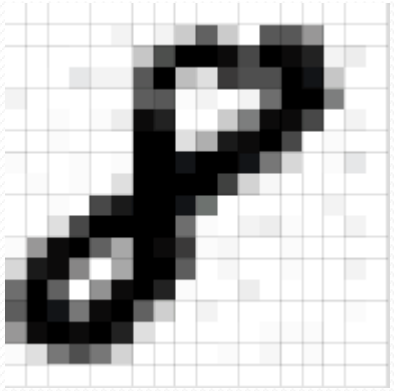
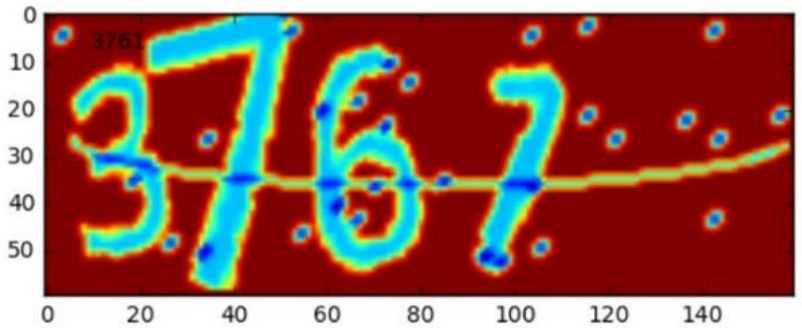
Lab4 基于RNN---LSTM+CTC的注册码识别实践

