Attention

Index

Attention

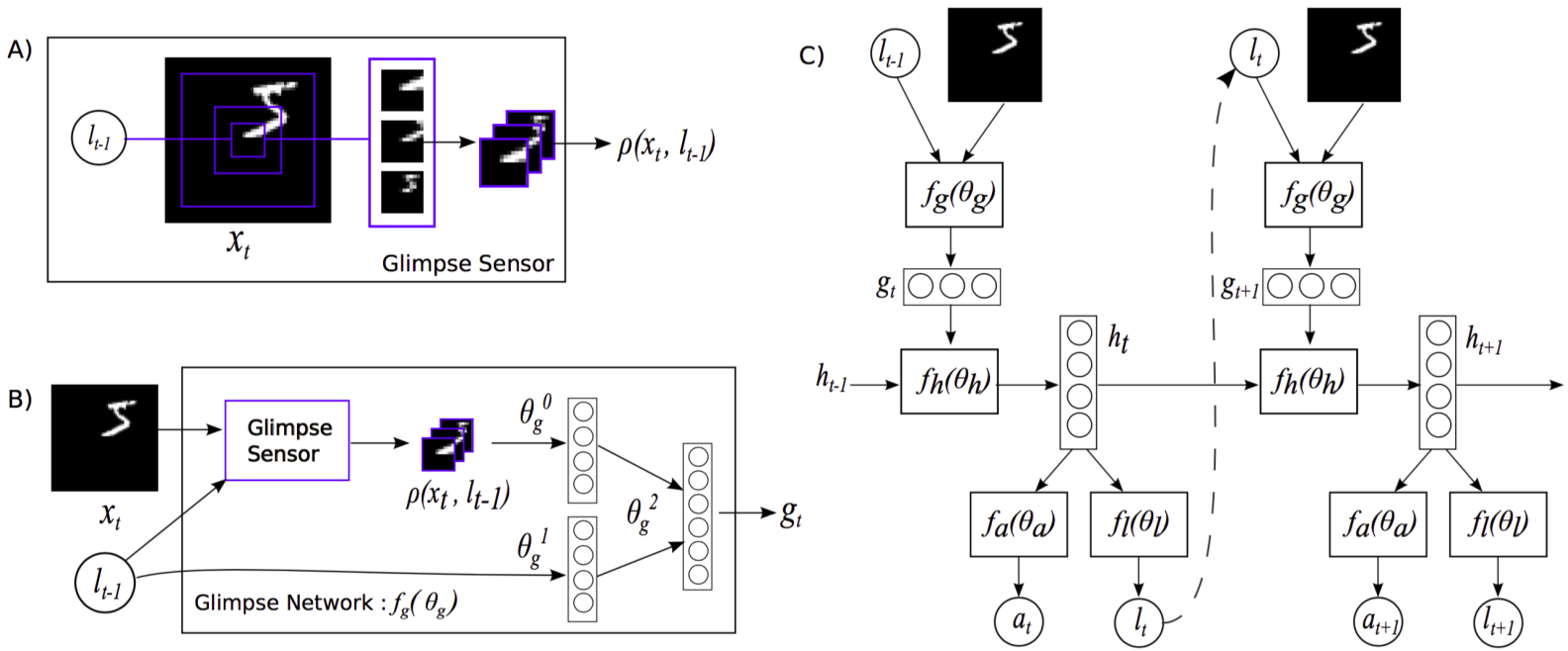
在引入Attention(注意力)之前，图像识别或语言翻译都是直接把完整的图像或语句直接塞到一个输入，然后给出输出。
而且图像还经常缩放成固定大小，引起信息丢失。
而人在看东西的时候，目光沿感兴趣的地方移动，甚至仔细盯着部分细节看，然后再得到结论。
Attention就是在网络中加入关注区域的移动、缩放、旋转机制，连续部分信息的序列化输入。

基于Attention的图片分类

参考 Recurrent Models of Visual Attention (http://arxiv.org/pdf/1406.6247v1.pdf) (2014.06.24)

模型

该模型称为The Recurrent Attention Model，简称RAM。



A、Glimpse Sensor: 在t步，图片x_t的l_{t-1}位置处取不同大小的区域，组合成数据$\rho(x_t, l_{t-1})$

B、Glimpse Network: 将图片局部信息与位置信息整合成g_t

C、Model Architecture: h_{t-1}为隐藏记忆单元，每轮加入新的g_t，生成新的h_t，并以此生成感兴趣的a_t与新的位置l_t

该模型每次迭代的时候，还可以输出缩放信息和结束标志。

训练

网络的参数可表示为$\theta = \{\theta_g, \theta_h, \theta_a, \theta_l\}$，强化学习的策略可表示成$\pi((l_t, a_t) \mid s_{1:t}; \theta)$，其中$s_{1:t} = x_1, l_1, a_1, \cdots, x_{t-1}, l_{t-1}, a_{t-1}, x_t$，另可用$u_t$表示$(l_t, a_t)$。

