组会报告

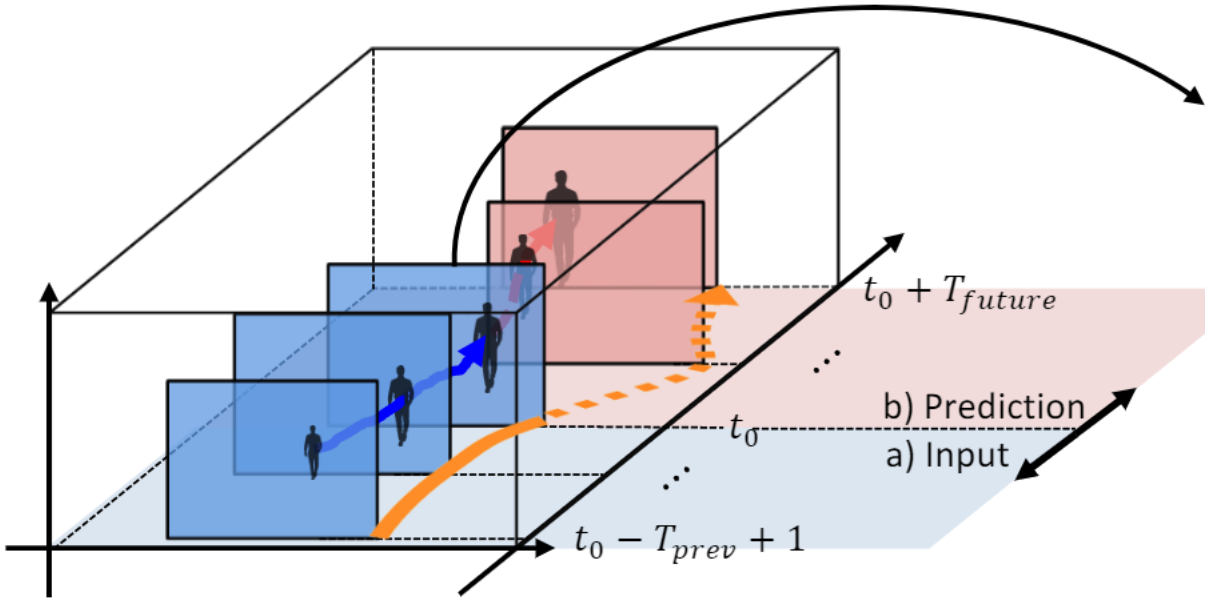
孙寒冰

- 目录：

- LSTM原理介绍
- Encoder-Decoder模型
- Attention机制
- 数据集处理方式

• 为什么用RNN

轨迹预测问题可以等价于序列生成问题，将过去的行人（或者其它目标）轨迹（坐标点、速度等其它变量）作为输入，求解的是未来的轨迹（坐标点的序列）。



将t_0作为当前时刻，
$[t_0 - T_{prev} + 1, t_0]$这一段时间内行人的轨迹作为输入，
$[t_0, t_0 + T_{future}]$这一段的轨迹作为预测目标

$$(x_0, y_0), (x_1, y_1) \dots (x_{t_o}, y_{t_0})$$

$$\downarrow$$

$$(x_{t_o+1}, y_{t_o+1}), (x_{t_o+2}, y_{t_o+2}) \dots (x_{t_o+T_{future}}, y_{t_0+T_{future}})$$

• RNN（循环神经网络）

◘ 概念

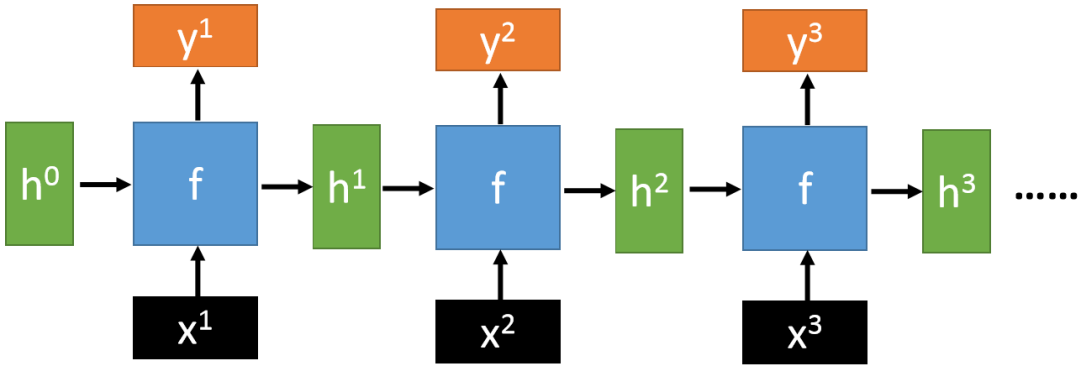
是具有记忆的前馈神经网络的推广，对于一个输入序列的每一步都执行相同的功能，并且当前的输出依赖于过去的每一步计算以及当前时刻的输入。也就是说，为了做出当前的决定，它考虑了当前的输入和从前的输入得到的输出。