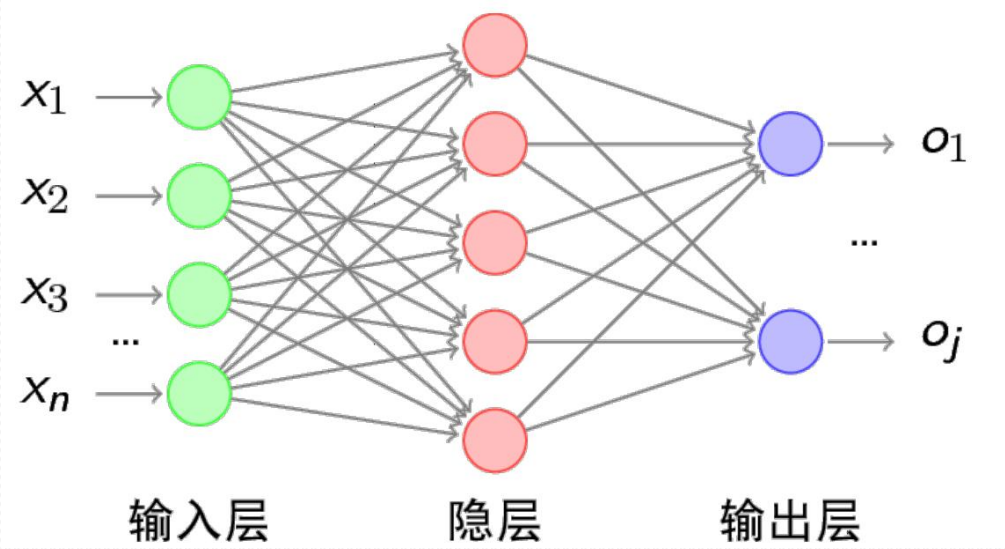
实验大纲

- RNN原理
  - LSTM
  - CTC
- 实验验证
  - 环境依赖
  - 示例程序
- 编程练习

回顾实验一、实验二





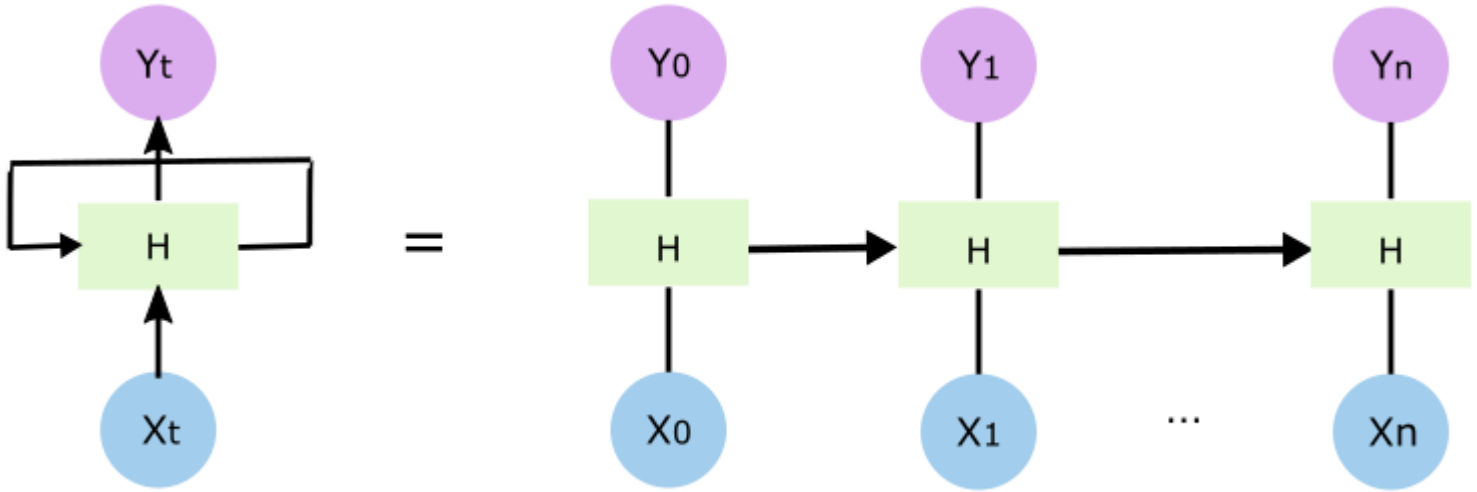


每次识别的对象都是独立，不相关的

循环神经网络RNN(Recurrent neural network)