强化学习得到奖赏为$R = \sum_{t=1}^{T} r_t$。例如，在物体识别中，a_T输出类型是正确时，奖赏为1，否则为0。其它时刻的奖赏为0。

奖赏期望为

$$J(\theta) = \mathbb{E}_{p(s_{1:T};\theta)}\left[\sum_{t=1}^{T} r_t\right] = \mathbb{E}_{p(s_{1:T};\theta)}[R] = \mathbb{E}_{p(s_{1:T};\theta)}\left[\prod_{t=1}^{T} \pi(u_t \mid s_{1:t};\theta) R\right]$$

强化学习的目标是提高 J ，等同于提高 $\log J$ 。对其求导

$$\nabla_\theta(\log J) = \mathbb{E}_{p(s_{1:T};\theta)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi(u_t \mid s_{1:t};\theta)\, R\right] \approx \frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T} \nabla_\theta \log \pi(u_t^i \mid s_{1:t}^i;\theta)\, R^i$$

其中$i = 1 \ldots M$表示第 i 次采样。

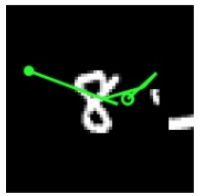
在学习训练过程中，$\nabla_\theta \log \pi(u_t^i \mid s_{1:t}^i;\theta)$不需要显示求出，可直接使用RNN模型的标准反馈梯度。

以上等式是梯度的无偏估计，但可引起高方差，所以引入以下估计

$$\frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T} \nabla_\theta \log \pi(u_t^i \mid s_{1:t}^i;\theta)\, (R_t^i - b_t)$$

