1 Runoff prediction architecture

1.1 Encoder

Input quantity $X_k^t[x,y,\tau]$ consists of a N_τ long sequence (starting at $t-N_\tau+1$ and ending at time t) of N_k 2D maps with $N_x \times N_y$ pixels, i.e.

$$x \in [1, N_x] \tag{1}$$
$$y \in [1, N_y] \tag{2}$$
$$\tau \in [1, N_\tau] \tag{3}$$
$$k \in [1, N_k] \tag{4}$$

For any gate $g \in [i, f, o, c]$ define the gate input as a spatial convolution of the instantaneous input with a learnable "Markovian" weighting kernel plus a "non-Markovian" part that consists of the hidden state from the previous time instance convoluted with another learnable kernel

$$g_h^t[x,y,\tau] = \mathcal{M}_{hk}^g[\xi,\eta]\, X_k^t[x-\xi, y-\eta, \tau] \tag{5}$$
$$+\mathcal{N}_{hh'}^g[\xi,\eta]\, H_{h'}^t[x-\xi, y-\eta, \tau-1] + \mathcal{B}_h^g[x,y] \tag{6}$$

The auxiliary dimensions are

$$h, h' \in [1, N_h] \tag{7}$$
$$\xi \in [-(N_\xi - 1)/2, (N_\xi - 1)/2] \tag{8}$$
$$\eta \in [-(N_\eta - 1)/2, (N_\eta - 1)/2] \tag{9}$$

where the convolution kernel sizes N_ξ, N_η are assumed to be odd. Thus the adjustable objects have dimensions

$$\mathcal{M}_{hk}^g[\xi,\eta] \in \mathbb{R}^{4\times(N_h\times N_c)\times(N_\xi\times N_\eta)} \tag{10}$$
$$\mathcal{N}_{hh'}^g[\xi,\eta] \in \mathbb{R}^{4\times(N_h\times N_h)\times(N_\xi\times N_\eta)} \tag{11}$$
$$\mathcal{B}_h^g[x,y] \in \mathbb{R}^{4\times(N_h)\times(N_x\times N_y)} \tag{12}$$

We can now introduce compact matrix-vector and spatial convolution notation

$$\vec{g}^{\,t}[\tau] = \left(\boldsymbol{\mathcal{M}}^g, \boldsymbol{\mathcal{N}}^g\right) * \left(\vec{X}^t[\tau], \vec{H}^t[\tau-1]\right) + \vec{\boldsymbol{\mathcal{B}}}^g \tag{13}$$

The hidden state evolves according to the following iteration formula for LSTM networks

$$I_h^t[x,y,\tau] = \sigma(i_h^t[x,y,\tau]) \tag{14}$$
$$F_h^t[x,y,\tau] = \sigma(f_h^t[x,y,\tau]) \tag{15}$$
$$O_h^t[x,y,\tau] = \sigma(o_h^t[x,y,\tau]) \tag{16}$$
$$C_h^t[x,y,\tau] = F_h^t[x,y,\tau]\, C_h^t[x,y,\tau-1] + I_h^t[x,y,\tau]\, \tanh(c_h^t[x,y,\tau]) \tag{17}$$
$$H_h^t[x,y,\tau] = O_h^t[x,y,\tau]\, \tanh(C_h^t[x,y,\tau]) \tag{18}$$

going over $\tau \in [1, N_\tau]$ with initial conditions $H_h[x,y,0,t] = 0$ and $C_h[x,y,0,t] = 0$. The stated LSTM connections of the gates can be abbreviated by a functions L that returns the hidden network state for the next iteration step

$$\left(\vec{H}^t[\tau], \vec{C}^t[\tau]\right) = L\left(\left\{\vec{g}^{\,t}[\tau]\right\}, \vec{C}^t[\tau-1]\right) \tag{19}$$

