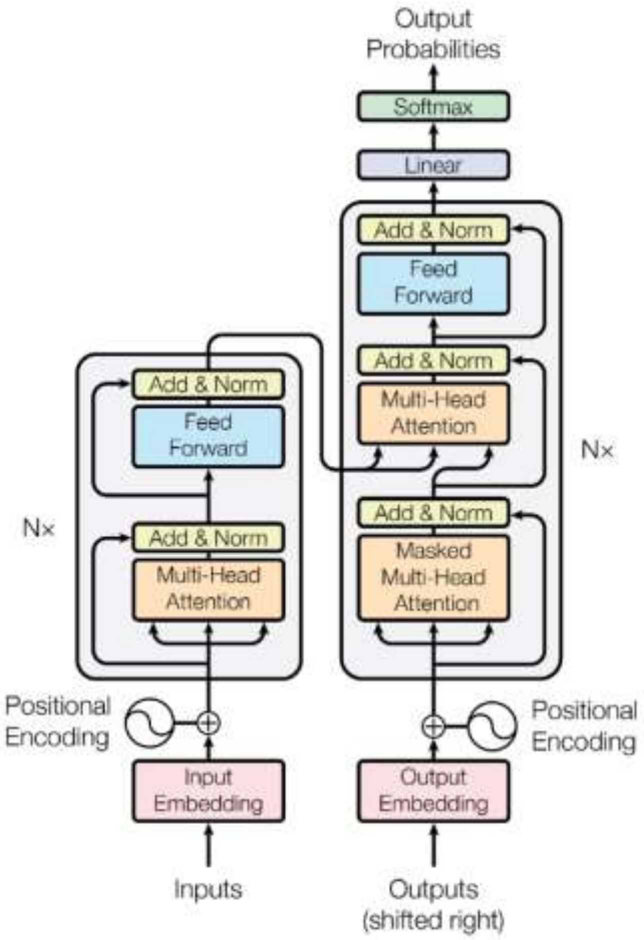
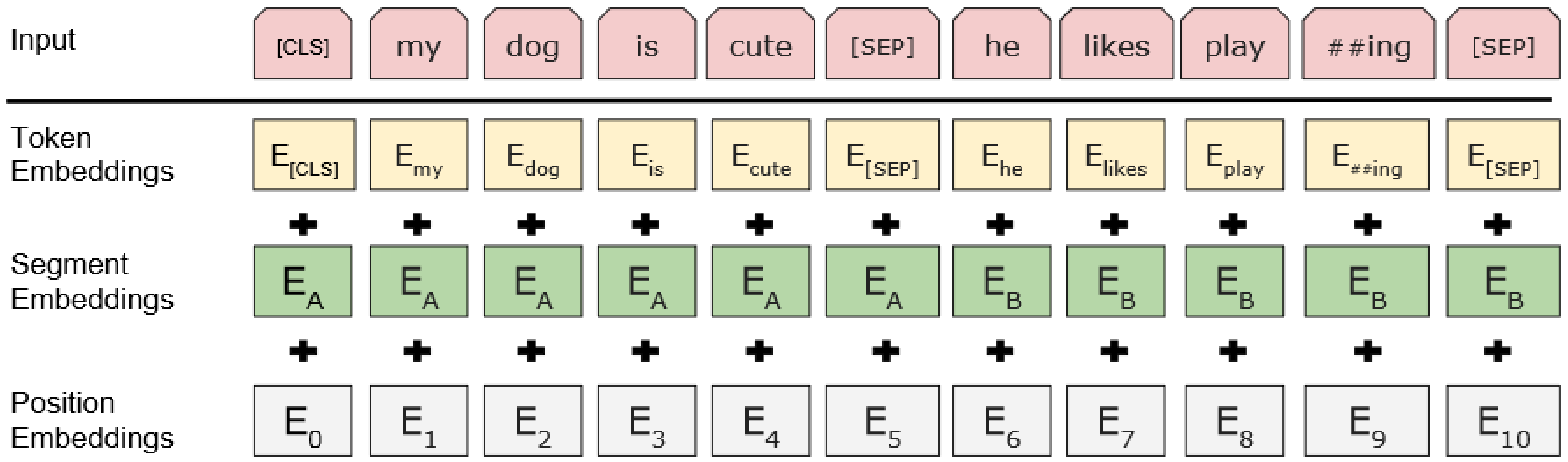


Figure 1: The Transformer - model architecture.

自然语言处理

BERT

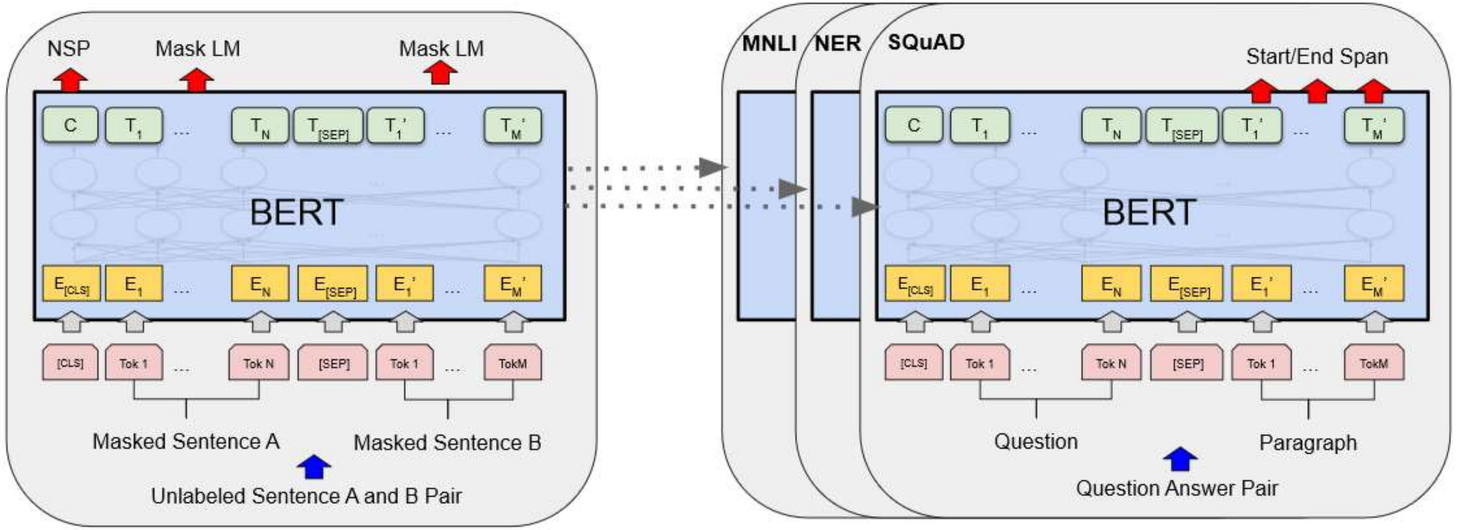
- **Model Architecture：Transformer Encoder**
- **Input Embedding：**