其它神经网络的输入都是独立的，而RNN可以利用内部存储记忆以前的输入，因此所有的输入都是有关联的。

应用层面：多用于语言处理，比如机器翻译（many to many），图片标题生成（one to many），情感分类（many to one）等

• RNN（循环神经网络）

◘ 内部结构



$$h_t = f(W_{xh}x_t + W_{hh'}h_{t-1})$$
$$y_t = g(W_{hy}h_t)$$

参数：

x:作为一个输入序列，x_t 作为t时刻的输入值

h:作为隐藏层输出，h_t作为t时刻的隐藏层输出

y:作为经过RNN的输出

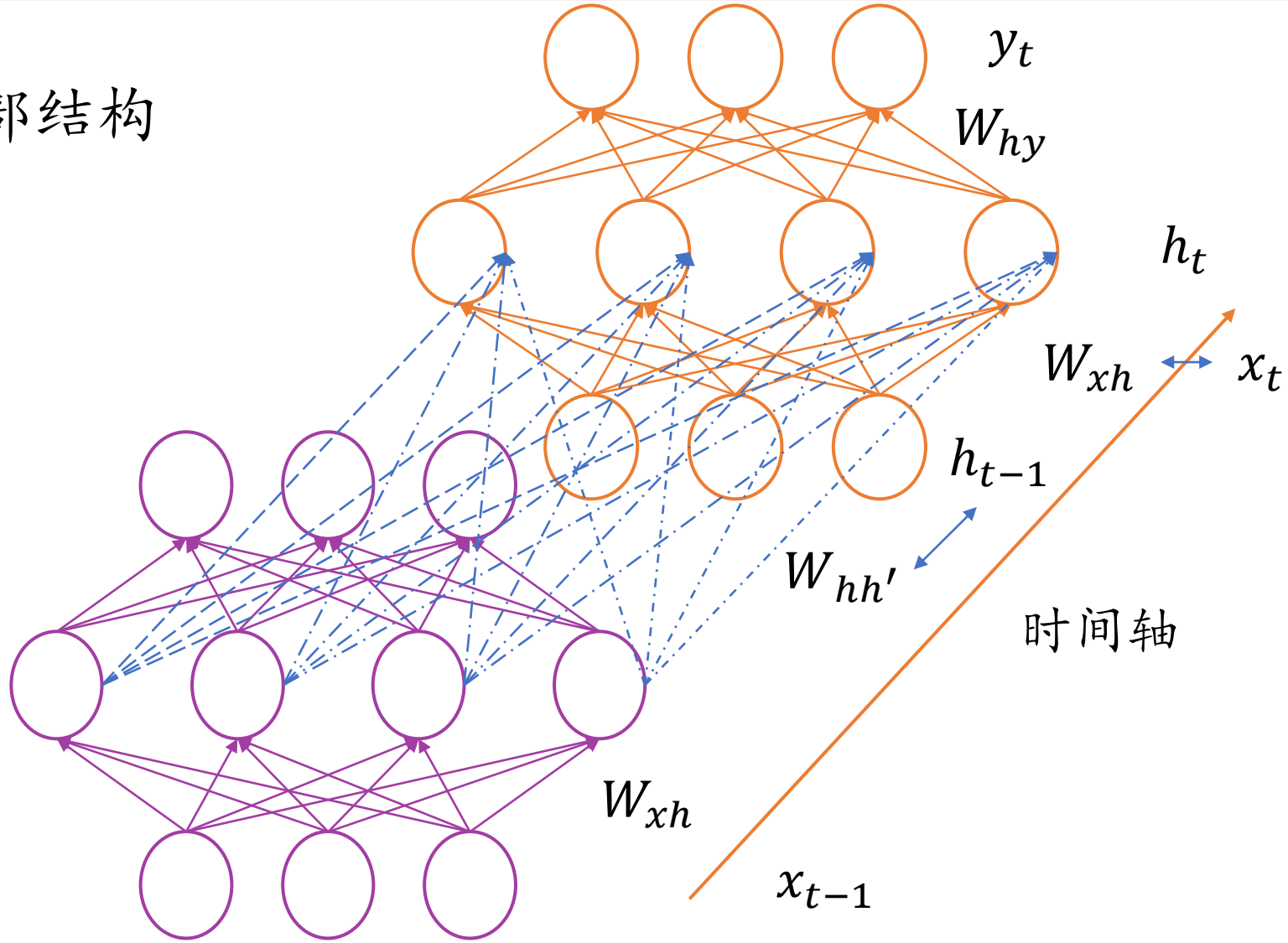
W_{xh}:从输入层到隐藏层的权重矩阵

$W_{hh'}$:从前一个隐藏层到后一个隐藏层的权重矩阵

W_{hy}:从隐藏层到输出层的权重

- **RNN**

内部结构


y_t
W_{hy}
h_t
W_{xh}
x_t
h_{t-1}
$W_{hh'}$
时间轴
W_{xh}
x_{t-1}
Output layer
Hidden layer
Input layer