1.2 Decoder

Take last hidden state of encoder $H_h^t[x,y,N_\tau]$ and use it as input for the decoder. Perform a single step (no iteration anymore!) with initial conditions $H_h^t[x,y,N_\tau]$ and $C_h^t[x,y,N_\tau]$, thus

$$\tilde{g}_h^t[x,y] = \tilde{\mathcal{M}}_{hh'}^g[\xi,\eta]\, H_{h'}^t[x-\xi, y-\eta, N_\tau] \tag{20}$$
$$+\tilde{\mathcal{N}}_{hh'}^g[\xi,\eta]\, H_{h'}^t[x-\xi, y-\eta, N_\tau] + \tilde{\mathcal{B}}_h^g[x,y] \tag{21}$$
$$= \tilde{\mathcal{Q}}_{hh'}^g[\xi,\eta]\, H_{h'}^t[x-\xi, y-\eta, N_\tau] + \tilde{\mathcal{B}}_h^g[x,y] \tag{22}$$

$$\tilde{I}^t_h[x,y] = \sigma(\tilde{i}^t_h[x,y]) \tag{23}$$
$$\tilde{F}^t_h[x,y] = \sigma(\tilde{f}^t_h[x,y]) \tag{24}$$
$$\tilde{O}^t_h[x,y] = \sigma(\tilde{o}^t_h[x,y]) \tag{25}$$
$$\tilde{C}^t_h[x,y] = \tilde{F}^t_h[x,y,\tau]\, C^t_h[x,y,N_\tau] + \tilde{I}^t_h[x,y]\, \tanh(\tilde{c}^t_h[x,y]) \tag{26}$$
$$\tilde{H}^t_h[x,y] = \tilde{O}^t_h[x,y]\, \tanh(\tilde{C}^t_h[x,y]) \tag{27}$$

The action of the decoder can be compactly written as

$$\left(\tilde{\vec{H}}^t, \tilde{\vec{C}}^t\right) = L\left(\{\tilde{\vec{g}}^t\}, \vec{C}^t[N_\tau]\right) \tag{28}$$

1.3 Fully connected layer

Choose $a \in [1,512]$, $b \in [1,256]$ then

$$R_r[t] = \mathscr{W}^3_{rb}\mathrm{ReLU}\left(\mathscr{W}^2_{ba}\mathrm{ReLU}\left(\mathscr{W}^1_{ah}[x,y]\tilde{H}^t_h[x,y]\right)\right) \tag{29}$$

with $r \in [1,97]$.

2 Interpretation of a linearized model

We can write our final result $R_r[t]$ for the r-th river as a function M_r (for model) which itself is a nested function call of the r-th component of the fully connected layer F, the decoder D and the encoder E on a given input time sequence $\{X^t_k[x,y,\tau]\}$, i.e.

$$\boxed{R_r[t] = M_r(\{X^t_k[x,y,\tau]\}) = F_r(D(E(\{X^t_k[x,y,\tau]\})))} \tag{30}$$

The idea is to approximate each $R_r[t]$ like

$$\boxed{R_r[t] \approx A_r[t] = a_r + \omega_{rk}[x,y,\tau]X^t_k[x,y,\tau]} \tag{31}$$

or expressed in continuous variables

$$A_r(t) = a_r + \int_0^{\tau_{\max}} \mathrm{d}\tau \int_{\mathbb{R}^2} \mathrm{d}x\mathrm{d}y\, \omega_{rk}(x,y,\tau)X^t_k(x,y,\tau) \tag{32}$$

which means that the runoff is approximately determined by the spatially dependent memory kernel that weights the influence of each time instance of each channel back to the time $t-\tau_{\max}$.

The offset a_r and the spatiotemporal weighting $\omega_{rk}[x,y,\tau]$ should be given by the first order Taylor expansion around a reference input sequence $\{X_k[x,y,\tau]\}$ that is not connected to any specific time t (thus no superscript t) i.e.

$$A_r[t] = M_r(\{X_k[x,y,\tau]\}) + \frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]}(X^t_k[x,y,\tau] - X_k[x,y,\tau]) \tag{33}$$

thus

$$\boxed{a_r = M_r(\{X_k[x,y,\tau]\})) - \frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]}X_k[x,y,\tau], \quad \omega_{rk}[x,y,\tau] = \frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]}} \tag{34}$$

The upcoming task is hence to calculate the derivative of the model network with respect to the input.

