0.1 Vanilla Recurrent Neural Network

Vanilla 计算单元

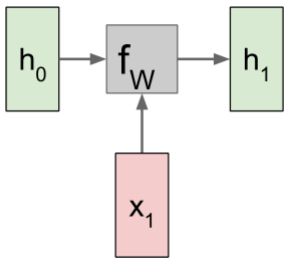


图 1: Vanilla Unit

计算方式:

$$h_t = f_W(h_{t-1}, x_t) \tag{1}$$

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{2}$$

$$y_t = W_{hy}h_t \tag{3}$$

- 公式1表示 RNN 在 t 时刻的隐藏状态向量 h_t 不仅和前一时刻状态 h_{t-1} 有关，而且还和 t 时刻输入 x_t 有关。
- 公式2是 RNN 隐藏状态的计算方式。
- 公式3是 RNN 单元的输出计算方式。

RNN 的计算图如下:

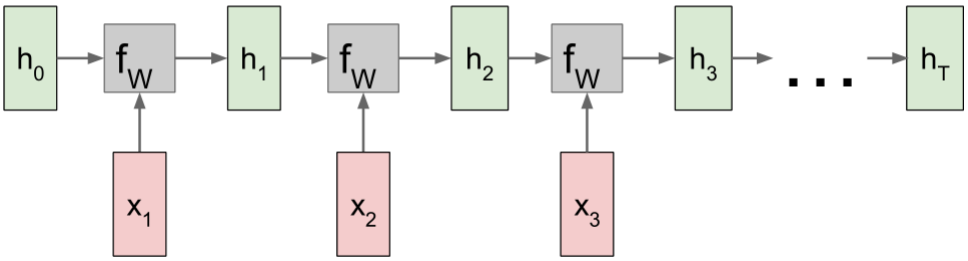


图 2: 循环神经网络的计算图

在每个时间步用相同的权重:

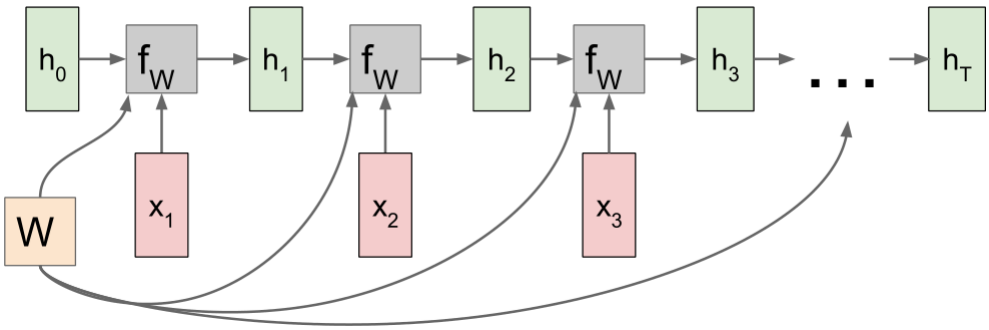


图 3: 循环神经网络的计算图（共享权重）

多对多的 RNN

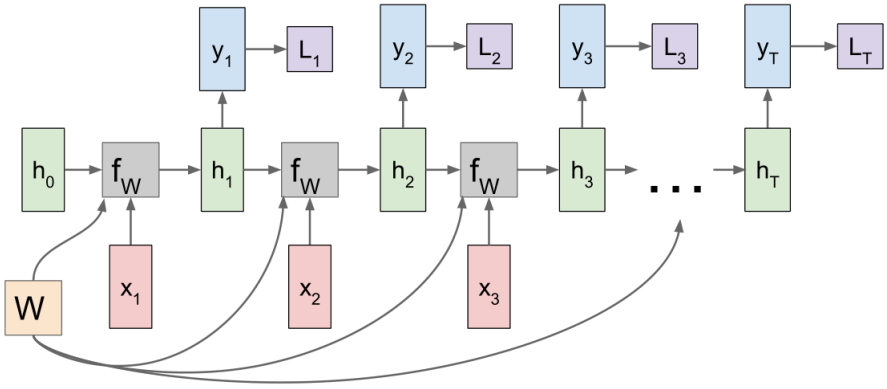


图 4: 多对多 RNN

多对一的计算图:

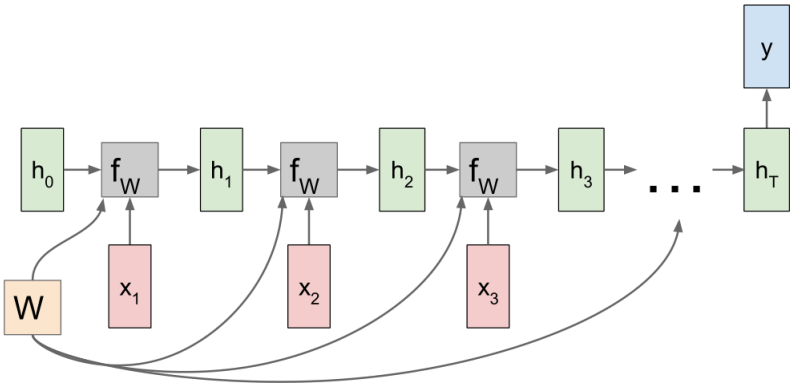


图 5: 多对一计算图

一对多的计算图:

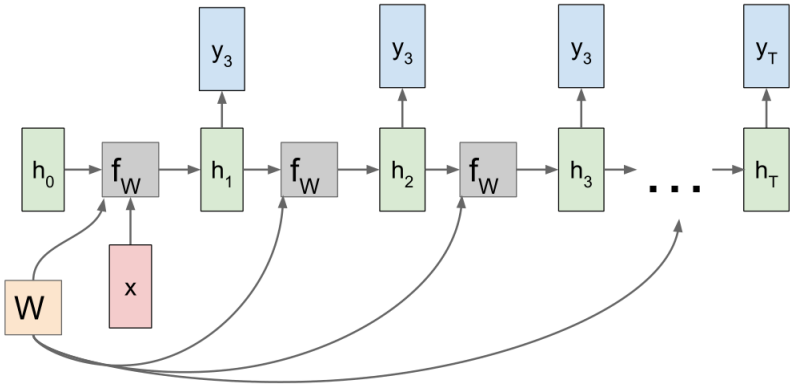


图 6: 一对多计算图

序列到序列（多对一到一对多）

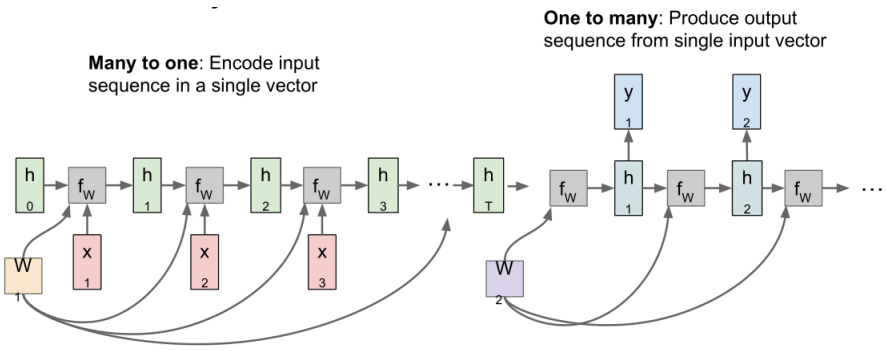


图 7: 序列到序列

反向传播过程:

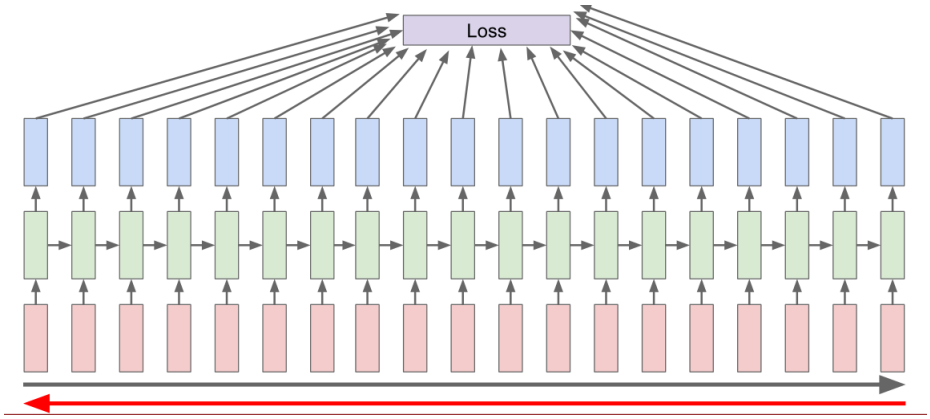


图 8: RNN 反向传递

截断的反向传播

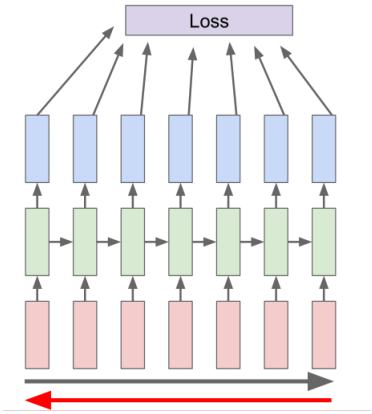


图 9: 截断的 RNN 反向传递 1

仅仅传播切断的序列而不是整个完整的序列。

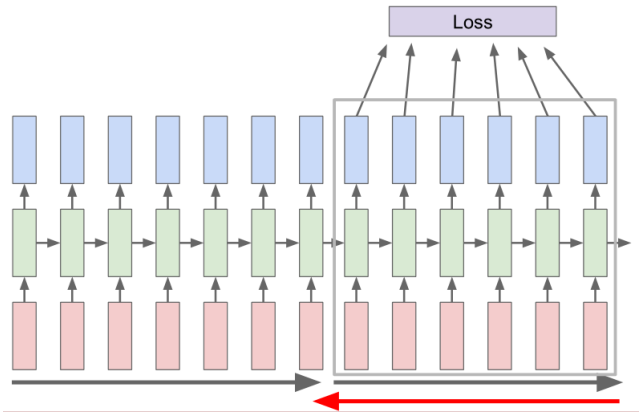


图 10: 截断的 RNN 反向传递 2

永远在时间步上更新隐藏状态，但是仅仅在一些小的时间步上反向传播。

应用:

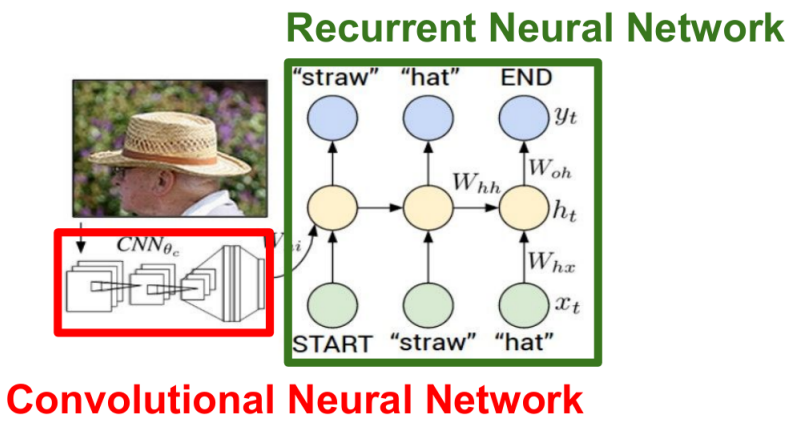


图 11: 图像的标注

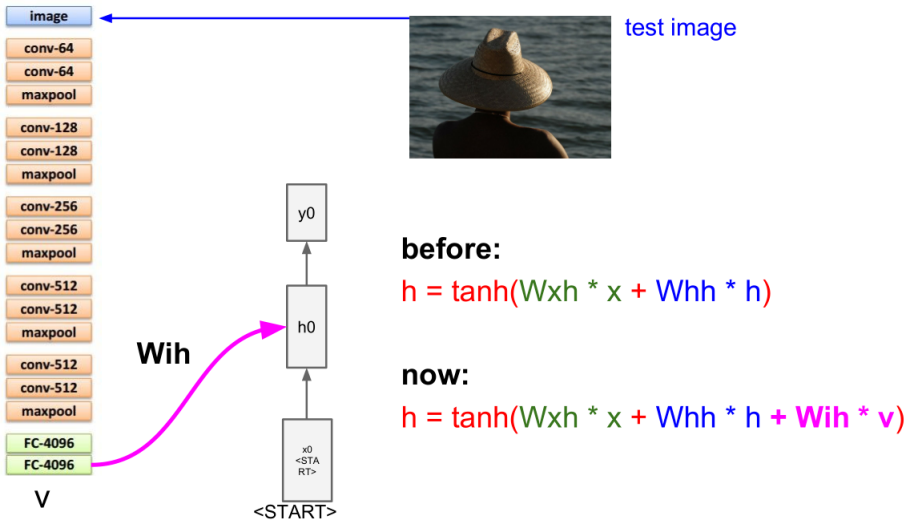


图 12: 图像的标注的处理

将 CNN 的 FC 层去掉，用 CNN 抽取的特征直接输入 RNN，RNN 的隐藏层的更新方式将发生变化, 之前的 $h = \tanh(W_{xh} \cdot x + W_{hh} \cdot h)$ 变为现在的 $h = \tanh(W_{xh} \cdot + W_{hh} \cdot h + Wih \cdot v)$