• 从RNN过渡到LSTM

区别在于多了三个门，input gate, output gate, forget gate

□ 内部结构

参数：

x:作为一个输入序列，x_t 作为t时刻的输入值

h:作为隐藏层输出，h_t作为t时刻的隐藏层输出

y:作为经过LSTM的输出

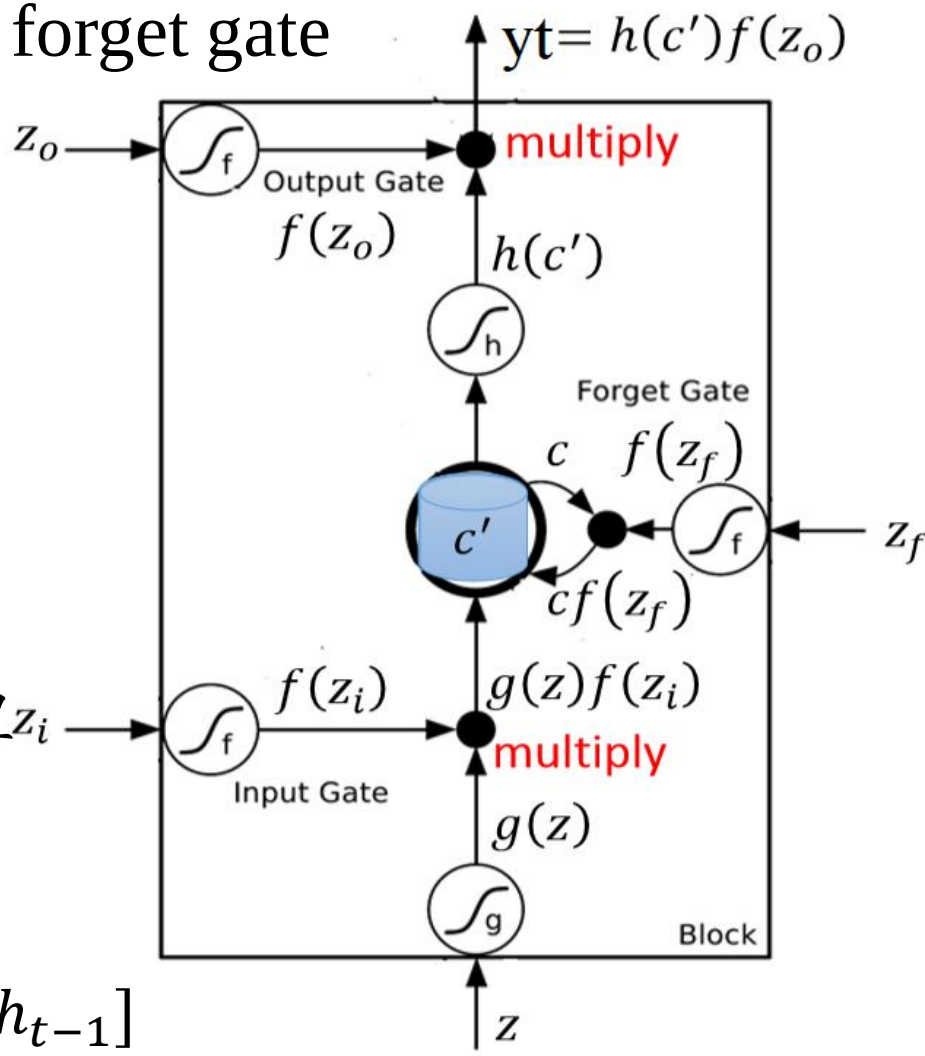
c:作为cell里面的值，随着时间步更替

z_i:input gate的计算值，决定是否将input输入到cell 里

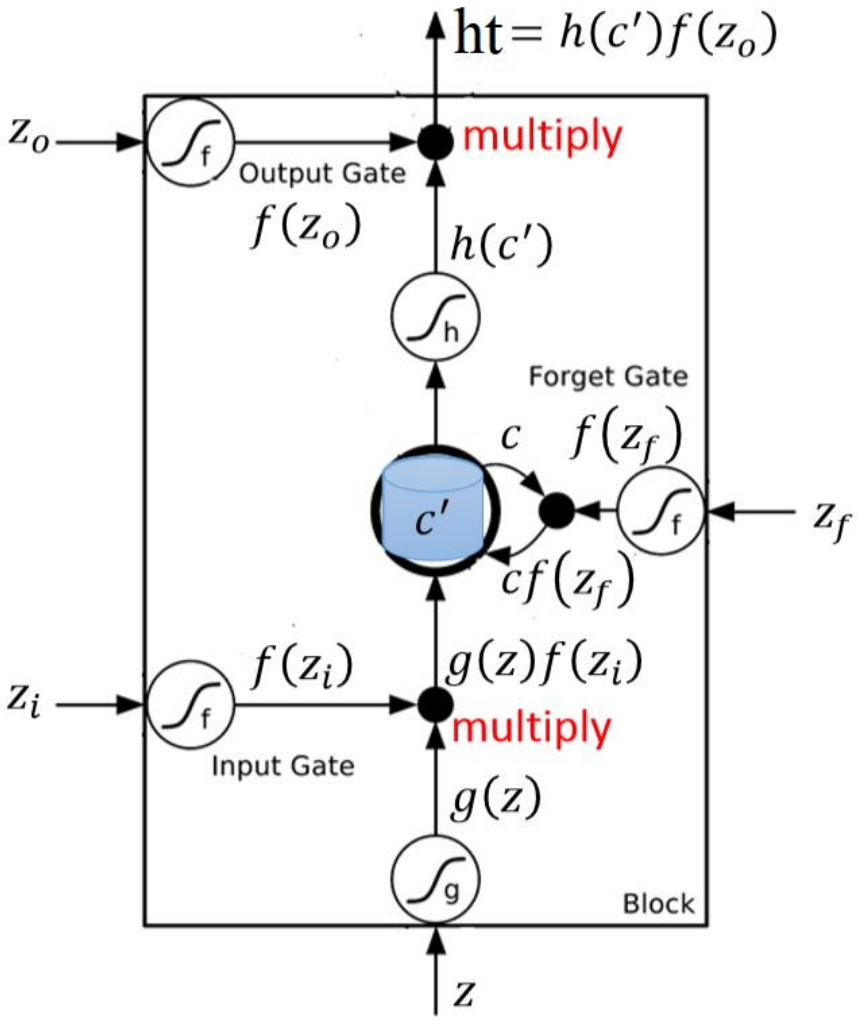
z_f:output gate的计算值，决定是否将cell里的值忘记

z_o:output gate的计算值，决定是否将cell里的值输出

z:经过向量拼接的输入值。即$[x_t, h_{t-1}, c_{t-1}]$或者$[x_t, h_{t-1}]$



• LSTM Cell



$$z_i = W_i \times [x_t, h_{t-1}]$$
$$z_o = W_o \times [x_t, h_{t-1}]$$
$$z_f = W_f \times [x_t, h_{t-1}]$$
$$z = W \times [x_t, h_{t-1}]$$

$$c_t = c_{t-1} \times f(z_f) + g(z) \times f(z_i)$$
$$h_t = f(z_o) \times \tanh(c_t)$$
$$y_t = f(W \times h_t)$$