自然语言处理

BERT

- **Pre-training BERT（Masked LM Task & Next Sentence Prediction Task）**

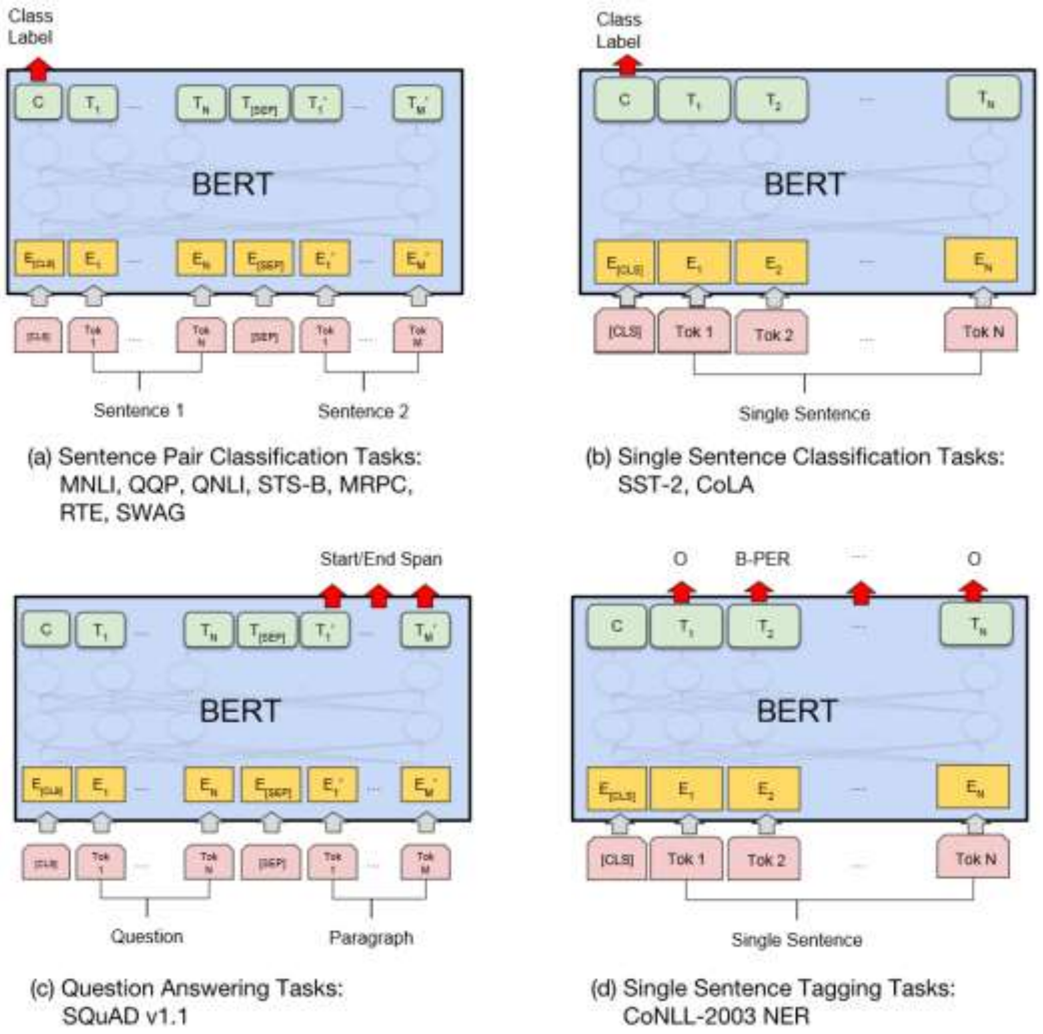


Pre-training　　　　Fine-Tuning

自然语言处理

BERT

- **Fine-tuning BERT on different tasks**



深度学习平台

Caffe

http://caffe.berkeleyvision.org

- From U.C. Berkeley
- Written in C++
- Has Python and MATLAB bindings
- Good for training or finetuning feedforward models

Torch

http://torch.ch

- From NYU + IDIAP
- Written in C and Lua
- Used a lot a Facebook, DeepMind

深度学习平台

Theano

http://deeplearning.net/software/theano/

- From Yoshua Bengio’s group at University of Montreal 蒙特利尔（加拿大）
- Embracing computation graphs, symbolic computation
- High-level wrappers: Keras, Lasagne

TensorFlow

https://www.tensorflow.org

- From Google
- Very similar to Theano - all about computation graphs
- Easy visualizations (TensorBoard)
- Multi-GPU and multi-node training

第十章 神经网络与深度学习

生成对抗学习

生成对抗模型原理

生成器（Generator）

尽可能去学习真实样本的分布，迷惑鉴别器。

鉴别器（Discriminator）

尽可能的正确判断输入数据是来自真实数据还是来自生成器。

损失函数

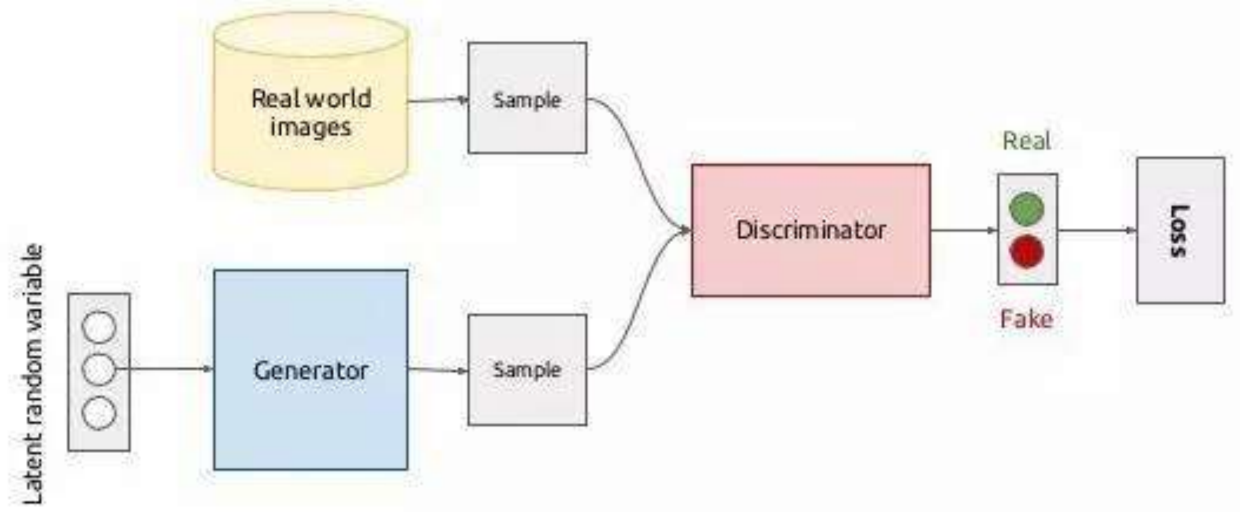
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))]$$

生成对抗学习

训练过程

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

生成器与鉴别器交替训练，互相提升各自的生成能力和鉴别能力，最终寻找二者之间的一个纳什均衡。



生成对抗学习

训练过程

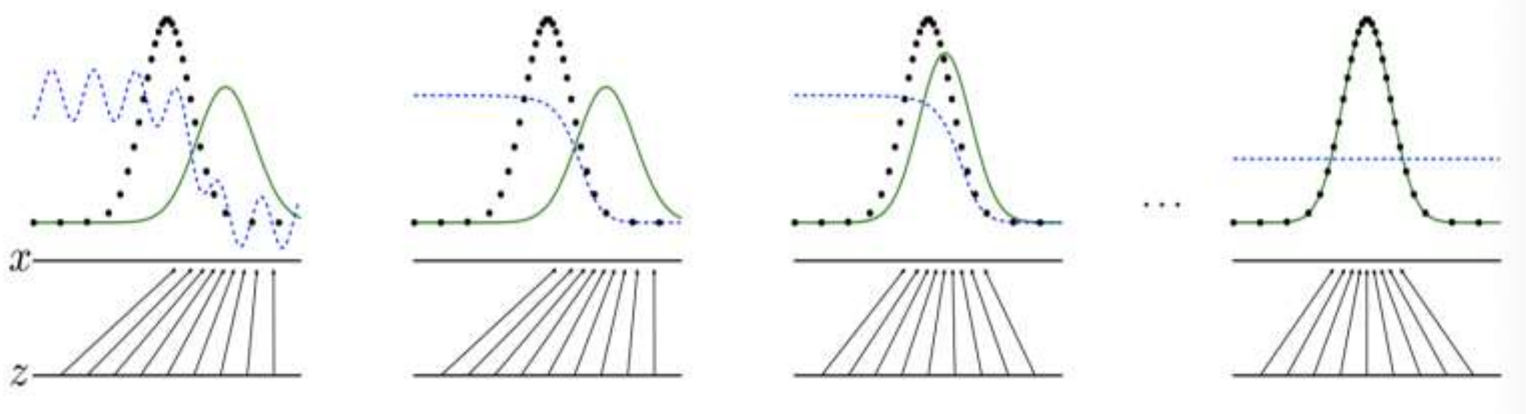
蓝色鉴别器输出的分布，黑色真实数据分布，绿色生成器输出分布

（a）初始化状态

（b）固定 G，训练 D：导致真实样本聚集区域 D 的输出大，生成器输出集中区域 D 的输出小。

（c）固定 D，训练 G；导致 G 的输出集中区域向 D 值高的区域稍微移动。

（d）最终的纳什均衡状态。



(a) (b) (c) (d)