其中$b_t = \mathbb{E}_\pi[R_t]$

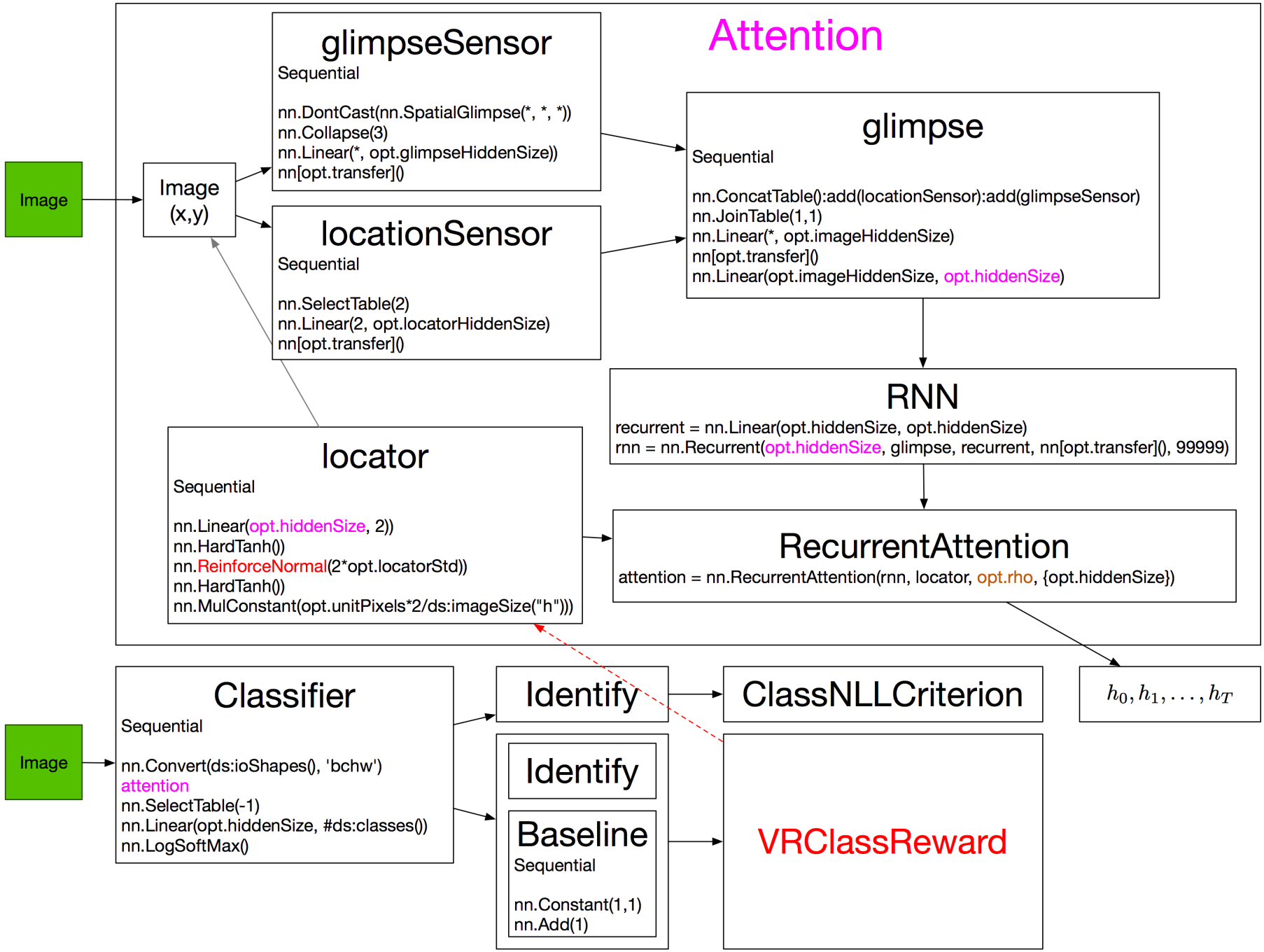
效果



以上是论文中在扩大和污染了的minst数据库上，glimpse的移动方向。
实心绿点是开始，空心绿点是结束。
可以看到，RAM模型顺着感兴趣的方向移动。
识别效果比全链接的网络，和基于CNN的网络都要好。

Torch代码结构

在博客Recurrent Model of Visual Attention (http://torch.ch/blog/2015/09/21/rmva.html)的训练代码 (https://github.com/Element-Research/rnn/blob/master/examples/recurrent-visual-attention.lua)中，结构如下

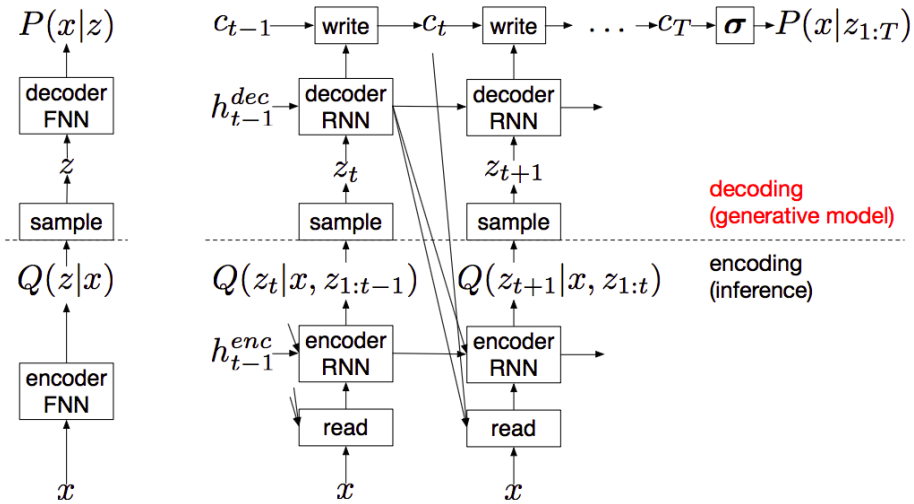

Attention
glimpseSensor
Sequential
nn.DontCast(nn.SpatialGlimpse(*, *, *))
nn.Collapse(3)
nn.Linear(*, opt.glimpseHiddenSize))
nn[opt.transfer]()
locationSensor
Sequential
nn.SelectTable(2)
nn.Linear(2, opt.locatorHiddenSize)
nn[opt.transfer]()
glimpse
Sequential
nn.ConcatTable():add(locationSensor):add(glimpseSensor)
nn.JoinTable(1,1)
nn.Linear(*, opt.imageHiddenSize)
nn[opt.transfer]()
nn.Linear(opt.imageHiddenSize, opt.hiddenSize)
RNN
recurrent = nn.Linear(opt.hiddenSize, opt.hiddenSize)
rnn = nn.Recurrent(opt.hiddenSize, glimpse, recurrent, nn[opt.transfer](), 99999)
locator
Sequential
nn.Linear(opt.hiddenSize, 2))
nn.HardTanh())
nn.ReinforceNormal(2*opt.locatorStd))
nn.HardTanh())
nn.MulConstant(opt.unitPixels*2/ds:imageSize("h")))
RecurrentAttention
attention = nn.RecurrentAttention(rnn, locator, opt.rho, {opt.hiddenSize})
Image
Image
(x,y)
Image
Classifier
Sequential
nn.Convert(ds:ioShapes(), 'bchw')
attention
nn.SelectTable(-1)
nn.Linear(opt.hiddenSize, #ds:classes())
nn.LogSoftMax()
Identify
ClassNLLCriterion
$h_0, h_1, \ldots, h_T$
Identify
Baseline
Sequential
nn.Constant(1,1)
nn.Add(1)
VRClassReward