In order to keep the overview it is beneficial to write the output of each function in the nest as individual variables, i.e.

$$\{\epsilon_{sh}[x,y]\} = E(\{X_k[x,y,\tau]\}) \tag{35}$$
$$\{\tilde{H}_h[x,y]\} = D(\{\epsilon^s_h[x,y]\}) \tag{36}$$
$$\phi_r = F_r(\{\tilde{H}_h[x,y]\}) \tag{37}$$

where the additional index s in $\epsilon_{sh}[x,y]$ goes over the internal states of the encoder's output, i.e.

$$\boxed{\epsilon_{1h}[x,y] = C_h[x,y,N_\tau], \quad \epsilon_{2h}[x,y] = H_h[x,y,N_\tau]. \tag{38}}$$

Note that the superscript t is now omitted in the internal states since we calculate these quantities with respect to the reference sequence $\{X_k[x,y,\tau]\}$ that should not be connected to any specific time. Note further that the decoder's output state $\tilde{H}_h[x,y]$ has already been defined above.

As a first step we can now further evaluate the derivative via the chain rule

$$\omega_{rk}[x,y,\tau] = \frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]} = \frac{\partial \phi_r}{\partial \tilde{H}_{h''}[x'',y'']} \frac{\partial \tilde{H}_{h''}[x'',y'']}{\partial \epsilon_{s'h'}[x',y']} \frac{\partial \epsilon_{s'h'}[x',y']}{\partial X_k[x,y,\tau]} \tag{39}$$

The next task is to unfold the dependence of $\epsilon_{sh}[x,y]$ on $\{X_k[x,y,\tau]\}$. For that purpose we define the action of one ConvLSTM cell as function K. Then $\epsilon_{sh}[x,y]$ is the N_τ-fold nested function

$$\epsilon_{sh}[x,y] = E(\{X_k[x,y,\tau]\}) = K(\{X_k[x,y,N_\tau]\}, K(\{X_k[x,y,N_\tau-1]\}, \ldots, \{X_k[x,y,2]\}, K(\{X_k[x,y,1]\},0)\ldots)) \tag{40}$$

We can now define the output of the K-calls recursively for a fixed τ as

$$\{\kappa_{sh}[x,y,\tau]\} = K\left(\{X_k[x,y,\tau]\}, \{\kappa_{sh}[x,y,\tau-1]\}\right) \tag{41}$$

with $\kappa_{sh}[x,y,0] = 0$ and $\kappa_{sh}[x,y,N_\tau] = \epsilon_{sh}[x,y]$. Now we can use the chain rule to calculate the derivative of $\epsilon_{s'h'}[x',y']$ with respect to $X_k[x,y,\tau]$ for a specific τ

$$\frac{\partial \epsilon_{s'h'}[x',y']}{\partial X_k[x,y,\tau]} = \left(\prod_{j=\tau}^{N_\tau - 1} \frac{\partial \kappa_{s_{j+1}h_{j+1}}[x_{j+1}, y_{j+1}, j+1]}{\partial \kappa_{s_j h_j}[x_j, y_j, j]} \right) \frac{\partial \kappa_{s_\tau h_\tau}[x_\tau, y_\tau, \tau]}{\partial X_k[x,y,\tau]} \tag{42}$$

with the boundary conditions for the auxiliary indices $x_{N_\tau} \equiv x'$, $y_{N_\tau} \equiv y'$, $h_{N_\tau} \equiv h'$, $s_{N_\tau} \equiv s'$. This can be inserted into the formula for $\omega_{rk}[x,y,\tau]$ yielding

$$\boxed{\omega_{rk}[x,y,\tau] = \frac{\partial \phi_r}{\partial \tilde{H}_{h''}[x'',y'']} \frac{\partial \tilde{H}_{h''}[x'',y'']}{\partial \epsilon_{s_{N_\tau}h_{N_\tau}}[x_{N_\tau}, y_{N_\tau}]} \left(\prod_{j=\tau}^{N_\tau - 1} \frac{\partial \kappa_{s_{j+1}h_{j+1}}[x_{j+1}, y_{j+1}, j+1]}{\partial \kappa_{s_j h_j}[x_j, y_j, j]} \right) \frac{\partial \kappa_{s_\tau h_\tau}[x_\tau, y_\tau, \tau]}{\partial X_k[x,y,\tau]} \tag{43}}$$

