简介

conv lstm其实就是lstm的卷积版本，本质就是将lstm的权重矩阵(W)变为卷积核，而输入变成了三维图像

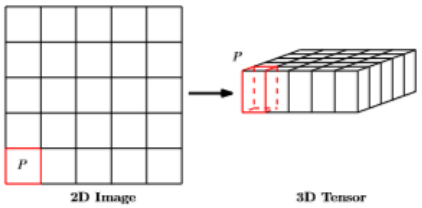

Figure 1: Transforming 2D image into 3D tensor

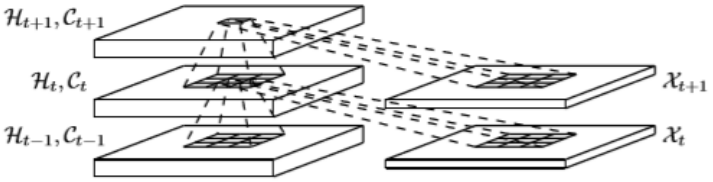

Figure 2: Inner structure of ConvLSTM

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

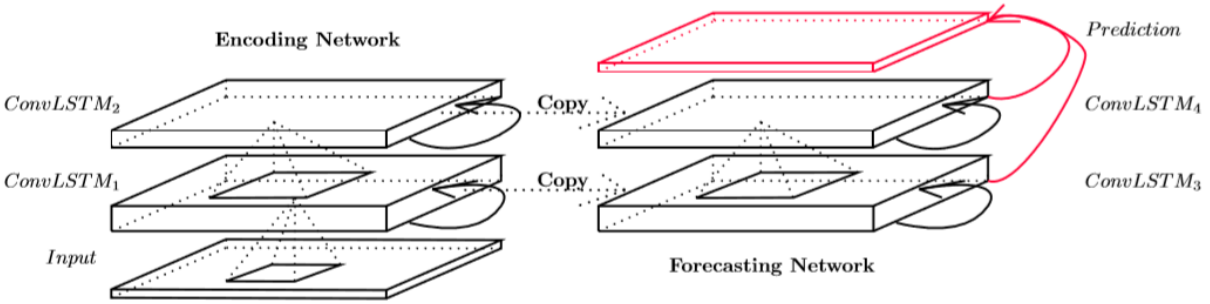

Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting

问题

1. hidden state的维度现在代表什么
2. pad 如何计算，convLstm的每次输出尺寸都和输入相同么
3. 计算的结果到底是什么，如何区分encode和decode阶段
4. torch.split,torch.stack
5. loss究竟如何计算

解答

1. hidden dim代表了一个convLstm模块卷积核W的维度，四个核会stack在一起进行卷积所以conv的输出维度为4 * hidden_dim
2. 相同，因为采用了kernel_size // 2的padding方法而且kernel_size为奇数
3. 在新的lstm框架中，cell state其实是原先的hidden state，在不同状态之间传递，而hideen_state其实是输出。因此可以发现cell_state作为隐藏状态在新旧状态之间传递，而hidden_state作为输出状态进入下一个状态的输入

```
h_cur, c_cur = cur_state

combined = torch.cat([input_tensor, h_cur], dim=1)  # concatenate along channel axis

combined_conv = self.conv(combined)
cc_i, cc_f, cc_o, cc_g = torch.split(combined_conv, self.hidden_dim, dim=1)
i = torch.sigmoid(cc_i)
f = torch.sigmoid(cc_f)
o = torch.sigmoid(cc_o)
g = torch.tanh(cc_g)

c_next = f * c_cur + i * g
h_next = o * torch.tanh(c_next)
```

原文中其实没有decoder这个模块，因为大小一直相同所以是forecasting network，其结构也是stack在一起的convLstm，之不过期intial cell and hidden state是从之前的encoder convLstm直接初始化得到的。最后的output产生方式是把forecasting各个层的输出拿出来concate在一起然后用

有点

能够充分利用到时空特性

缺点

因为卷积后的输出尺寸一直不变，没有pool的过程，其实保留了很多冗余信息，因此网络本身也做不了太深（太耗资源）

下一步方向

- C3D
- MCNet