基于Attention的图片生成

Auto-Encoding Variational Bayes (http://arxiv.org/abs/1312.6114) (2014.05.01)

DRAW: A Recurrent Neural Network For Image Generation (http://arxiv.org/pdf/1502.04623.pdf) (2015.05.20)

模型



约定: W 为线性变换。

隐变量 z_t 符从高斯分布$N(Z_t \mid \mu_t, \sigma_t)$:

$$\mu_t = W(h_t^{enc})$$
$$\sigma_t = \exp(W(h_t^{enc}))$$

模型结构的具体计算

$$\hat{x}_t = x - \sigma(c_{t-1})$$
$$r_t = read(x_t, \hat{x}_t, h_{t-1}^{dec})$$
$$h_t^{enc} = RNN^{enc}(h_{t-1}^{enc}, [r_t, h_{t-1}^{dec}])$$
$$z_t \sim Q(Z_t \mid h_t^{enc})$$
$$h_t^{dec} = RNN^{dec}(h_{t-1}^{dec}, z_t)$$
$$c_t = c_{t-1} + write(h_t^{dec}))$$

代价函数

$$
\begin{aligned}
L^x &= -\log D(x \mid c_T) \\
L^z &= \sum_{t=1}^{T} KL(Q(Z_t \mid h_t^{enc}) \mid\mid P(Z_t)) \\
L &= L^x + L^z
\end{aligned}
$$

其中 $P(Z_t)$ 是标准正态分布，所以 L^z 为

$$
L^z = \frac{1}{2} \sum_{t=1}^{T} \left(\mu_t^2 + \sigma_t^2 - \log \sigma_t^2\right) - T/2
$$

生成过程

以上是训练过程，下面是生成过程

$$
\begin{aligned}
z_t &\sim P(Z_t) \\
h_t^{dec} &= RNN^{dec}(h_{t-1}^{dec}, z_t) \\
c_t &= c_{t-1} + write(h_t^{dec})) \\
x &\sim D(X \mid C_T)
\end{aligned}
$$

如果结果为二值域，D 可为伯努利分布。

读写

读写使用 $N \times N$ 的格子框，高斯分布进行。
格子点的位置为

$$
\begin{aligned}
\mu_X^i &= g_X + (i - N/2 - 0.5)\delta \\
\mu_Y^j &= g_Y + (j - N/2 - 0.5)\delta
\end{aligned}
$$

其中参数由如下生成

$$
\begin{aligned}
&(\hat{g}_X, \hat{g}_Y, \log \sigma^2, \log \hat{\delta}, \log \gamma) = W(h^{dec}) \\
&g_X = \frac{A+1}{2}(\hat{g}_X + 1) \\
&g_Y = \frac{A+1}{2}(\hat{g}_Y + 1) \\
&\delta = \frac{\max(A, B) - 1}{N - 1}\hat{\delta}
\end{aligned}
$$

A、B 表示输入图片 x 的大小。

σ 为高斯采样的方差。

γ 为读写强度。

采样矩阵如下生成

$$F_X[i,a] = \frac{1}{Z_X}\exp(-\frac{(a-\mu_X^i)^2}{2\sigma^2})$$
$$F_Y[j,b] = \frac{1}{Z_Y}\exp(-\frac{(b-\mu_Y^i)^2}{2\sigma^2})$$

两个矩阵大小分别为 $N \times A$ 与 $N \times B$。

a 与 b 分别为图像上点的位置坐标。

Z_X 与 Z_Y 为归一化: $\sum_a F_X[i,a] = 1$ 与 $\sum_b F_X[j,b] = 1$。

$$read(x_t, \hat{x}_t, h_{t-1}^{dec}) = \gamma[F_Y x F_X^T, F_Y \hat{x} F_X^T]$$

$$w_t = W(h_t^dec)$$

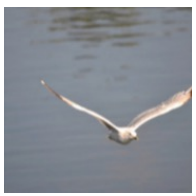
$$write(h^{dec}) = \frac{1}{\hat{\gamma}}\hat{F}_Y^T w_t \hat{F}_X$$

实现

https://github.com/CosmosShadow/DRAW (https://github.com/CosmosShadow/DRAW)