At this point we have to evaluate the appearing derivatives explicitly. We start with the dependence of the κ at sequence time j on the previous κ at $j-1$.

$$\frac{\partial \kappa_{1h'}[x',y',j]}{\partial \kappa_{1h}[x,y,j-1]} = \frac{\partial C_{h'}[x',y',j]}{\partial C_h[x,y,j-1]} = F_{h'}[x',y',j]\delta_{hh'}\delta_{xx'}\delta_{yy'} \tag{44}$$

$$\frac{\partial \kappa_{2h'}[x',y',j]}{\partial \kappa_{1h}[x,y,j-1]} = \frac{\partial H_{h'}[x',y',j]}{\partial C_h[x,y,j-1]} = \frac{O_{h'}[x',y',j]F_{h'}[x',y',j]}{\cosh^2(C_{h'}[x',y',j])}\delta_{hh'}\delta_{xx'}\delta_{yy'} \tag{45}$$

$$\frac{\partial \kappa_{1h'}[x',y',j]}{\partial \kappa_{2h}[x,y,j-1]} = \frac{\partial C_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} = \dot{\sigma}(f_{h'}[x',y',j]) \frac{\partial f_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} C_{h'}[x',y',j-1] \tag{46}$$

$$+ \dot{\sigma}(i_{h'}[x',y',j]) \frac{\partial i_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} \tanh(c_{h'}[x',y',j]) \tag{47}$$

$$+ \frac{\sigma(i_{h'}[x',y',j])}{\cosh^2(c_{h'}[x',y',j])} \frac{\partial c_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} \tag{48}$$

$$\frac{\partial \kappa_{2h'}[x',y',j]}{\partial \kappa_{2h}[x,y,j-1]} = \frac{\partial H_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} = \dot{\sigma}(o_{h'}[x',y',j]) \frac{\partial o_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} \tanh(C_{h'}[x',y',j]) \tag{49}$$

$$+ \frac{\sigma(o_{h'}[x',y',j])}{\cosh^2(C_{h'}[x',y',j])} \frac{\partial C_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} \tag{50}$$

Note that $\partial\sigma(x)/\partial x = \sigma(x)(1-\sigma(x)) =: \dot{\sigma}(x)$ and $\partial \tanh(x)/\partial x = 1/\cosh^2(x)$. The remaining task is to calculate the derivative of the gate functions $g_h[x,y,j]$ with respect to the previous hidden state $H_h[x,y,j-1]$, i.e.

$$\frac{\partial g_{h'}[x',y',j]}{\partial H_h[x,y,j-1]} = \mathcal{N}^g_{h'h}[x'-x, y'-y] \tag{51}$$

Then we finally have

$$\frac{\partial \kappa_{1h'}[x',y',j]}{\partial \kappa_{1h}[x,y,j-1]} = F_{h'}[x',y',j]\delta_{hh'}\delta_{xx'}\delta_{yy'} \tag{52}$$

$$\frac{\partial \kappa_{2h'}[x',y',j]}{\partial \kappa_{1h}[x,y,j-1]} = \frac{O_{h'}[x',y',j]F_{h'}[x',y',j]}{\cosh^2(C_{h'}[x',y',j])}\delta_{hh'}\delta_{xx'}\delta_{yy'} \tag{53}$$

$$\frac{\partial \kappa_{1h'}[x',y',j]}{\partial \kappa_{2h}[x,y,j-1]} = \dot\sigma(f_{h'}[x',y',j])C_{h'}[x',y',j-1]\mathcal{N}^f_{h'h}[x'-x,y'-y] \tag{54}$$

$$+\dot\sigma(i_{h'}[x',y',j])\tanh(c_{h'}[x',y',j])\mathcal{N}^i_{h'h}[x'-x,y'-y] \tag{55}$$

$$+\frac{\sigma(i_{h'}[x',y',j])}{\cosh^2(c_{h'}[x',y',j])}\mathcal{N}^c_{h'h}[x'-x,y'-y] \tag{56}$$

$$\frac{\partial \kappa_{2h'}[x',y',j]}{\partial \kappa_{2h}[x,y,j-1]} = \dot\sigma(o_{h'}[x',y',j])\tanh(C_{h'}[x',y',j])\mathcal{N}^o_{h'h}[x'-x,y'-y] \tag{57}$$

$$+\frac{\sigma(o_{h'}[x',y',j])}{\cosh^2(C_{h'}[x',y',j])}\frac{\partial \kappa_{1h'}[x',y',j]}{\partial \kappa_{2h}[x,y,j-1]} \tag{58}$$