f一般是sigmoid
g一般是tanh

• LSTM in keras

```
>>>tf.keras.layers.LSTM(units=num_units,
                        return_suquences=False,
                        return_state=False)
```

return_sequences=True, return_state=True	返回值有三个，分别是所有时间步的hidden vector, 最后一个时间步的hidden vector, 以及最后一个时间的cell state
return_sequences=True, return_state=False	返回值有一个，为所有时间步的hidden vector [batch_size, max_time, rnn_units]
return_sequences=False, return_state=True	返回值有三个，为最后一个时间步的hidden state*2，和最后一个时间步的cell state, [batch_size, rnn_units], [batch_size, rnn_units], [batch_size, rnn_units]
return_sequences=False, return_state=False	返回值有一个，为最后一个时刻的hidden vector[batch_size, rnn_units]

• 直接应用LSTM网络的例子

☐ Social-LSTM:

考虑因素比较少的行人轨迹预测，采用ETH和UCY数据集

将每一个行人的坐标值序列和该行人周围一定范围内的邻居的hidden state作为LSTM的输入，从而进行预测。

P.S. 回答上上次组会关于LSTM的小问题：

Q:行人可能在一系列的帧里不连续？

A:数据集做筛选的时候，可以把不连续的行人剔除出去，只选择在一段视频帧中都出现的行人。

（来源： https://github.com/t2kasa/social_lstm_keras_tf的相关数据处理方法）

- 目录：

• End-to-End

现在有关于轨迹预测应用较多的是端到端框架，对于直接利用坐标值进行预测的方法较少。

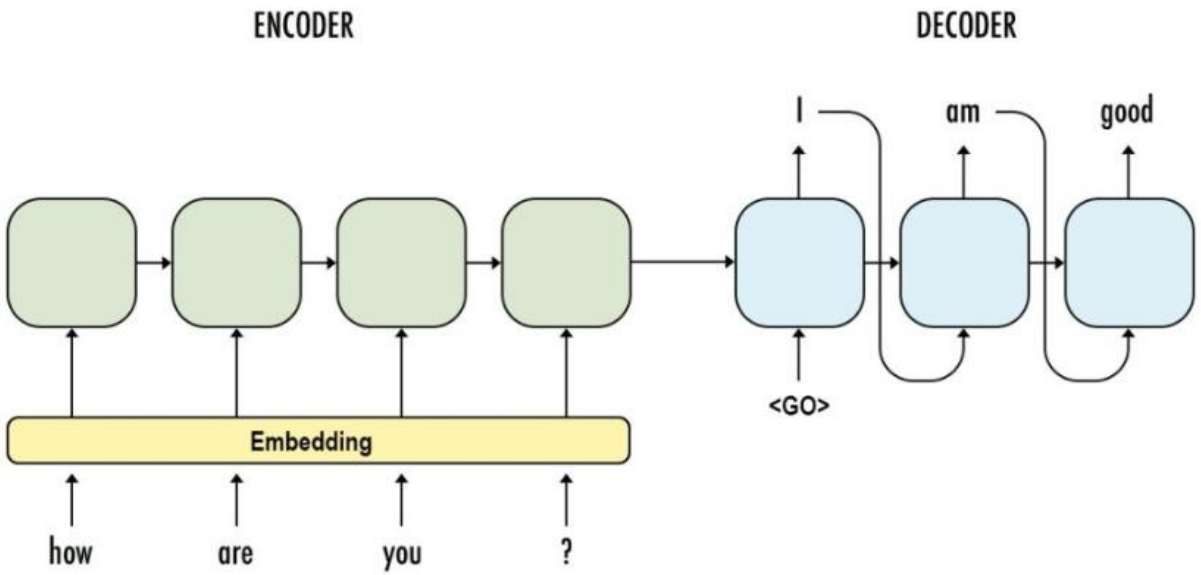
- 可以将LSTM的Encoder和Decoder结合，利用先编码再解码的方式进行预测

因此首先对Encoder-Decoder（又称Seq2Seq, End to End）进行简单介绍

• Encoder-Decoder

Encoder：将输入序列转成为一个固定长度的向量u

Decoder：将向量u转成为输出序列



在这里选择了一张机器翻译的利用该框架的示意图，实际上在轨迹预测中，原理差不多

• Encoder-Decoder (LSTM)