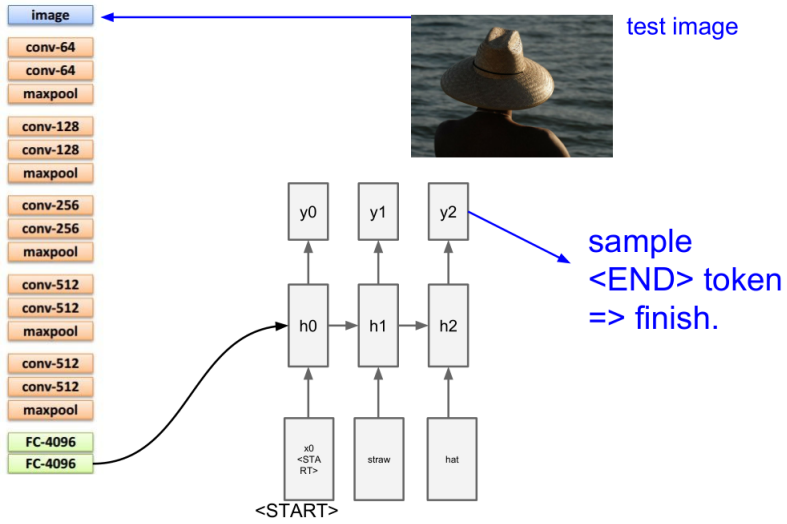


图 13: 图像的标注的处理

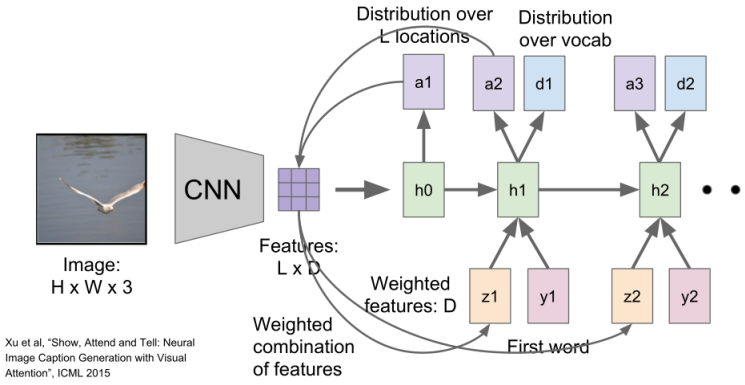


图 14: 图像的焦点标注的处理

vanilla rnn 的计算过程

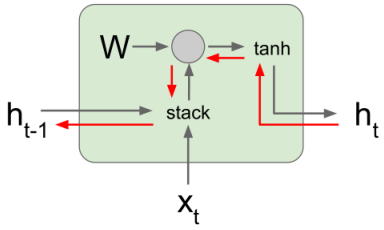


图 15: vanilla 反向传播

$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{bmatrix} W_{hh} & W_{hx} \end{bmatrix} \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right) \\
&= \tanh\left(\mathbf{W} \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right)
\end{aligned} \tag{4}
$$

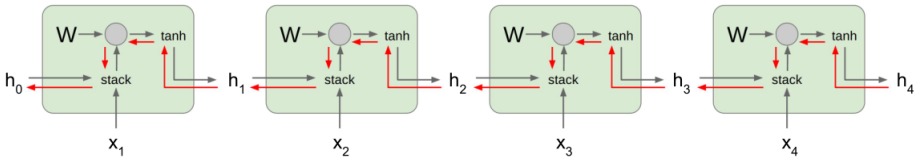


图 16: vanilla 梯度流

W 的最大奇异值 >1: 梯度爆炸 → 正规化

```
grad_norm = np.sum(grad*grad)
if grad_norm>threshold:
    grad *=(threshold/grad_norm)
```