The next task is to calculate the dependence of the κ on the input quantity X.

$$\frac{\partial \kappa_{1h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \frac{\partial C_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \dot\sigma(f_{h'}[x',y',\tau])\frac{\partial f_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]}C_{h'}[x',y',\tau-1] \tag{59}$$

$$+\dot\sigma(i_{h'}[x',y',\tau])\frac{\partial i_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]}\tanh(c_{h'}[x',y',\tau]) \tag{60}$$

$$+\frac{\sigma(i_{h'}[x',y',\tau])}{\cosh^2(c_{h'}[x',y',\tau])}\frac{\partial c_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} \tag{61}$$

$$\frac{\partial \kappa_{2h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \frac{\partial H_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \dot\sigma(o_{h'}[x',y',\tau])\frac{\partial o_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]}\tanh(C_{h'}[x',y',\tau]) \tag{62}$$

$$+\frac{\sigma(o_{h'}[x',y',\tau])}{\cosh^2(C_{h'}[x',y',\tau])}\frac{\partial C_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} \tag{63}$$

The remaining task is to calculate the derivative of the gate functions $g_h[x,y,\tau]$ with respect to the current input $X_k[x,y,\tau]$, i.e.

$$\frac{\partial g_{h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \mathcal{M}^g_{h'k}[x'-x,y'-y] \tag{64}$$

Inserted into the equations above yields

$$\frac{\partial \kappa_{1h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \dot\sigma(f_{h'}[x',y',\tau])C_{h'}[x',y',\tau-1]\mathcal{M}^f_{h'k}[x'-x,y'-y] \tag{65}$$

$$+\dot\sigma(i_{h'}[x',y',\tau])\tanh(c_{h'}[x',y',\tau])\mathcal{M}^i_{h'k}[x'-x,y'-y] \tag{66}$$

$$+\frac{\sigma(i_{h'}[x',y',\tau])}{\cosh^2(c_{h'}[x',y',\tau])}\mathcal{M}^c_{h'k}[x'-x,y'-y] \tag{67}$$

$$\frac{\partial \kappa_{2h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} = \dot\sigma(o_{h'}[x',y',\tau])\tanh(C_{h'}[x',y',\tau])\mathcal{M}^o_{h'k}[x'-x,y'-y] \tag{68}$$

$$+\frac{\sigma(o_{h'}[x',y',\tau])}{\cosh^2(C_{h'}[x',y',\tau])}\frac{\partial \kappa_{1h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} \tag{69}$$

Now that we have expressions for the derivatives we can give them names

$$\Lambda_{s's,h'h}[x',x,y',y,j] = \frac{\partial \kappa_{s'h'}[x',y',j]}{\partial \kappa_{sh}[x,y,j-1]} \tag{70}$$

$$\lambda_{s'h',k}[x',x,y',y,\tau] = \frac{\partial \kappa_{s'h'}[x',y',\tau]}{\partial X_k[x,y,\tau]} \tag{71}$$

and insert it into the full expression

$$\omega_{rk}[x,y,\tau] = \frac{\partial \phi_r}{\partial \tilde{H}_{h''}[x'',y'']} \frac{\partial \tilde{H}_{h''}[x'',y'']}{\partial \epsilon_{s_{N_\tau} h_{N_\tau}}[x_{N_\tau},y_{N_\tau}]} \left(\prod_{j=\tau}^{N_\tau - 1} \Lambda_{s_{j+1}s_j,h_{j+1}h_j}[x_{j+1},x_j,y_{j+1},y_j,j+1] \right) \lambda_{s_\tau h_\tau,k}[x_\tau,x,y_\tau,y,\tau] \tag{72}$$