算法流程

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

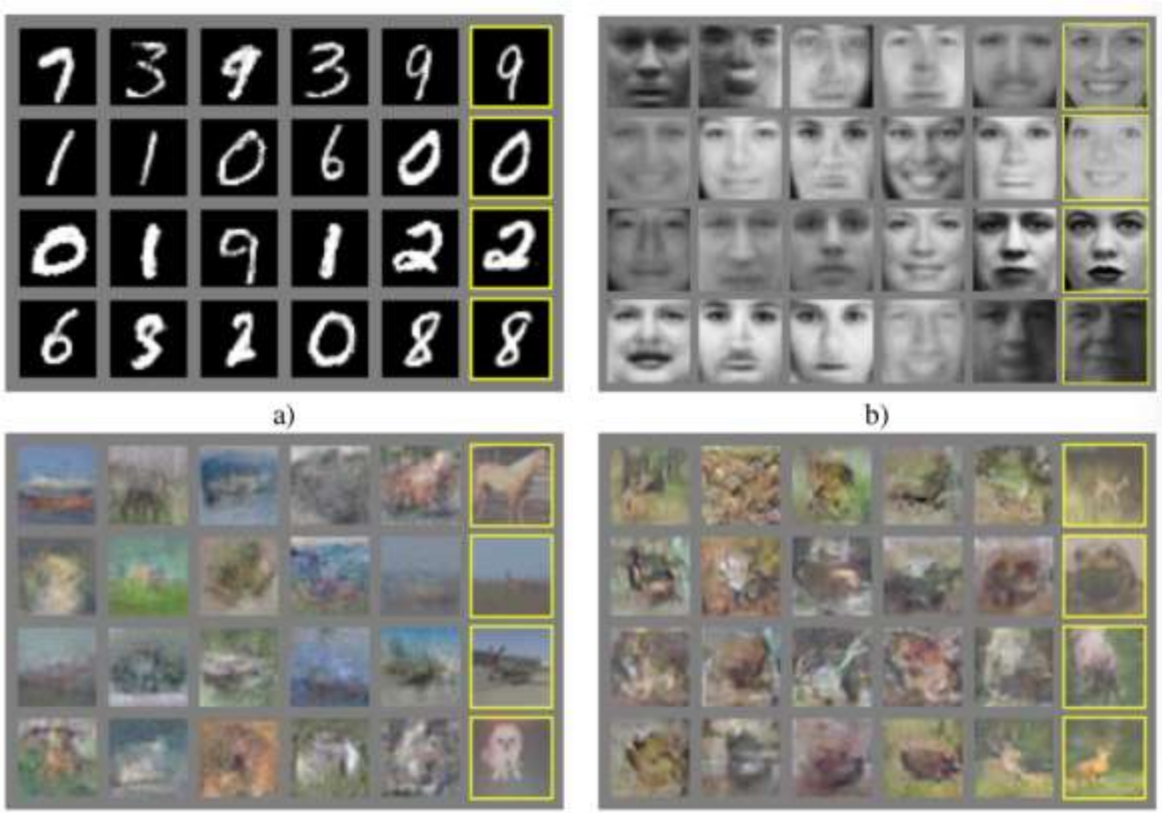
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