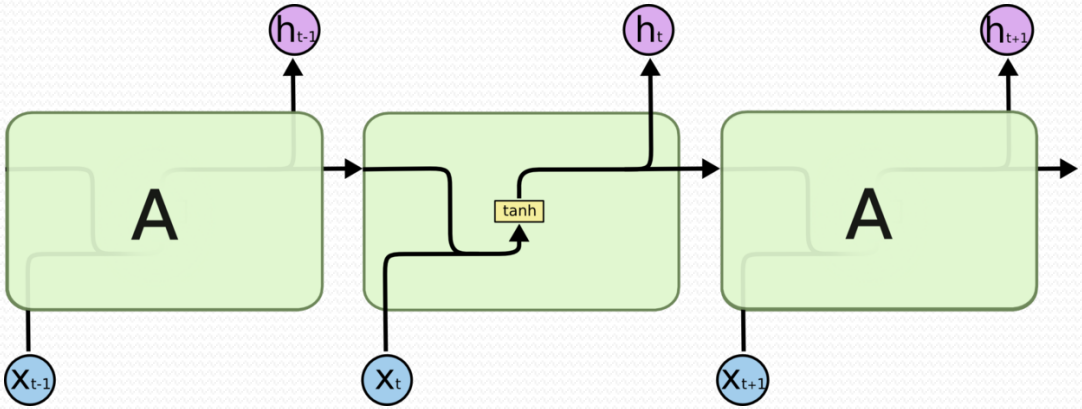
- 概念：
  对时间序列上的变化进行建模的一种神经网络

- 优点：
  基于之前的运行结果或者时间点，进行当前的预测

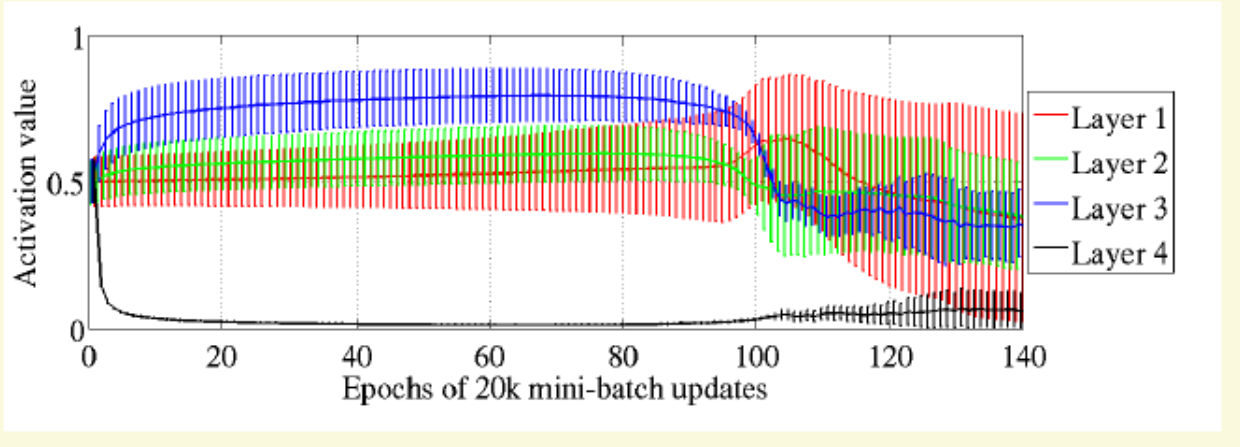


两个输出结果：一是当前层输出结果，二是参与下一层运算输出的中间结果

循环神经网络RNN



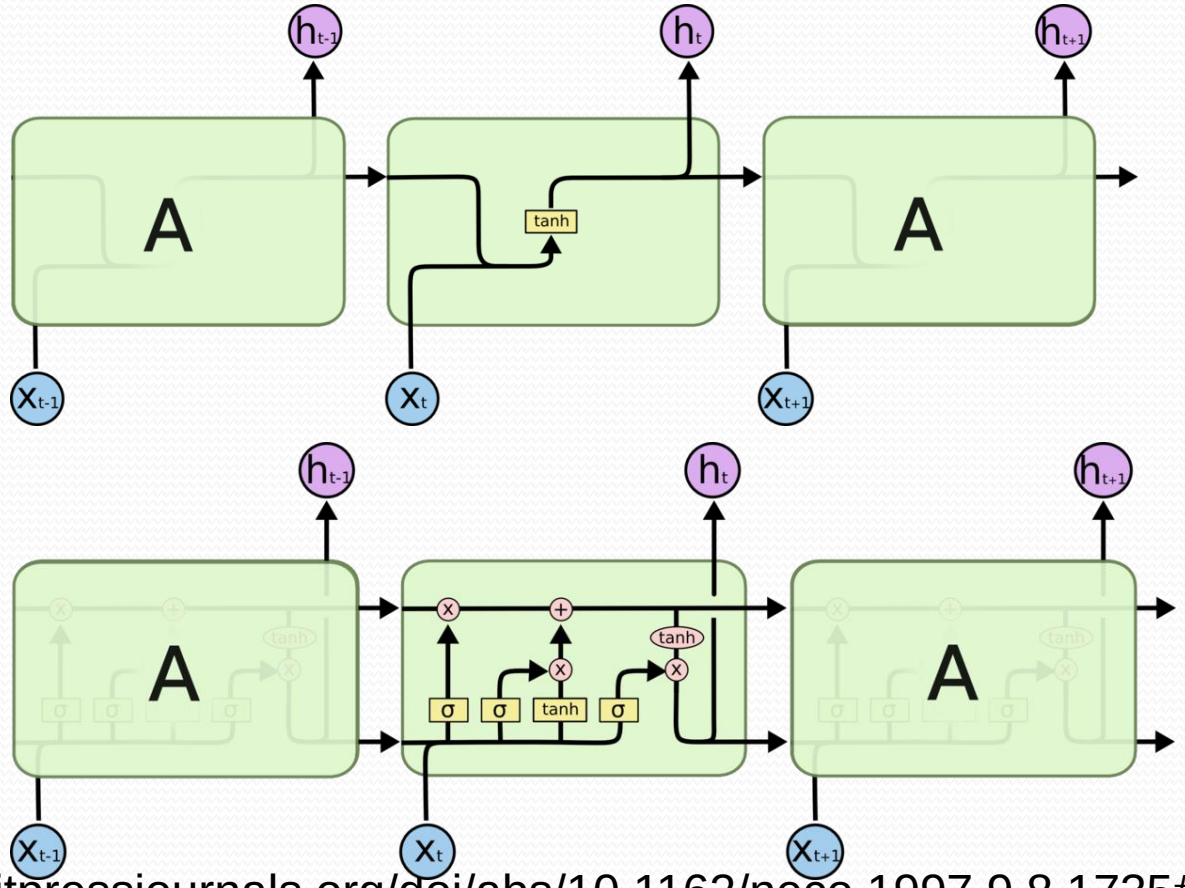
然而，RNN在处理长期依赖（时间序列上距离较远的节点）时，距离较远的节点之间的联系时会涉及雅可比矩阵的多次相乘，这会带来梯度消失（经常发生）或者梯度膨胀（较少发生）的问题



Glorot, Xavier, and YoshuaBengio. "Understanding the difficulty of training deep feed forward neuralnetworks." International conference on artificial intelligence andstatistics.