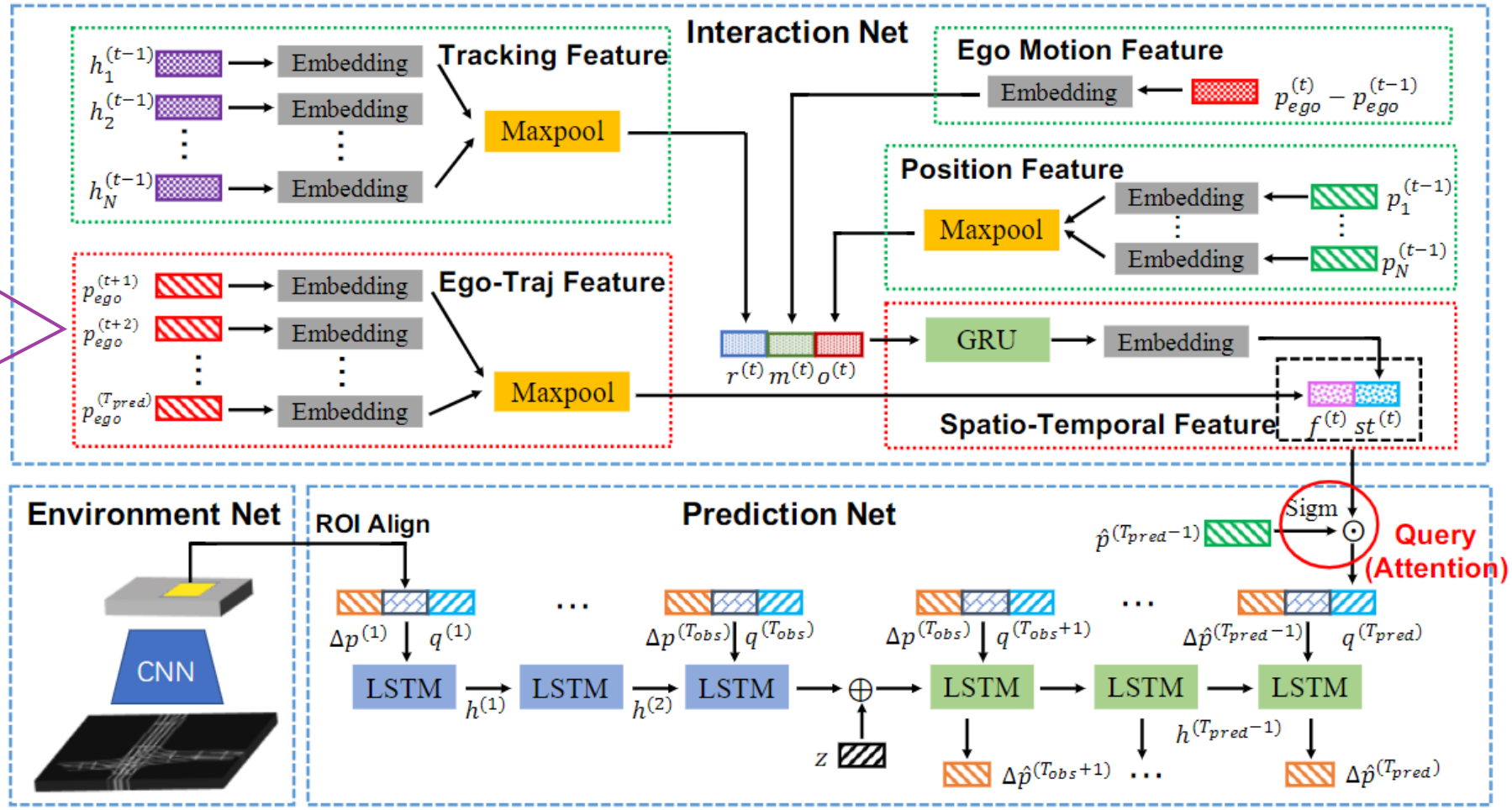
2020年ICRA轨迹预测冠军方法：

利用了encoder对行人历史轨迹进行编码，提取行人在动态环境中的运动模式，decoder利用了编码器中提取的行人特征，预测他们未来的轨迹分布，environment module可以视为一个encoder模块，作用是对周围环境特征进行提取。