生成对抗学习

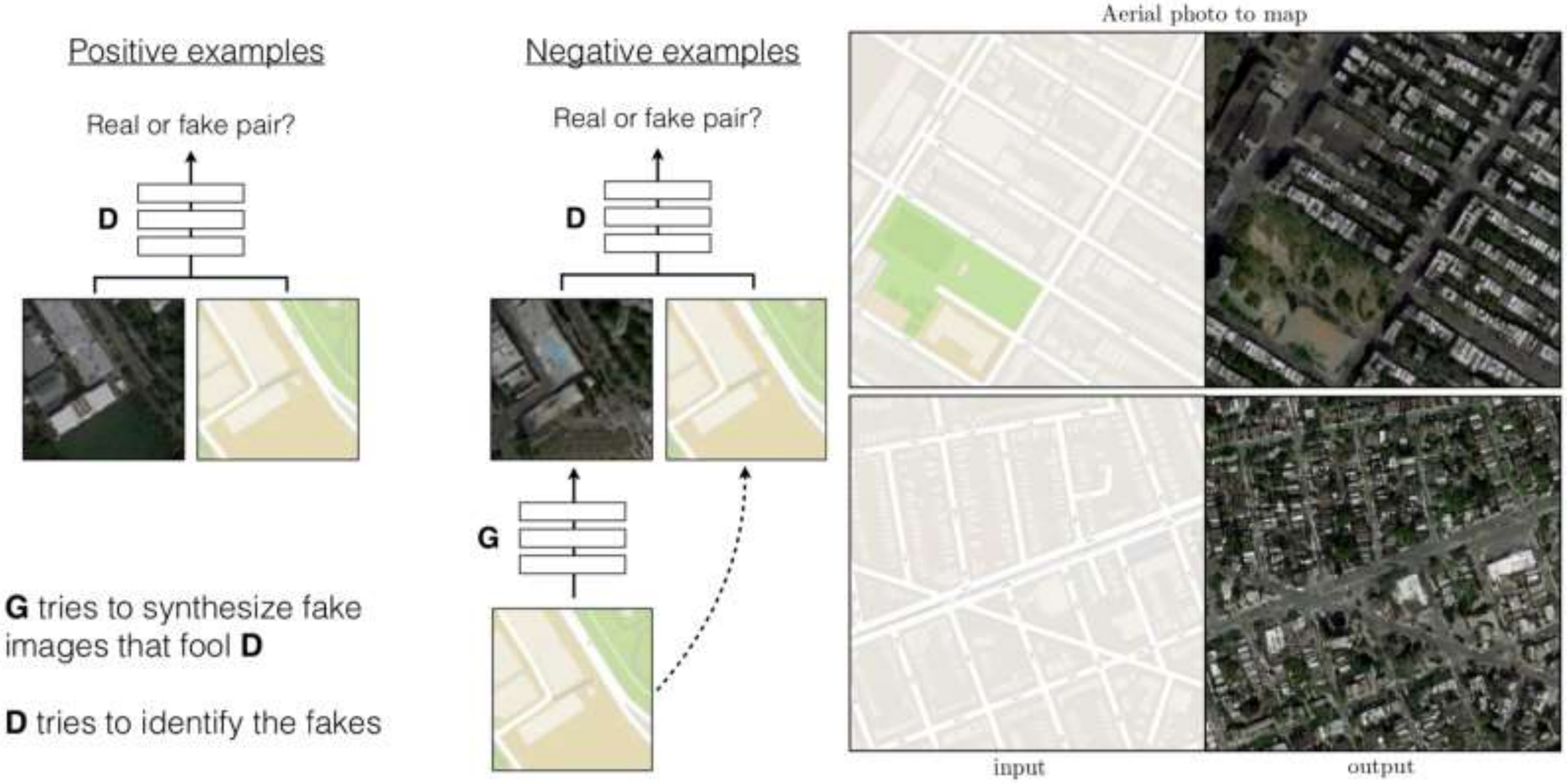
生成效果



生成对抗学习

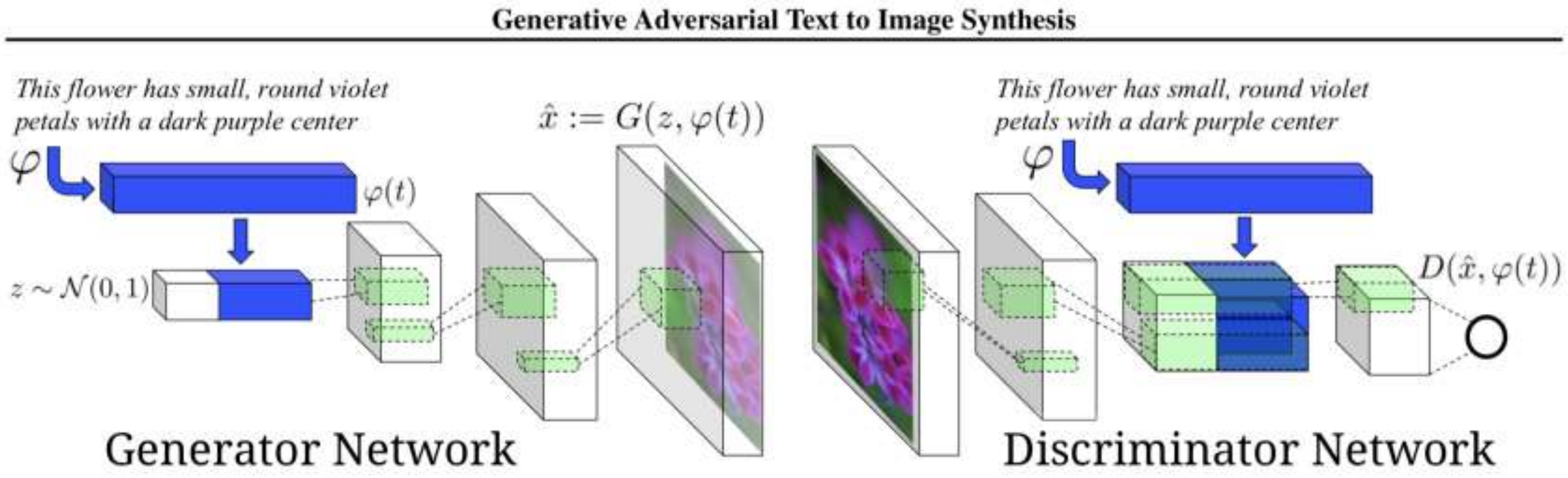
GAN 应用

- **Image to Image**



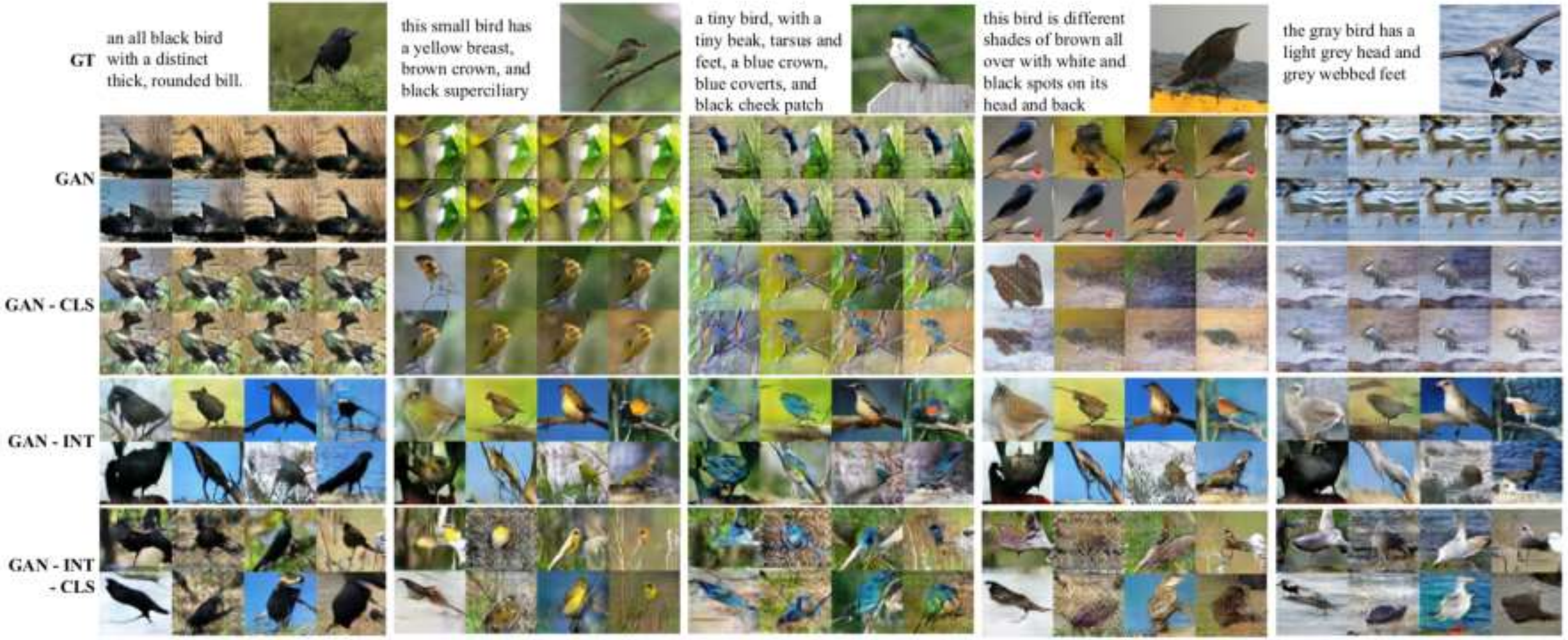
GAN 应用

- **Text to Image**



GAN 应用

- **Text to Image**



生成对抗学习

GAN 应用

- **Super-Resolution**

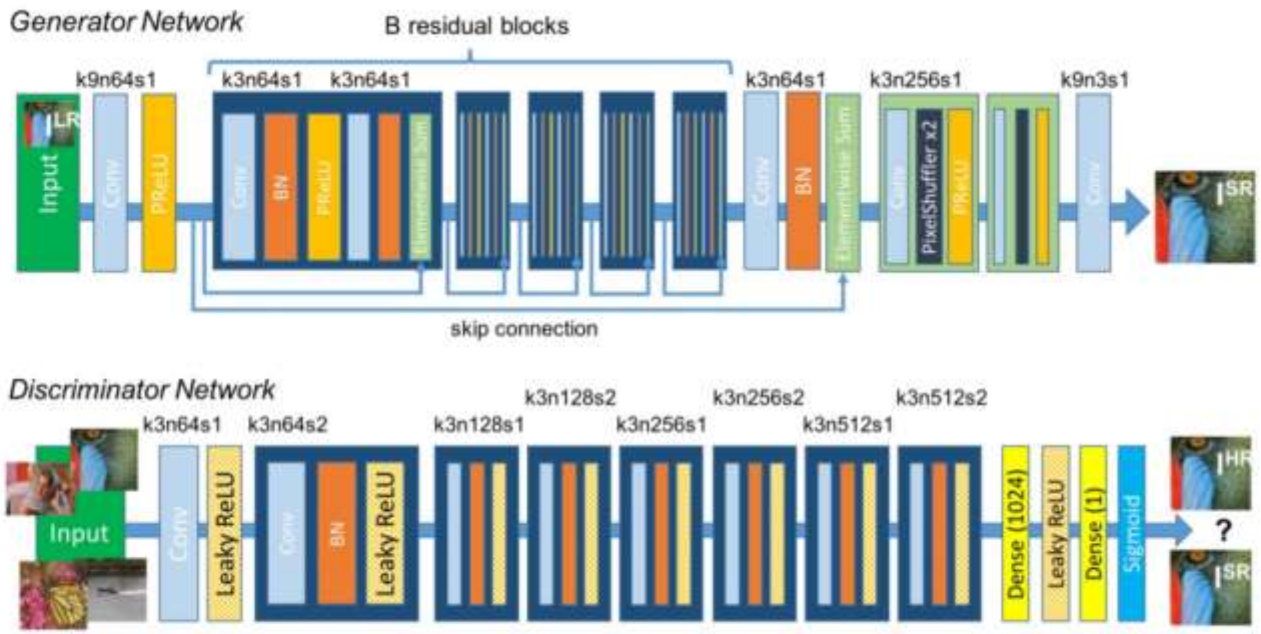


Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

GAN 应用

- **Super-Resolution**

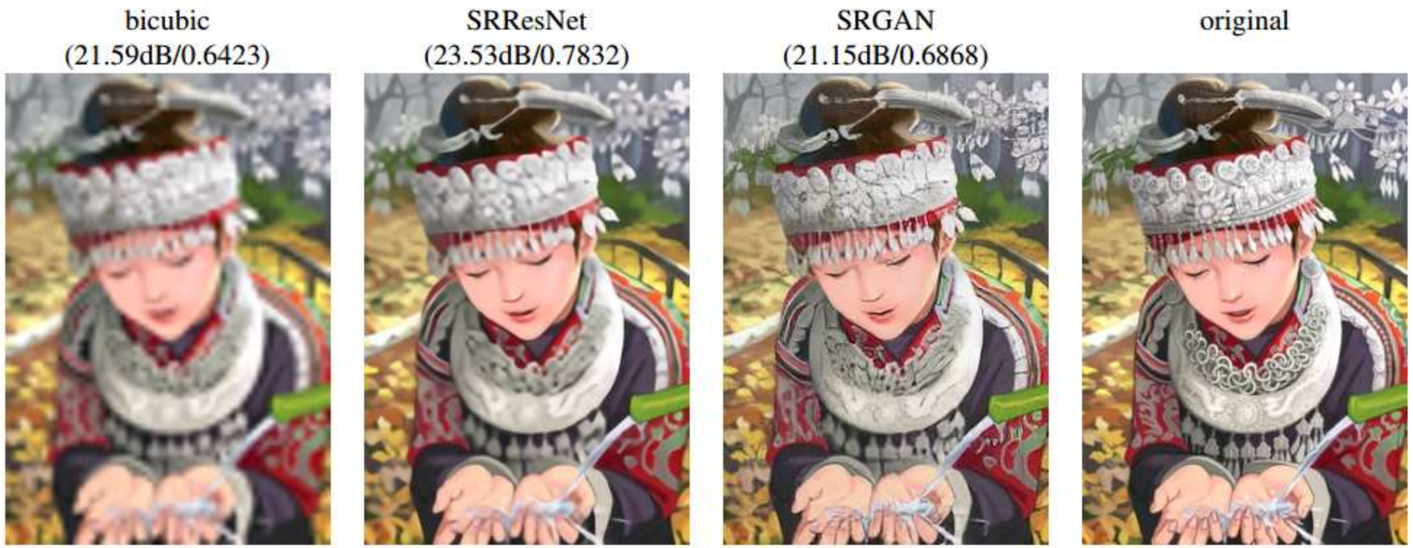
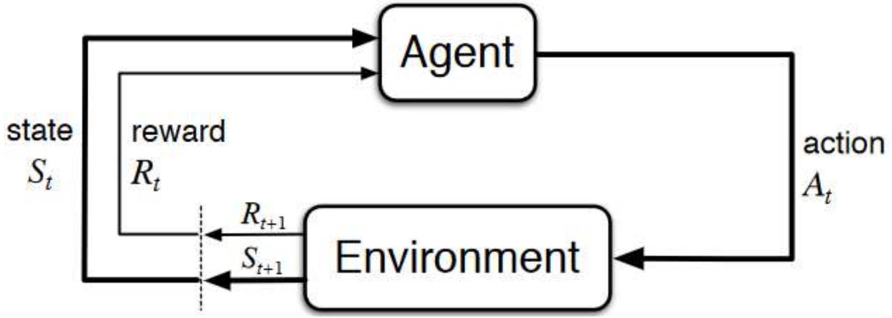


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

马尔科夫决策过程

智能体环境交互

智能体的目标是最大化将来的**期望累积奖励（expected return）**



Episodes:

S_1, a_1, R_2, S_2, a_2, R_3, S_3, a_3, R_4, … …

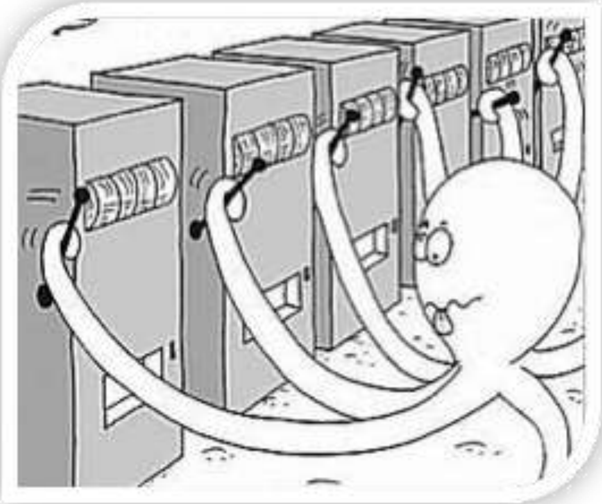
折扣回报（**discounted return**）：

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

强化学习

多臂老虎机

问题： 一台赌博机有多个摇臂，每个摇臂摇出的奖励(reward)大小不确定，玩家希望摇固定次数的臂所获得的期望累积奖励最大.



强化学习

多臂老虎机

形式化定义：行为 A_t 代表第 t 次选择哪个臂，奖励 R_t 代表第 t 次摇臂带来的奖励。采取行为a的期望奖励为:

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$$

假设一共要进行 T 次摇臂，在期望奖励未知的情况下，采取什么样的策略可以获得最大的累计奖励?