循环神经网络RNN

- LSTM http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- LSTM算法全称为Long short-term memory，最早由 Sepp Hochreiter和Jürgen Schmidhuber于1997年提出，是一种特定形式的RNN



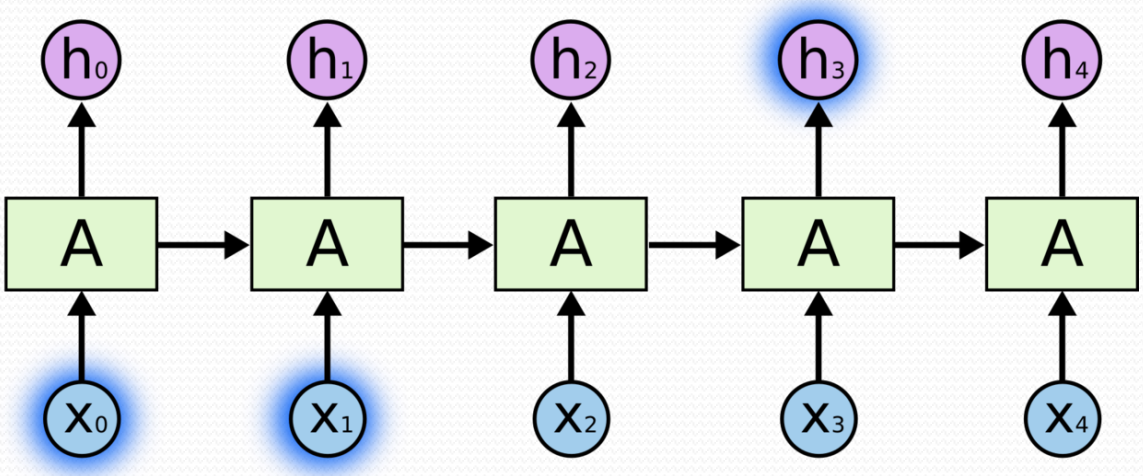
http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735#.V9fMNZN95TY

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

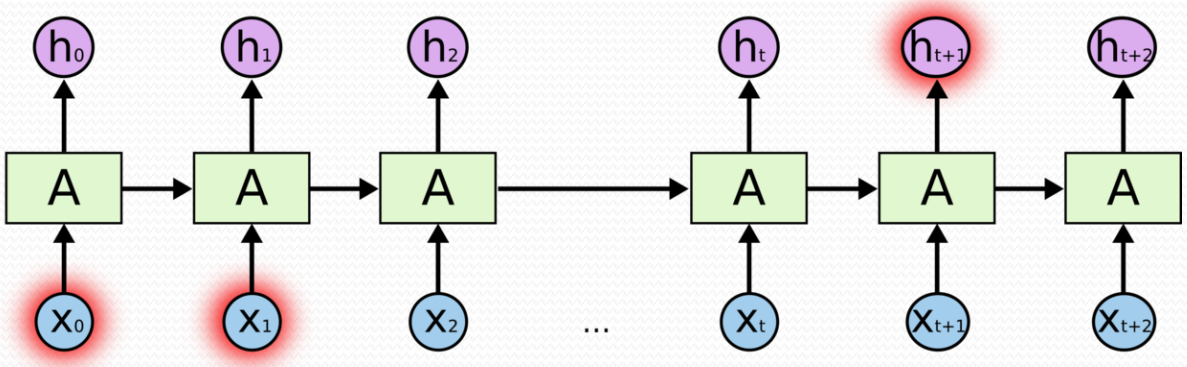
循环神经网络RNN

- LSTM

"the clouds are in the *sky*"