Robust trajectory forecasting for multiple intelligent agents in dynamic scene. arXiv

• Encoders-Decoder (LSTM)

这部分由于是美团的外卖无人车计划，因此他有明确的目的地，所以他们未来计划作为一部分输入加入了



• Decoder (LSTM)

□ Interaction Net(Encoder)：在动态场景之下所有行人的交互特征。

有三个模块作为输入部分——（在t时刻进行编码时）

post trajectories: $o^{(t)} = Cat([e_{o,0}^{(t)}, e_{o,1}^{(t)}, e_{o,2}^{(t)} \dots e_{o,N}^{(t)}])$ # $e_{o,j}^{(t)}$为第j个行人在t时刻的坐标变换值

hidden states: $r^{(t)} = Cat([e_{r,0}^{(t)}, e_{r,1}^{(t)}, e_{r,2}^{(t)} \dots e_{r,N}^{(t)}])$ # $e_{r,j}^{(t)}$为第j个行人上一时刻的hidden vector变换值

the planned trajectory of ego-agent: $m^{(t)} = W_m(p_o^{(t)} - p_o^{(t-1)})$ # $p_o^{(t)}$为ego在t时刻的坐标值

以上三个部分concatenate之后作为encoder部分LSTM网络的输入

$h^t = GRU\left(Cat\left[o^{(t)}, r^{(t)}, m^{(t)}\right]\right)$　　输出为：　$st^t = Wh^t$

• Encoder-Decoder (LSTM)

- Environment Network(Encoder)：对周围环境进行特征提取，在任何时间步，该模块将道路图像I作为输入，并通过预先训练的ResNet18网络对环境进行编码

对于给定的agent（即被预测的目标），从获得的环境特征提取agent周围Rs米之内的信息，并且生成一个形状为$[C, K, K]$的环境特征矩阵G_i^t（C，K，K均为CNN网络参数）

最后经过处理得到$v_i^{(t)} = WG_i^t$

• Decoder (LSTM)

- Trajectory Prediction Network(Decoder)：在该部分引入了attention，也就是在Encoder给出的hidden state中，（Encoder对于整个画面的所有agents都进行的位置定位的计算，这个hidden state 包括了全部agents的信息）也就是说，首要关注离ego-agent（被预测目标）最近的影响因素。

attention计算：$q_i^{(t)} = (st^t)^T \odot e_{c,i}^{(t)}$

$e_{c,i}^{(t)} = \sigma(W p_i^t)$

#个人认为这里应该用h^t…

decoder计算：$h_D^t = LSTM([st_i^t, v_i^t, q_i^t])$

输出：$y_D^t = W h_D^t$

- **目录：**

- LSTM原理介绍
- Encoder-Decoder模型
- Attention机制
- 数据集处理方式

- **Attention**

简单介绍一下两种，在encoder-decoder框架中，attention可以将注意力集中到目前t时刻进行的计算周围。

在上个例子中，attention将注意力集中到目标agent周围对它影响可能比较大的环境中（，在机器翻译中，attention使当前翻译的词汇更注意到源语言里面意义相近的词汇周围。）

LuongAttention, BahdanauAttention 计算score公式不同:

BahdanauAttention:

$$score = v^T \tanh(W_1 \times h + W_2 \times d_t)$$

LuongAttention:

$$score = d_t \times W_1 \times h$$

- **Attention**

d_t是decoder当前时刻的向量，size为[1, rnn_units]

h是所有时刻encoder的输出，size为[max_time, rnn_units]

用d_t来和h每一个维度作比较，得出一个分数score, size为[max_time, 1]

#均未考虑batch，仅单个向量

$$\alpha_s = softmax(score)$$

axis=0上进行softmax

$$c_t = \sum_s \alpha_s h_s$$

用来加权计算出context vector, size为[1, rnn_units], 与原有的decoder输入进行拼接之后作为decoder的总体输入

- 目录：

- LSTM原理介绍
- Encoder-Decoder模型
- Attention机制
- 数据集处理

- **References**

[1]Future person localization in the future (paper)

[2] https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

[3] https://machinelearningmastery.com/return-sequences-and-return-states-for-lstms-in-keras/

[4] https://tech.meituan.com/2020/06/11/meituan-icra-2020.html

[5] Robust trajectory forecasting for multiple intelligent agents in dynamic scene

Thank you !