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

答案：可以使用 Q_t（a）对 q_*（a）进行估算。

强化学习

多臂老虎机

贪心策略:

$$A_t \doteq \arg\max_a Q_t(a)$$

***ε*-贪心策略:**

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

深度强化学习

- **Alpha-Go**

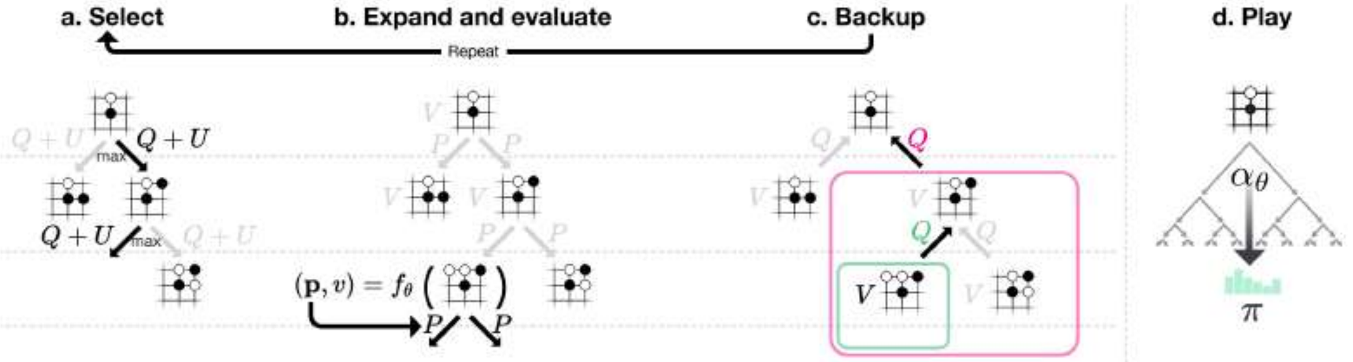


Figure 2: **Monte-Carlo tree search in *AlphaGo Zero*.** **a** Each simulation traverses the tree by selecting the edge with maximum action-value Q, plus an upper confidence bound U that depends on a stored prior probability P and visit count N for that edge (which is incremented once traversed). **b** The leaf node is expanded and the associated position s is evaluated by the neural network $(P(s,\cdot), V(s)) = f_\theta(s)$; the vector of P values are stored in the outgoing edges from s. **c** Action-values Q are updated to track the mean of all evaluations V in the subtree below that action. **d** Once the search is complete, search probabilities $\boldsymbol{\pi}$ are returned, proportional to $N^{1/\tau}$, where N is the visit count of each move from the root state and τ is a parameter controlling temperature.