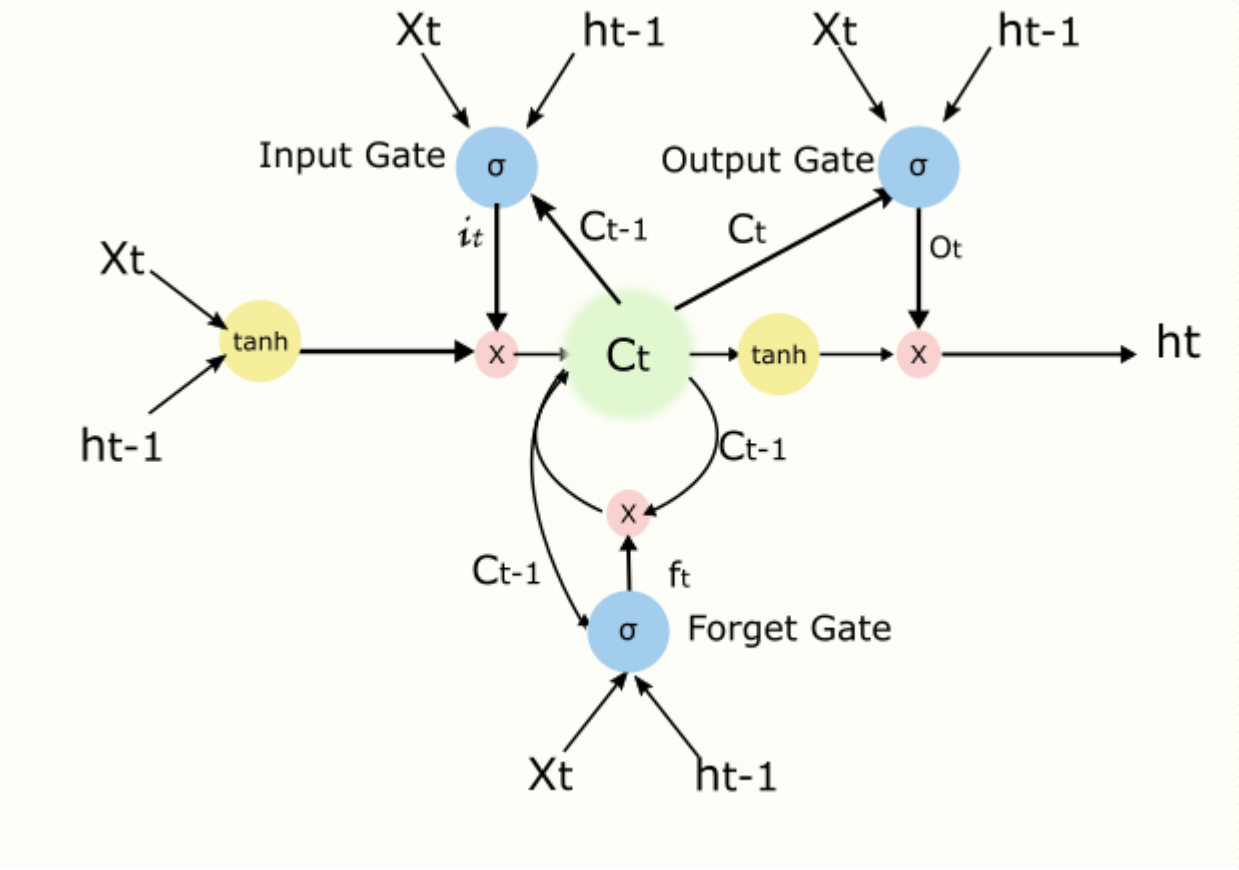
"I grew up in France... I speak fluent *French*"