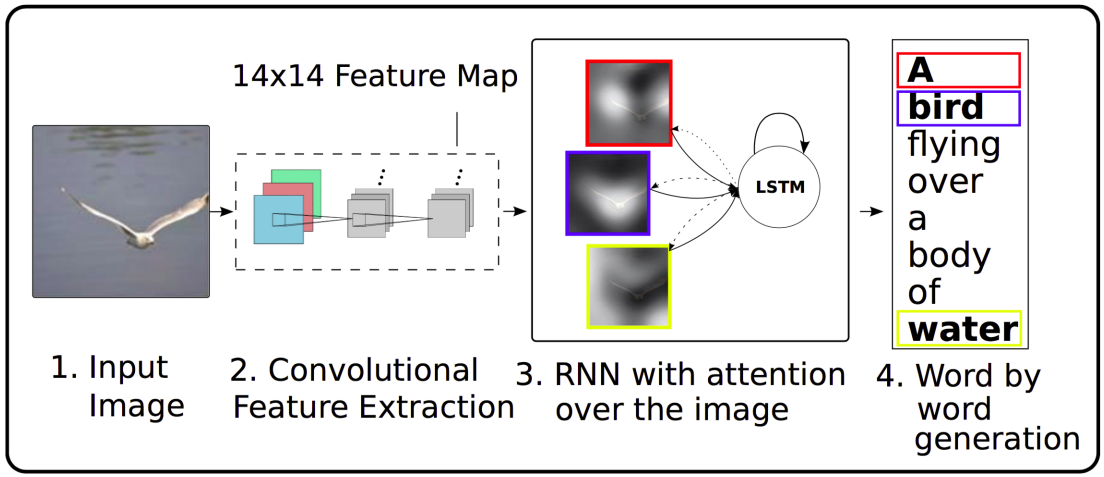
基于Attention的图片主题生成

参考 Show, Attend and Tell: Neural Image Caption Generation with Visual Attention (http://arxiv.org/pdf/1502.03044v2.pdf) (2015.02.10)

 $\Longrightarrow$ A bird flying over a body of water

如上，根据图片，生成主题描述。

模型



如上图，模型把图片经过CNN网络，变成特征图。
LSTM的RNN结构在此上运行Attention模型，最后得到主题输出。

编码

特征图均匀地切割成多个区域，表示为

$$a = \{\mathbf{a}_1, \ldots, \mathbf{a}_L\},\ a_i \in \mathbb{R}^D$$

L表示切割的区域个数。
如区域大小为14×14，$D = 196$。

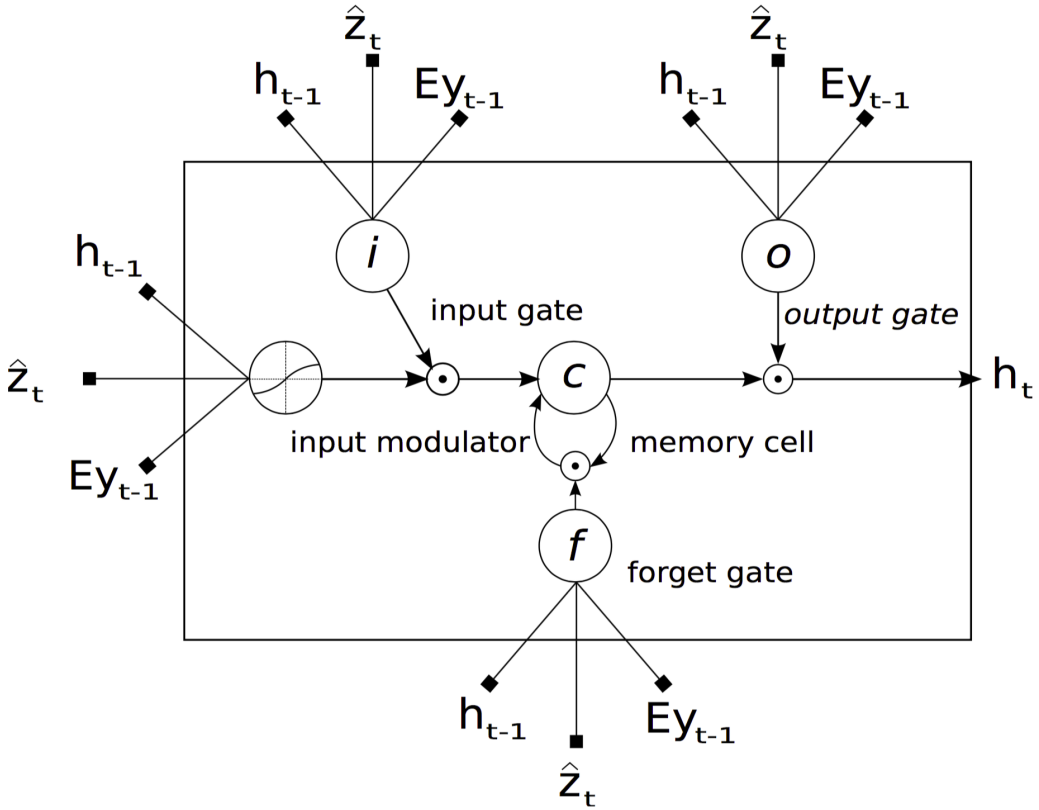
输出的主题y可以编码为

$$y = \{\mathbf{y}_1, \ldots, \mathbf{y}_C\}, y_i \in \mathbb{R}^K$$

K是字典的单词个数，C是句子长度。
$\mathbf{y}_i$的形式为$(0, 0, \ldots, 0, 1, 0, \ldots, 0, 0)$，即只有第$i$处位置为1，其它位置为0。