W 的最大奇异值 <1: 梯度消失, 改变 RNN 结构。

0.2 LSTM

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \mathbf{W} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- f: 忘记门，是否删除 cell
- i: 输入门，是否写入 cell
- g:Gate 门，如何写入 cell
- o: 输出门，如何反映 cell

$RNN \rightarrow LSTM$

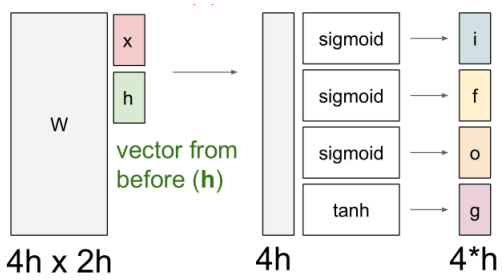


图 17: RNN 到 LSTM

LSTM 结构

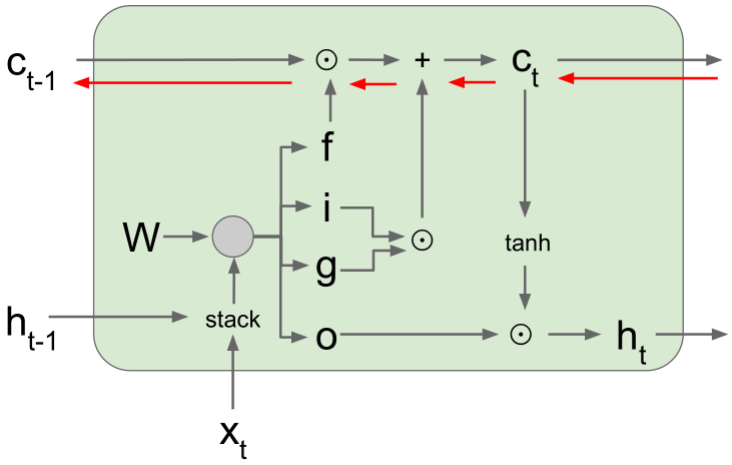


图 18: LSTM 结构

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \mathbf{W} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \tag{5}$$

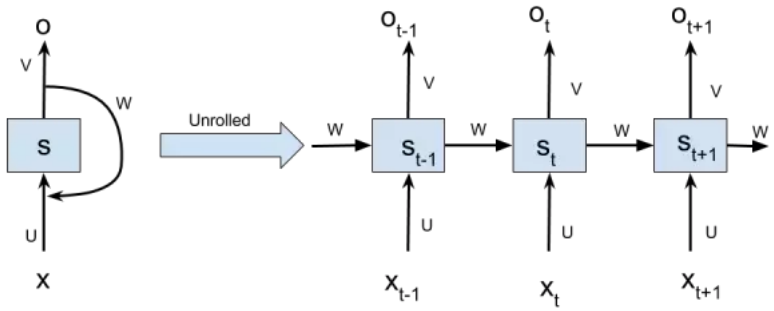
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM GRU

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz} + W_{hz}h_{t-1} + b_z) \\
\hat{h}_t &= \tanh(W_{xz}x_t + W_{hh}(r_{\odot}h_{t-1}) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h}_t
\end{aligned}
\tag{6}
$$

循环神经网络按时间轴展开的时候，如下图所示：



A Recurrent neural network unrolled through time.

图中：

1. x_t 代表时间步 t 的输入；
2. s_t 代表时间步 t 的隐藏状态，可看作该网络的「记忆」；
3. o_t 作为时间步 t 时刻的输出；
4. U、V、W 是所有时间步共享的参数，共享的重要性在于我们的模型在每一时间步以不同的输入执行相同的任务。