http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735#.V9fMNZN95TY

循环神经网络RNN

- Peephole LSTM



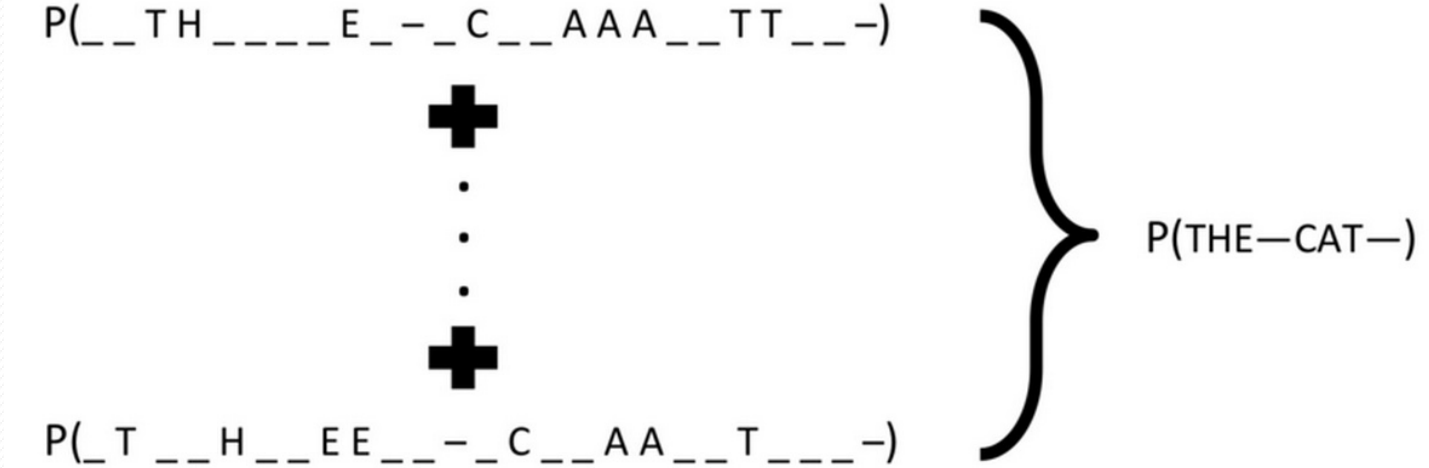
Graves A. Supervisedsequence labelling with recurrent neural networks[M]. Heidelberg: Springer,2012.

循环神经网络RNN

- CTC

Connectionist Temporal Classifier 一般译为联结主义时间分类器 [1]，适合于输入特征和输出标签之间对齐关系不确定的时间序列问题,CTC可以自动端到端地同时优化模型参数和对齐切分的边界

DNN模型Softmax层的输出，获得了基于输入特征序列 ftr 的每一个相互独立建模单元个体（包括“空白”节点在内）的类属概率分布。在此基础上，CTC进一步计算出基于 ftr 的给定输出标签序列 lab 的条件概率 p(lab|ftr)，这是所有可能映射到 lab 的子序列概率之



[1]A. Graves. Supervised Sequence Labelling with Recurrent Neural Networks. Textbook, chapter7，Studies in Computational Intelligence, Springer, 2012.

循环神经网络RNN

- CTC

Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

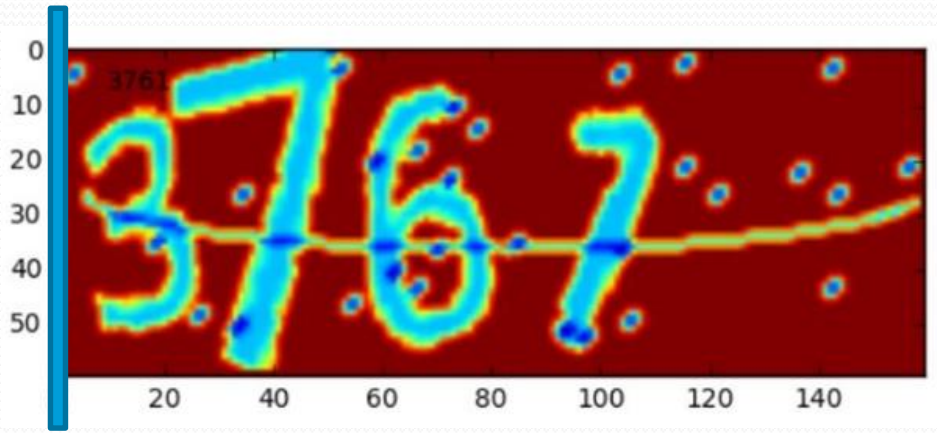
Alex Graves[1] ALEX@IDSIA.CH
Santiago Fernández[1] SANTIAGO@IDSIA.CH
Faustino Gomez[1] TINO@IDSIA.CH
Jürgen Schmidhuber[1,2] JUERGEN@IDSIA.CH