强化学习

深度强化学习

- **SeqGAN**

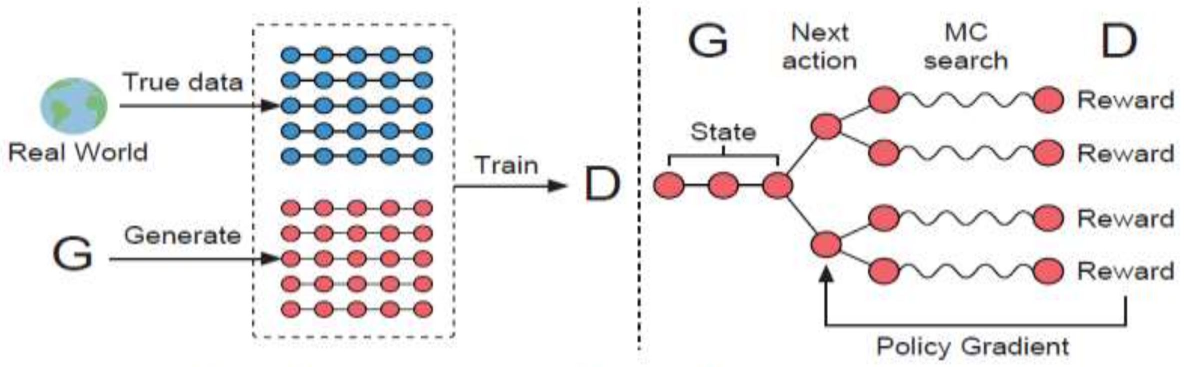


Figure 1: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

强化学习

深度强化学习

- **Visual navigation**

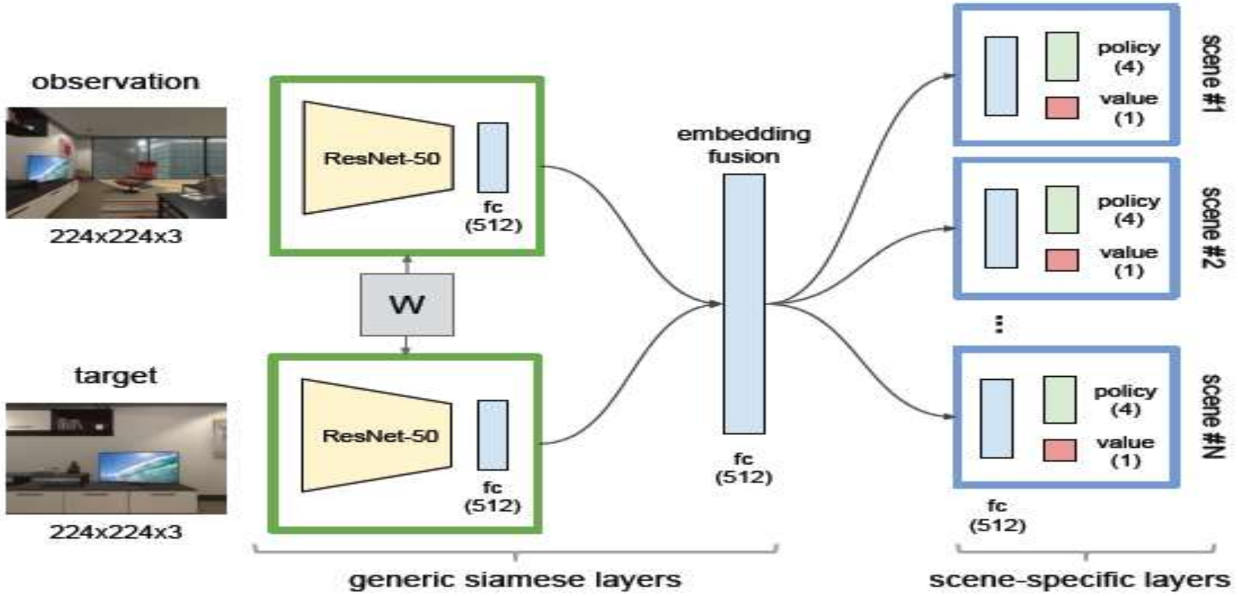


Fig. 4. Network architecture of our deep siamese actor-critic model. The numbers in parentheses show the output dimensions. Layer parameters in the green squares are shared. The ResNet-50 layers (yellow) are pre-trained on ImageNet and fixed during training.

知识图谱

背景

知识图谱的概念最早出现于 **Google** 公司的知识图谱项目，体现在使用 **Google** 搜索引擎时，出现于搜索结果右侧的相关知识展示。

截止到 **2016** 年底，**Google** 知识图谱中的知识数量已经达到了 **600** 亿条，关于 **1500** 个类别的 **5.7** 亿个实体，以及它们之间的 **3.5** 万种关系。

实体、关系和事实

实体(entity)：现实世界中可区分、可识别的事物或概念。

关系(relation)：实体和实体之间的语义关联。

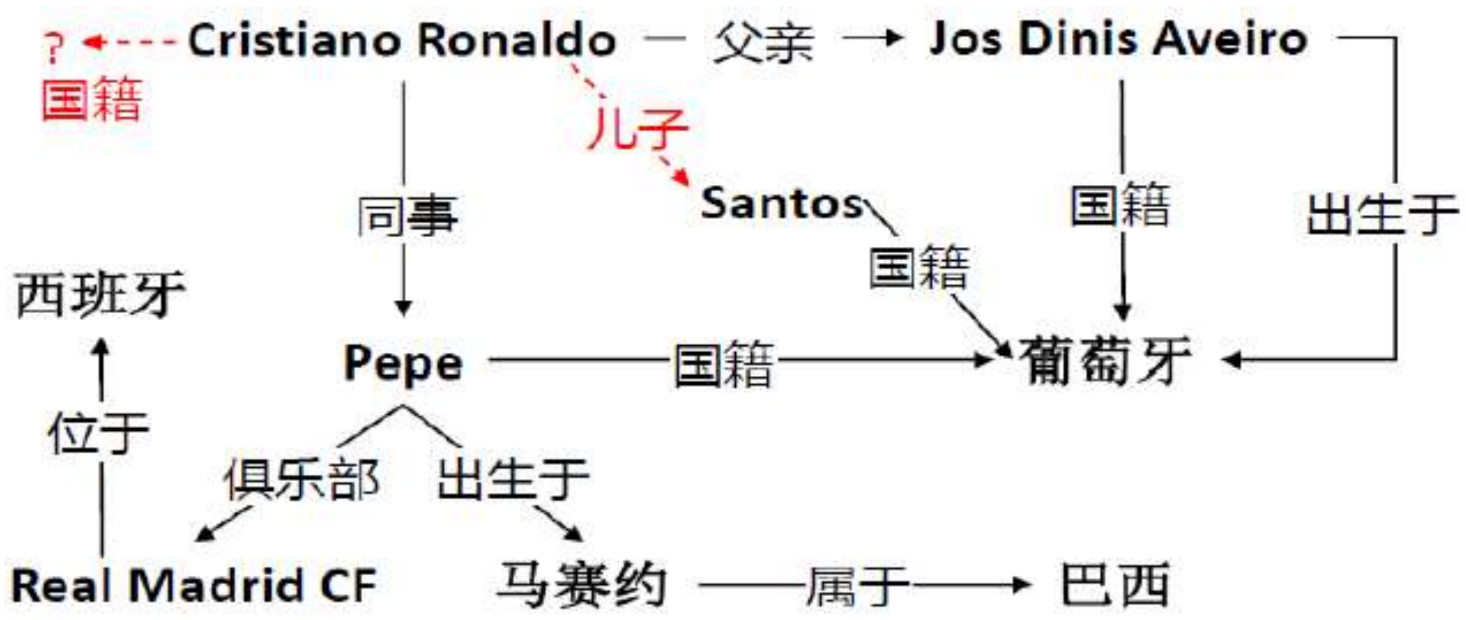
事实(fact)： (head entity, relation, tail entity) 三元组形式。

知识图谱

狭义知识图谱

狭义知识图谱： 具有图结构的三元组知识库。

节点：实体。　边：事实（由头实体指向尾实体）。 边的类型：关系。



知识图谱

链接预测、三元组分类

知识图谱上的链接预测[Bordes et al., 2013]:

INPUT (HEAD AND LABEL)	PREDICTED TAILS
J. K. Rowling influenced by	*G. K. Chesterton*, J. R. R. Tolkien, *C. S. Lewis*, **Lloyd Alexander**, Terry Pratchett, Roald Dahl, Jorge Luis Borges, *Stephen King*, Ian Fleming
Anthony LaPaglia performed in	*Lantana*, *Summer of Sam*, *Happy Feet*, *The House of Mirth*, Unfaithful, **Legend of the Guardians**, Naked Lunch, X-Men, The Namesake
Camden County adjoins	**Burlington County**, *Atlantic County*, *Gloucester County*, Union County, Essex County, New Jersey, Passaic County, Ocean County, Bucks County
The 40-Year-Old Virgin nominated for	*MTV Movie Award for Best Comedic Performance*, *BFCA Critics' Choice Award for Best Comedy*, *MTV Movie Award for Best On-Screen Duo*, MTV Movie Award for Best Breakthrough Performance, **MTV Movie Award for Best Movie**, MTV Movie Award for Best Kiss, D. F. Zanuck Producer of the Year Award in Theatrical Motion Pictures, Screen Actors Guild Award for Best Actor - Motion Picture
Costa Rica football team has position	*Forward*, *Defender*, *Midfielder*, **Goalkeepers**, Pitchers, Infielder, Outfielder, Center, Defenseman
Lil Wayne born in	**New Orleans**, Atlanta, Austin, St. Louis, Toronto, New York City, Wellington, Dallas, Puerto Rico
WALL-E has the genre	Animations, Computer Animation, *Comedy film*, *Adventure film*, *Science Fiction*, **Fantasy**, Stop motion, *Satire*, Drama