当把 RNN 展开的时候，网络可被看作每一个时间步都受上一时间步输出影响（时间步之间存在连接）的前馈网络。

两个注意事项

为了更顺利的进行实现，需要清楚两个概念的含义：

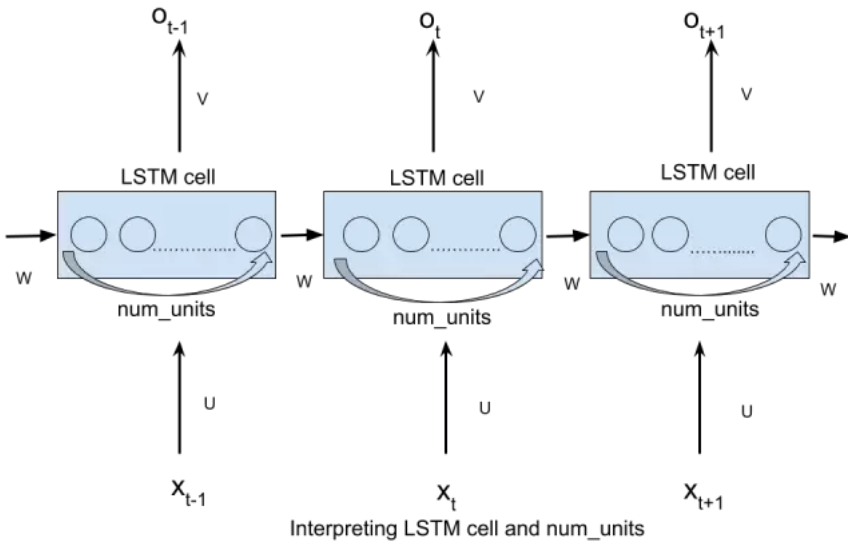
1. TensorFlow 中 LSTM 单元格的解释；
2. 数据输入 TensorFlow RNN 之前先格式化。

TensorFlow 中 LSTM 单元格的解释

在 TensorFlow 中，基础的 LSTM 单元格声明为：

tf.contrib.rnn.BasicLSTMCell(num_units)

这里，num_units 指一个 LSTM 单元格中的单元数。num_units 可以比作前馈神经网络中的隐藏层，前馈神经网络的隐藏层的节点数量等于每一个时间步中一个 LSTM 单元格内 LSTM 单元的 num_units 数量。下图可以帮助直观理解：



Interpreting LSTM cell and num_units

每一个 num_units LSTM 单元都可以看作一个标准的 LSTM 单元：

以上图表来自博客（地址），该博客有效介绍了 LSTM 的概念。

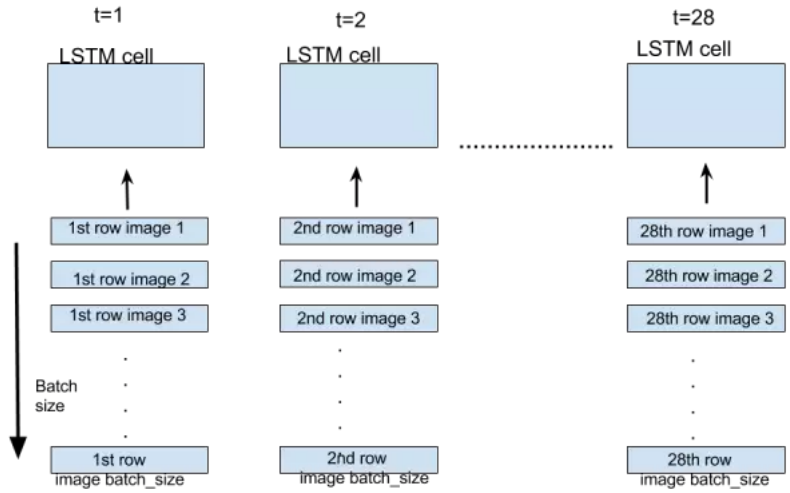
数据输入 TensorFlow RNN 之前先格式化

在 TensorFlow 中最简单的 RNN 形式是 static_rnn，在 TensorFlow 中定义如下：

tf.static_rnn(cell,inputs)

虽然还有其它的注意事项，但在这里我们仅关注这两个。

inputs 引数接受形态为 [batch_size,input_size] 的张量列表。列表的长度为将网络展开后的时间步数，即列表中每一个元素都分别对应网络展开的时间步。比如在 MNIST 数据集中，我们有 28x28 像素的图像，每一张都可以看成拥有 28 行 28 个像素的图像。我们将网络按 28 个时间步展开，以使在每一个时间步中，可以输入一行 28 个像素（input_size），从而经过 28 个时间步输入整张图像。给定图像的 batch_size 值，则每一个时间步将分别收到 batch_size 个图像。详见下图说明：



Visualization of inputs in Tensorflow RNNs

由 static_rnn 生成的输出是一个形态为 [batch_size,n_hidden] 的张量列表。列表的长度为将网络展开后的时间步数，即每一个时间步输出一个张量。在这个实现中我们只需关心最后一个时间步的输出，因为一张图像的所有行都输入到 RNN，预测即将在最后一个时间步生成。