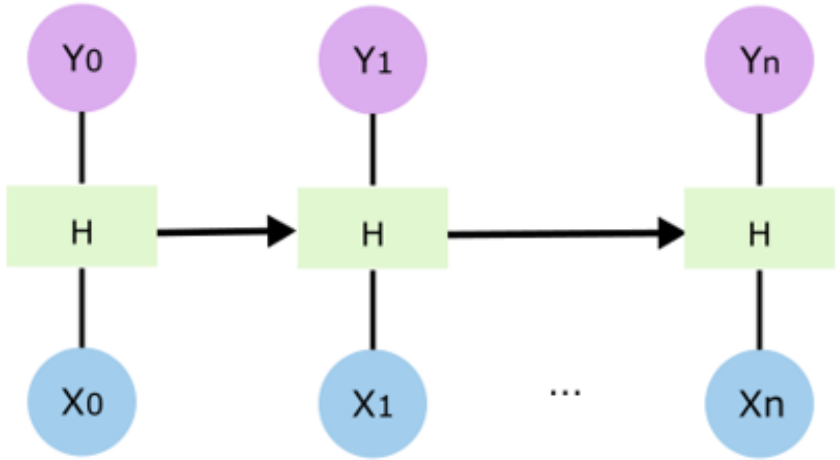
[1] Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland
[2] Technische Universität München (TUM), Boltzmannstr. 3, 85748 Garching, Munich, Germany

https://distill.pub/2017/ctc/

循环神经网络RNN

- CTC





实验

- 环境依赖 选项1 only linux Python2.7+TensorFlow+freetype-py
  - 该次实验基于以下环境以及依赖包，需要在Anaconda中搭建新的虚拟环境
    - Python2.7
      - conda install –n py27 python=2.7
      - conda activate py27
    - Tensorflow
      - conda install tensorflow 或者 conda install tensorflow-gpu
    - freetype-py
      - pip install freetype-py
    - Numpy
      - pip install numpy
    - Opencv
      - pip install opencv-python
    - PIL
      - pip install pillow

实验

- 环境依赖 选项2 for windows Python3.6+TensorFlow+freetype-py
  - 该次实验基于以下环境以及依赖包，需要在Anaconda中搭建新的虚拟环境
    - Python3.6
      - conda install –n py36 python=3.6
      - conda activate py36
    - Tensorflow
      - conda install tensorflow 或者 conda install tensorflow-gpu
    - freetype-py
      - pip install freetype-py
    - Numpy
      - pip install numpy
    - Opencv
      - pip install opencv-python
    - PIL
      - pip install pillow

实验

- 本次实验内容参考
  - https://www.jianshu.com/p/45828b18f133
  - https://github.com/jimmyleaf/ocr_tensorflow_cnn

- Github使用
  - 代码获取:
    - 安装git
    - git clone https://github.com/jimmyleaf/ocr_tensorflow_cnn.git

实验

Py3中语法兼容性修改：
1. xrange改成range；
2. genIdCard.py中，freetype.matrix()方法中参数 数字后缀L去掉；
3. Print 后加()

- github源码修改
  - 图像生成语句修改
    - 把文件tf_cnn_lstm_ctc.py的145行的obj.gen_image(True)里面的True删除
  - 字符生成属性修改
    - 把文件的genIDCard.py的120行的注释去掉，并把121行注释

```
120         #size = random.randint(1, self.max_size)
121         size = self.max_size
```

↓

```
120         size = random.randint(1, self.max_size)
121         #size = self.max_size
```

实验

- github源码修改
  - **启用训练**
    - **把tf_cnn_lstm_ctc.py文件的main函数的train函数的注释去掉**
  - 减少冗余(可选)
    - 把tf_cnn_lstm_ctc.py文件的第150行的print targets注释，否则每个batch都会打印一边batch产生的字符串
  - 添加训练停止条件语句
    - 如果想当达到一定的精度后返回，应该在哪里加？怎么加？
  - 添加模型参数保存语句
    - 没有保存模型文件的语句，应该在哪里加？

实验

- 参数定义

```
num_epochs = 10000

num_hidden = 64
num_layers = 1

obj = gen_id_card()

num_classes = obj.len + 1 + 1  # 10位数字 + blank + ctc blank
```