知识图谱

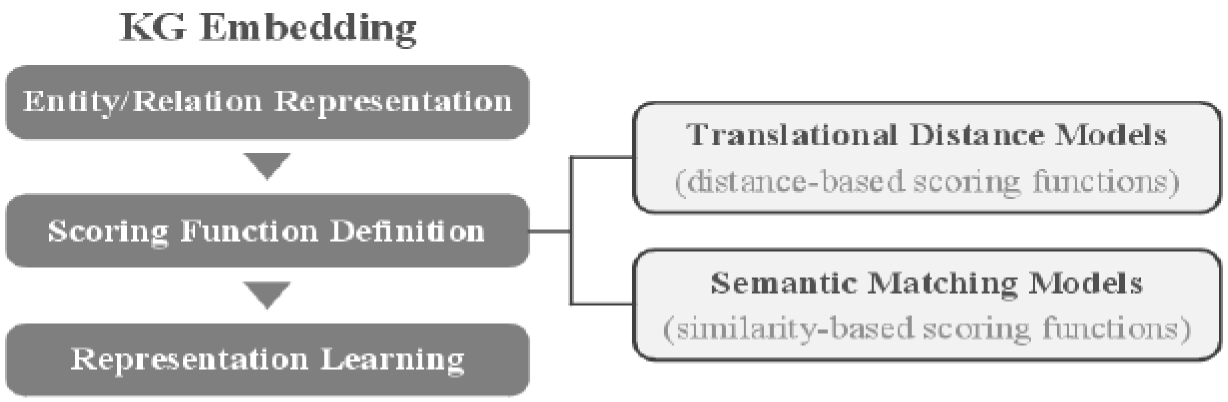
分布式知识表示方法分类[Wang et al., 2017]：

位移距离模型 (translational distance models)：

采用基于距离的打分函数来衡量三元组成立的可能性。

语义匹配模型 (semantic matching models)：

采用基于相似度的打分函数来衡量三元组成立的可能性。



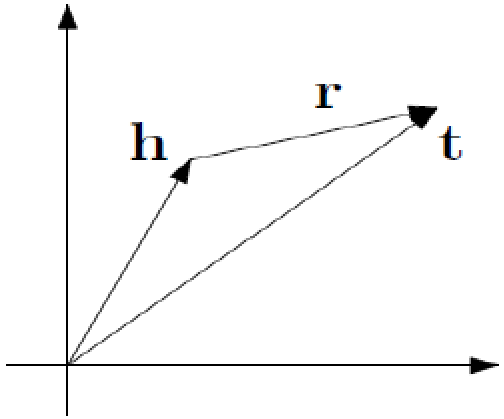
主要的方法

位移距离模型：TransE

TransE [Bordes et al., 2013] 是最具代表性的位移距离模型，其核心思想是实体和关系间的位移假设，即 $\mathbf{h}+\mathbf{r}\approx\mathbf{t}$

China – Beijing = France – Paris = capital-of

Beijing + capital-of = China　　**Paris + capital-of = France**



TransE
实体表示：向量 $\mathbf{h},\mathbf{t}$
关系表示：向量 $\mathbf{r}$
位移操作：$\mathbf{h}+\mathbf{r}\approx\mathbf{t}$
打分函数：$f(h,r,t)=\Vert\mathbf{h}+\mathbf{r}-\mathbf{t}\Vert_{1/2}$

知识图谱

语义匹配模型

计算实体和关系在隐式向量空间的语义匹配程度，以此来判断三元组成立的可能性。

简单匹配模型：RESCAL及其变种。

将头实体和尾实体的表示进行组合后再与关系的表示进行匹配，即:

Matching (relation, Composition (head, tail))

复杂匹配模型：深度神经网络。

利用较为复杂的神经网络结构完成实体和关系的语义匹配，即:

NueralMatching (head, relation, tail)

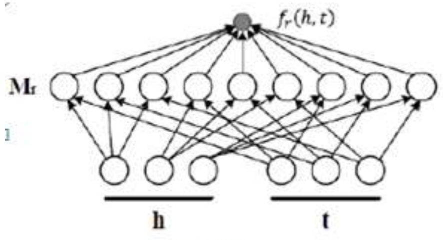
知识图谱

RESCAL 及其变种

将头尾实体的表示进行组合后再与关系的表示进行匹配。

RESCAL [Nickel et al., 2011]

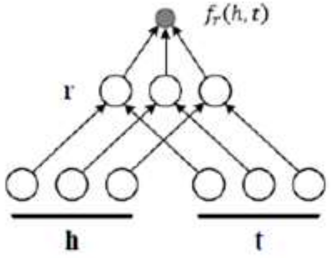
$$[\mathbf{h} \otimes \mathbf{t}]_{ij} = [\mathbf{h}]_i \cdot [\mathbf{t}]_j, \quad f_r(h,t) = \langle \mathbf{M}_r, \mathbf{h} \otimes \mathbf{t} \rangle = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} [\mathbf{M}_r]_{ij} \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_j$$



(a) RESCAL.

DistMult [Yang et al., 2015]

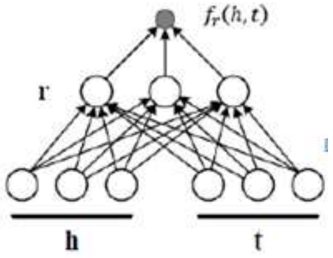
$$[\mathbf{h} \odot \mathbf{t}]_i = [\mathbf{h}]_i \cdot [\mathbf{t}]_i, \quad f_r(h,t) = \langle \mathbf{r}, \mathbf{h} \odot \mathbf{t} \rangle = \sum_{i=0}^{d-1} [\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_i$$



(b) DistMult.

HolE [Nickel et al., 2016]

$$[\mathbf{h} \star \mathbf{t}]_i = \sum_{k=0}^{d-1} [\mathbf{h}]_k \cdot [\mathbf{t}]_{(k+i) \bmod d}, \quad f_r(h,t) = \langle \mathbf{r}, \mathbf{h} \star \mathbf{t} \rangle = \sum_{i=0}^{d-1} [\mathbf{r}]_i \sum_{k=0}^{d-1} [\mathbf{h}]_k \cdot [\mathbf{t}]_{(k+i) \bmod d}$$

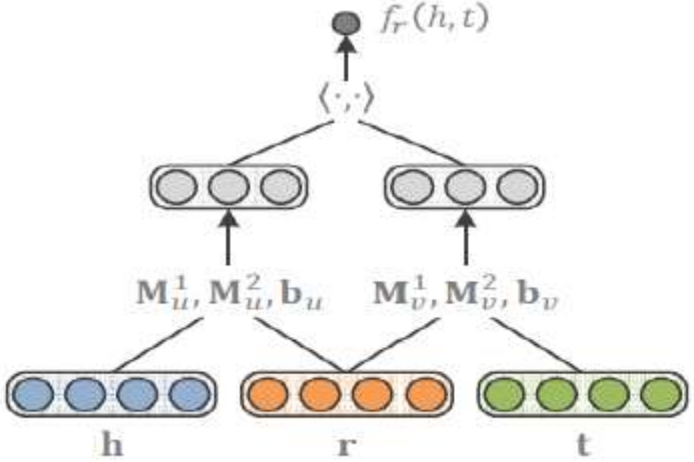


(c) HolE.

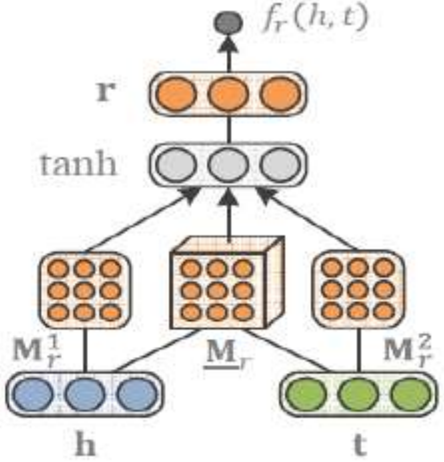
深度神经网络

利用神经网络结构完成实体和关系的语义匹配。

SME[Bordes et al., 2014], NTN[Socher et al., 2013]



(a) SME.



(b) NTN.