解码

该模型使用的LSTM
(http://www.cosmosshadow.com/cs/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0/2015/11/16/%E4%BB%8ERNN%E5%88%B0LSTM.html)
如下图所示



运算为

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{T}_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

其中 σ 是 sigmoid 函数，$\odot$ 表示同位置的元素相乘。
$\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ 分别为输入门、遗忘门、输出门，$\mathbf{g}_t$是对输入的转化，$\mathbf{c}_t$是cell状态，$\mathbf{h}_t$是隐藏状态。
$\mathbf{E} \in \mathbb{R}^{m \times K}$ ，通过随机初始化学习到的矩阵。

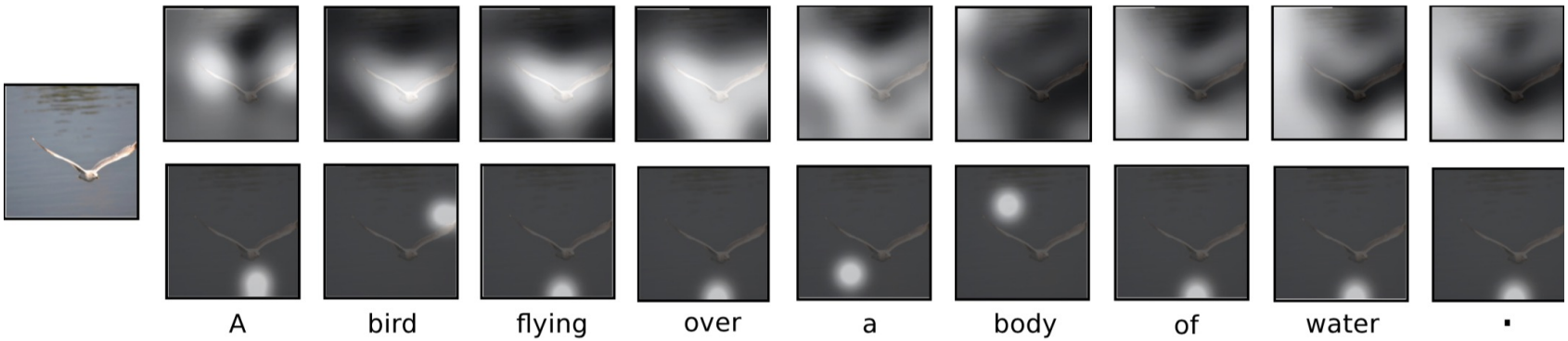
$\hat{\mathbf{z}}_t$是对整张图片部分信息的动态表示，一个Attention模型，如下计算

$$e_{ti} = f_{\mathbf{att}}(\mathbf{a}_i, \mathbf{h}_{t-1})$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}$$

$$\hat{\mathbf{z}}_t = \phi(\{\mathbf{a}_i\}, \{\alpha_{ti}\})$$

其中 i 表示第 i 个特征区域，共 L 个。
函数 $f_{\mathbf{att}}$ 采用多层网络实现，利用前一时刻的隐藏状态 $\mathbf{h}_{t-1}$ 与 L 个特征区域，分别得到每个区域的权重 α_{ti} 。
权重 α_{ti} 可以理解为(1)下一步选择哪一个特征区域的概率，也可以理解为(2)每一个特征区域在下一次输入中所占的比例。
不同的理解与应用，体现在函数 ϕ 的不同实现上。
按(1)实现称为 Stochastic “Hard” Attention ，按(2)实现称为 Deterministic “Soft” Attention。
下图上一排为 soft 模型，下一排为 hard 模型。



LSTM中的记忆单元与隐藏单元的初始值，是两个不同的多层感知机，采用所有特征区域的平均值来进行预测的:

$$\mathbf{c}_0 = f_{\mathbf{init.c}}(\frac{1}{L}\sum_{i}^{L}\mathbf{a}_i)$$

$$\mathbf{h}_0 = f_{\mathbf{init.h}}(\frac{1}{L}\sum_{i}^{L}\mathbf{a}_i)$$

而最终的单词概率输出，采用深度输出层实现

$$p(\mathbf{y}_t \mid \mathbf{a}, \mathbf{y}_{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h\mathbf{h}_t + \mathbf{L}_z\hat{\mathbf{z}}_t))$$