- LSTM模型定义

```
205     #定义LSTM网络
206     cell = tf.contrib.rnn.LSTMCell(num_hidden, state_is_tuple=True)
207     stack = tf.contrib.rnn.MultiRNNCell([cell] * num_layers, state_is_tuple=True)
208     outputs, _ = tf.nn.dynamic_rnn(cell, inputs, seq_len, dtype=tf.float32)
209
210     shape = tf.shape(inputs)
211     batch_s, max_timesteps = shape[0], shape[1]
212
213     outputs = tf.reshape(outputs, [-1, num_hidden])
214     W = tf.Variable(tf.truncated_normal([num_hidden,
215                                          num_classes],
216                                         stddev=0.1), name="W")
217     b = tf.Variable(tf.constant(0., shape=[num_classes]), name="b")
218
219     logits = tf.matmul(outputs, W) + b
220
221     logits = tf.reshape(logits, [batch_s, -1, num_classes])
222
223     logits = tf.transpose(logits, (1, 0, 2))
224
225     return logits, inputs, targets, seq_len, W, b
```


代码参考：Lab4_ocr_tensorflow_lstm_freetype

实验

- CTC Loss

```
236     loss = tf.nn.ctc_loss(labels=targets,inputs=logits, sequence_length=seq_len)
237     cost = tf.reduce_mean(loss)
238
239     #optimizer = tf.train.MomentumOptimizer(learning_rate=learning_rate,momentum=MOMENTUM).minimize(cost, global_step=global_step)
240     optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss,global_step=global_step)
241     decoded, log_prob = tf.nn.ctc_beam_search_decoder(logits, seq_len, merge_repeated=False)
```

- 训练

```
256     def do_batch():
257         train_inputs, train_targets, train_seq_len = get_next_batch(
            BATCH_SIZE)
258
259         feed = {inputs: train_inputs, targets: train_targets, seq_len:
            train_seq_len}
260
261         b_loss,b_targets, b_logits, b_seq_len,b_cost, steps, _ =
            session.run([loss, targets, logits, seq_len, cost, global_step,
             optimizer], feed)
262
```


代码参考：Lab4_ocr_tensorflow_lstm_freetype

实验

- CTC Loss

```
236     loss = tf.nn.ctc_loss(labels=targets,inputs=logits, sequence_length=seq_len)
237     cost = tf.reduce_mean(loss)
238
239     #optimizer = tf.train.MomentumOptimizer(learning_rate=learning_rate,momentum=MOMENTUM).minimize(cost, global_step=global_step)
240     optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss,global_step=global_step)
241     decoded, log_prob = tf.nn.ctc_beam_search_decoder(logits, seq_len, merge_repeated=False)
```

- 训练

```
256     def do_batch():
257         train_inputs, train_targets, train_seq_len = get_next_batch(
            BATCH_SIZE)
258
259         feed = {inputs: train_inputs, targets: train_targets, seq_len:
            train_seq_len}
260
261         b_loss,b_targets, b_logits, b_seq_len,b_cost, steps, _ =
            session.run([loss, targets, logits, seq_len, cost, global_step,
             optimizer], feed)
262
```

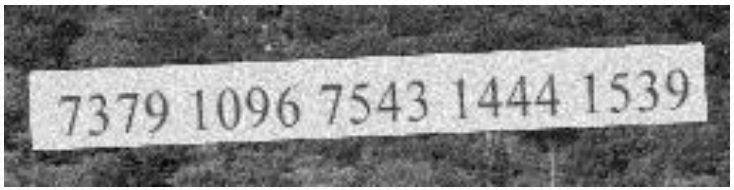
训练时：
用ctc_loss处理lstm的logits

- 预测

```
test_inputs,test_targets,test_seq_len = get_next_batch(
BATCH_SIZE)
test_feed = {inputs: test_inputs,
             targets: test_targets,
             seq_len: test_seq_len}
dd, log_probs, accuracy = session.run([decoded[0], log_prob,
acc], test_feed)
```

预测时：
用ctc_beam_search_decoder
处理logits

实验结果示例



73791096754314441539

编程练习

- 使用Lab3中的captcha， 或 FreeType 生成图片样本集，并通过LSTM+CTC 的网络结构完成识别
  - 要求
    - 字符个数不限(比如2-4个字符)
  - 选做附加题：生成数字+字母的组合序列码，进行识别

Thanks