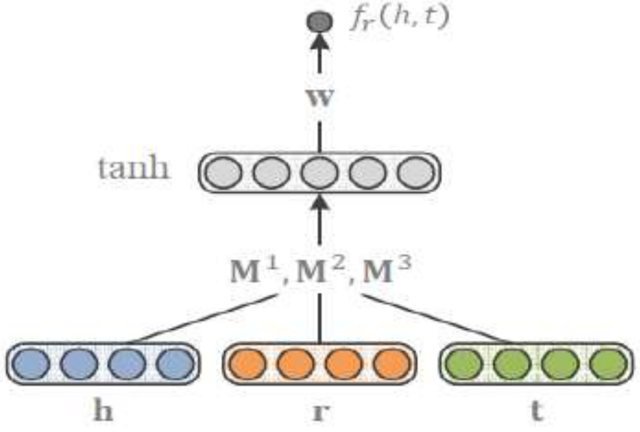
$$f_r(h,t) = g_u(\mathbf{h},\mathbf{r})^\top g_v(\mathbf{t},\mathbf{r}).$$

$$f_r(h,t) = \mathbf{r}^\top \tanh(\mathbf{h}^\top \underline{\mathbf{M}}_r \mathbf{t} + \mathbf{M}_r^1 \mathbf{h} + \mathbf{M}_r^2 \mathbf{t} + \mathbf{b}_r),$$

深度神经网络

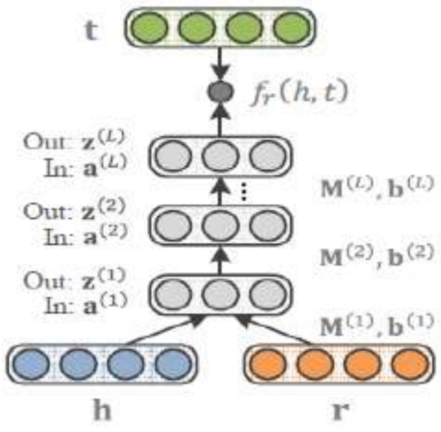
利用神经网络结构完成实体和关系的语义匹配。

MLP[Dong et al., 2014], NAM[Liu et al., 2016]



(c) MLP.

$$f_r(h,t) = \mathbf{w}^\top \tanh(\mathbf{M}^1\mathbf{h} + \mathbf{M}^2\mathbf{r} + \mathbf{M}^3\mathbf{t}).$$



(d) NAM.

$$\mathbf{z}^{(0)} = [\mathbf{h}; \mathbf{r}] \in \mathbb{R}^{2d},$$

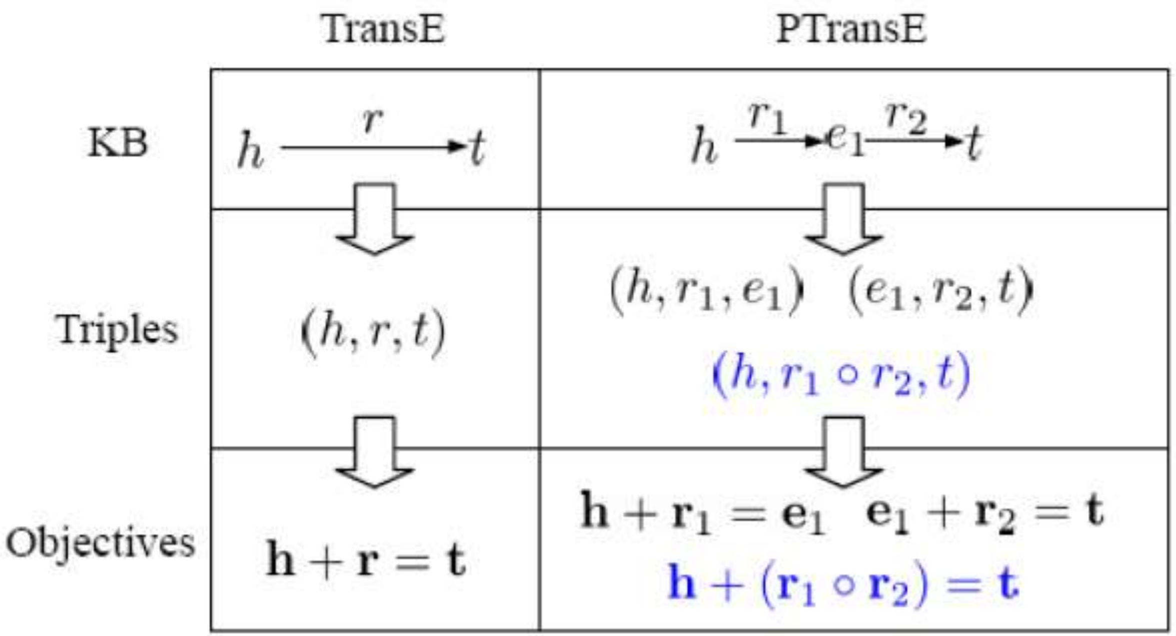
$$\mathbf{a}^{(\ell)} = \mathbf{M}^{(\ell)}\mathbf{z}^{(\ell-1)} + \mathbf{b}^{(\ell)}, \quad \mathbf{z}^{(\ell)} = \mathrm{ReLU}(\mathbf{a}^{(\ell)}),$$

$$f_r(h,t) = \mathbf{t}^\top \mathbf{z}^{(L)}.$$

融合关系路径：PTransE

Path-based TransE (PTransE) [Lin et al., 2015a]：

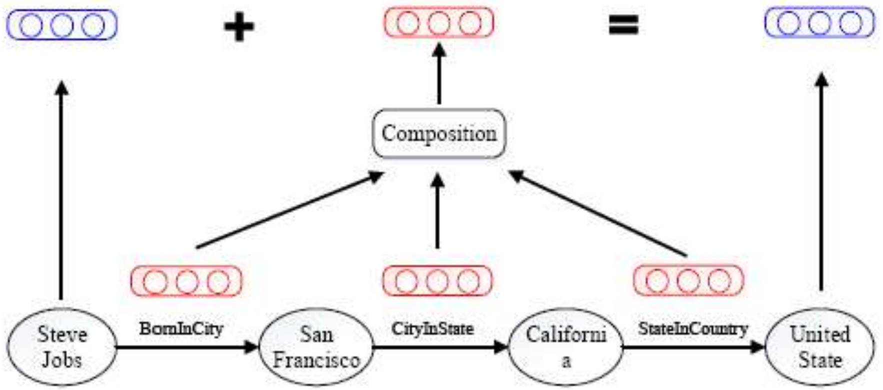
关系路径可被视为远距离实体间的位移操作。



知识图谱

融合关系路径：PTransE

路径表示等于路径上关系表示的语义组合。



Addition : $\mathbf{p} = \mathbf{r}_1 + \ldots + \mathbf{r}_l.$

Multiplication : $\mathbf{p} = \mathbf{r}_1 \cdot \ldots \cdot \mathbf{r}_l.$

Recurrent Neural Network : $c_i = f(W[c_{i-1}; r_i]),$

融合关系路径：PTransE

建模三元组(h, r, t)

$$\mathcal{L}(h,r,t) = \sum_{(h',r,t')\in\mathcal{S}^-} [\gamma + \|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_1 - \|\mathbf{h}'+\mathbf{r}-\mathbf{t}'\|_1]_+$$

建模关系路径(h, p, t)

$$E(h,p,t) = ||\mathbf{p}-(\mathbf{t}-\mathbf{h})|| = ||\mathbf{p}-\mathbf{r}|| = E(p,r),$$

$$\mathcal{L}(p,r) = \sum_{(h,r',t)\in\mathcal{S}^-} [\gamma + \|\mathbf{p}-\mathbf{r}\|_1 - \|\mathbf{p}-\mathbf{r}'\|_1]_+$$

联合学习

$$L(\mathbf{S}) = \sum_{(h,r,t)\in\mathbf{S}} \Big[L(h,r,t) + \frac{1}{Z}\sum_{p\in P(h,t)} R(p|h,t)L(p,r)\Big].$$

本讲参考文献

1. 邱锡鹏，《深度学习与自然语言处理》Slides@CCF ADL 20160529。

2. Stanford Class-CS231n: Convolutional Neural Networks for Visual Recognition。

3. 第十一届中国中文信息学会暑期学校，暨中国中文信息学会《前沿技术讲习班》，201607。

4. Simon Haykin，Neural Network and Learning Machine. 3rd