其中 $\mathbf{L}_o \in \mathbb{R}^{K\times m}$ ，$\mathbf{L}_h \in \mathbb{R}^{m\times n}$ ，$\mathbf{L}_z \in \mathbb{R}^{m\times D}$。

Stochastic “Hard” Attention

$s_{t,i}$ 指示是否选择 L 个特征图中的第 i 个，如果设置成1，表示选中，0表示不选中。
在随机’Hard’模型中，只有唯一的选中。
$\hat{\mathbf{z}}_t$ 变量如下计算

$$p(s_{t,i} = 1 \mid \mathbf{a}) = \alpha_{t,i}$$

$$\hat{\mathbf{z}}_t = \sum_{i=1}^{L} s_{t,i}\mathbf{a}_i$$

我们设置 $\log p(\mathbf{y} \mid \mathbf{a})$ 函数的下限为目标函数 L_s:

$$\begin{aligned} L_s &= \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a}) \\ &\leq \log \sum_s p(s \mid \mathbf{a})\, p(\mathbf{y} \mid s, \mathbf{a}) \\ &= \log p(\mathbf{y} \mid \mathbf{a}) \end{aligned}$$

对其进行参数求导有

$$\frac{\partial L_s}{\partial W} = \sum_s p(s \mid \mathbf{a}) \left[\frac{\partial \log p(\mathbf{y} \mid s, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid s, \mathbf{a}) \frac{\partial \log p(s \mid \mathbf{a})}{\partial W} \right]$$

以上参数求导可用Monte Carlo
(http://www.cosmosshadow.com/cs/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0/2015/07/27/%E9%9A%8F%E6%9C%BA%E6%A8%A1%E6%8B%9F
方法采样实现

$$\widetilde{s}_t \sim \text{Multinoulli}_L(\{\alpha_i\})$$

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} p(\widetilde{s}^n \mid \mathbf{a}) \left[\frac{\partial \log p(\mathbf{y} \mid \widetilde{s}^n, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid \widetilde{s}^n, \mathbf{a}) \frac{\partial \log p(\widetilde{s}^n \mid \mathbf{a})}{\partial W} \right]$$

为减少估计方差，可采用冲量方式，第k个 mini-batch 的时候

$$b_k = 0.9 \times b_{k-1} + 0.1 \times \log p(\mathbf{y} \mid \widetilde{s}_k, \mathbf{a})$$

为进一步减少估计方差，引入 multinoulli 分布的熵 $H(s)$

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N}\sum_{n=1}^{N} p(\tilde{s}^n \mid \mathbf{a}) \left[\frac{\partial \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})}{\partial W} + \lambda_r (\log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a}) - b) \frac{\partial \log p(\tilde{s}^n \mid \mathbf{a})}{\partial W} + \lambda_e \frac{\partial H[\tilde{s}^n]}{\partial W} \right]$$

λ_r 与 λ_e 是两个超参。
以上参数求导优化的过程就是强化学习，每次选择下一个特征图的过程都朝目标更好的方向变化。

Deterministic “Soft” Attention

上面的随机模型需要采样位置 s_t ，我们还可以通过直接计算 $\hat{\mathbf{z}}_t$

$$\mathbb{E}_{p(s_t|a)}[\hat{\mathbf{z}}_t] = \sum_{i=1}^{L} \alpha_{t,i} \mathbf{a}_i$$

这就是Deterministic “Soft” Attention模型，通过 α 来选择感兴趣的特征区域。
该模型可以通过端到端的的反馈方法进行学习。

在计算 α 的时候，有 $\sum_i \alpha_{t,i} = 1$ 来保证感兴趣的所有区域的权重和为1。
另外，可以加入一个新的正则化，每一个区域在 T 步中，被观察的权重拉近:

$$\sum_t \alpha_{t,i} \approx 1$$

这个正则的加入，可以使得生成的主题更加丰富。就是结果更好嘛！

另外，在 $\hat{\mathbf{z}}_t$ 的计算中添加一个标量进行缩放，通过前一个隐藏单元 $\mathbf{h}_{t-1}$ 来计算

$$\mathbb{E}_{p(s_t|a)}[\hat{\mathbf{z}}_t] = \beta \sum_{i=1}^{L} \alpha_{t,i} \mathbf{a}_i$$

$$\beta_t = \sigma(f_\beta(\mathbf{h}_{t-1}))$$

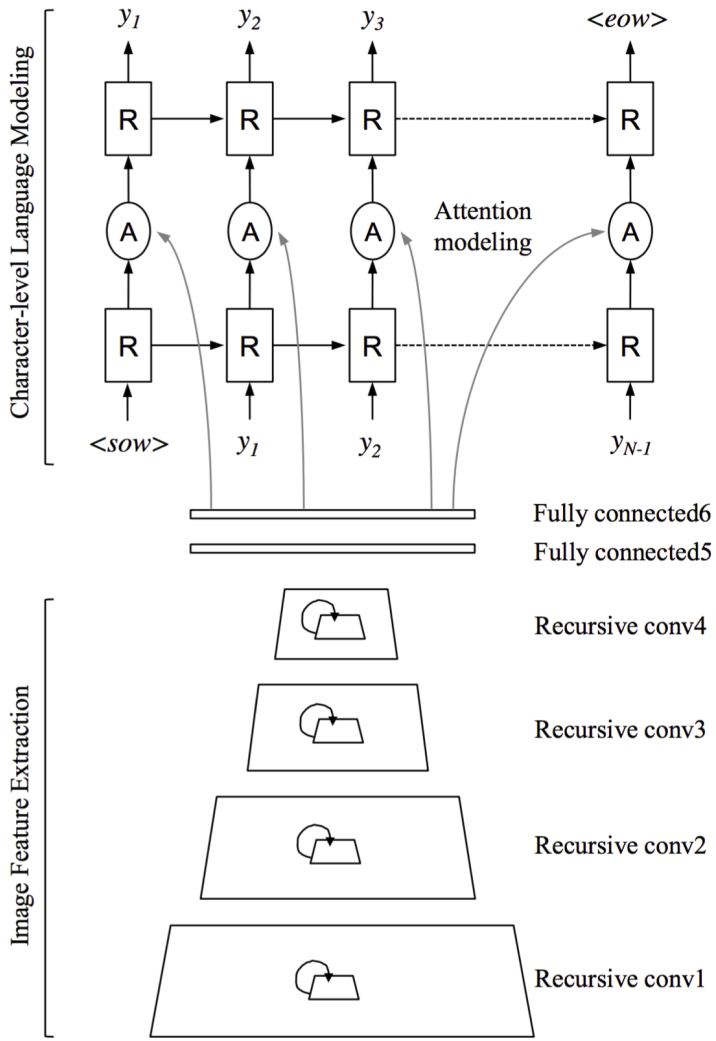
最终，端到端的目标函数可写为

$$L_d = -\log(P(\mathbf{y} \mid \mathbf{x})) + \lambda \sum_i^L (1 - \sum_t^C \alpha_{t,i})^2$$

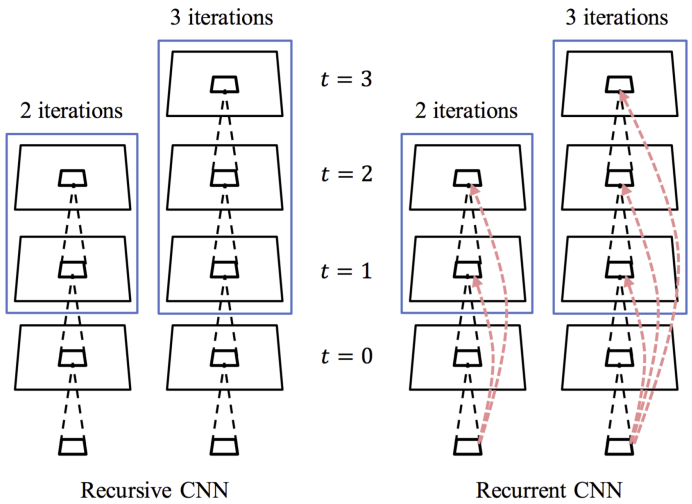
基于Attention的字符识别

参考 Recursive Recurrent Nets with Attention Modeling for OCR in the Wild (http://arxiv.org/pdf/1603.03101v1.pdf) (2016.03.09)

模型



Recursive / Recurrent CNN



CNN是卷积层权重共享。

Recursive CNN是在卷积层中添加多层，每层的卷积核共享:

$$h_{i,j,k}(t) = \begin{cases} \sigma((\mathbf{w}_k^{hh})^T \mathbf{x}_{i,j} + b_k) & at\ t = 0 \\ \sigma((\mathbf{w}_k^{hh})^T \mathbf{h}_{i,j}(t-1) + b_k) & at\ t > 0 \end{cases}$$

Recurrent CNN也是在卷积层中添加多层，但每层都在最初信息的参与，卷积核可以共享，也可能不共享:

$$h_{i,j,k}(t) = \sigma\left((\mathbf{w}_k^r)^T \mathbf{h}_{i,j}(t-1) + (\mathbf{w}_k^f)^T \mathbf{x}_{i,j} + b_k\right)$$

Recursive与Recurrent CNN有都提高感受野，减少参数的作用。

在参考这篇论文中，有提到Recursive CNN效果比Recurrent CNN好。

Attention is all you need

[Paper] (https://arxiv.org/pdf/1706.03762.pdf)