To further shorten the expression we can formally carry out the product as

$$\chi_{s_n h_n,k}[x_n,x,y_n,y,n,\tau] = \left(\prod_{j=\tau}^{n-1} \Lambda_{s_{j+1}s_j,h_{j+1}h_j}[x_{j+1},x_j,y_{j+1},y_j,j+1] \right) \lambda_{s_\tau h_\tau,k}[x_\tau,x,y_\tau,y,\tau] \tag{73}$$

with $\tau \le n \le N_\tau$ then

$$\boxed{\chi_{s_{n+1}h_{n+1},k}[x_{n+1},x,y_{n+1},y,n+1,\tau] = \Lambda_{s_{n+1}s_n,h_{n+1}h_n}[x_{n+1},x_n,y_{n+1},y_n,n+1]\chi_{s_n h_n,k}[x_n,x,y_n,y,n,\tau]} \tag{74}$$

for $n = \tau, \ldots N_\tau - 1$ and $\chi_{s_\tau h_\tau,k}[x_\tau,x,y_\tau,y,\tau,\tau] = \lambda_{s_\tau h_\tau,k}[x_\tau,x,y_\tau,y,\tau]$.

The next task is to evaluate the derivative of the decoder with respect to the encoders output. Since the decoder is itself a ConvLSTM cell with input from encoders last hidden state we can write

$$\{\tilde{H}_h[x,y]\} = \tilde{K}_2\left(\{\epsilon_{2h}[x,y]\},\{\epsilon_{sh}[x,y]\}\right) \tag{75}$$

Following the same line of reasoning as above we obtain for the derivatives

$$\boxed{\begin{aligned}
\frac{\partial \tilde{H}_{h'}[x',y']}{\partial \epsilon_{1h}[x,y]} = \frac{\partial \tilde{H}_{h'}[x',y']}{\partial C_h[x,y,N_\tau]} &= \frac{\tilde{O}_{h'}[x',y']\tilde{F}_{h'}[x',y']}{\cosh^2(\tilde{C}_{h'}[x',y'])}\delta_{hh'}\delta_{xx'}\delta_{yy'} && (76)\\
\frac{\partial \tilde{H}_{h'}[x',y']}{\partial \epsilon_{2h}[x,y]} = \frac{\partial \tilde{H}_{h'}[x',y']}{\partial H_h[x,y,N_\tau]} &= \dot{\sigma}(\tilde{o}_{h'}[x',y'])\tanh(\tilde{C}_{h'}[x',y'])\tilde{\mathcal{Q}}^o_{h'h}[x'-x,y'-y] && (77)\\
&+ \frac{\sigma(\tilde{o}_{h'}[x',y'])}{\cosh^2(\tilde{C}_{h'}[x',y'])}\tilde{\Lambda}_{1,2,h'h}[x',x,y',y] && (78)
\end{aligned}}$$

with

$$\begin{aligned}
\tilde{\Lambda}_{1,2,h'h}[x',x,y',y] &= \dot{\sigma}(\tilde{f}_{h'}[x',y'])C_{h'}[x',y',N_\tau]\tilde{\mathcal{Q}}^f_{h'h}[x'-x,y'-y] && (79)\\
&+ \dot{\sigma}(\tilde{i}_{h'}[x',y'])\tanh(\tilde{c}_{h'}[x',y'])\tilde{\mathcal{Q}}^i_{h'h}[x'-x,y'-y] && (80)\\
&+ \frac{\sigma(\tilde{i}_{h'}[x',y'])}{\cosh^2(\tilde{c}_{h'}[x',y'])}\tilde{\mathcal{Q}}^c_{h'h}[x'-x,y'-y] && (81)
\end{aligned}$$

Now we can also rename these derivatives as

$$\boxed{\tilde{\chi}_{s_{N_\tau} h_{N_\tau},h''}[x'',x_{N_\tau},y'',y_{N_\tau}] = \frac{\partial \tilde{H}_{h''}[x'',y'']}{\partial \epsilon_{s_{N_\tau} h_{N_\tau}}[x_{N_\tau},y_{N_\tau}]}} \tag{82}$$

and insert into the full expression

$$\omega_{rk}[x,y,\tau] = \frac{\partial \phi_r}{\partial \tilde{H}_{h''}[x'',y'']}\tilde{\chi}_{s_{N_\tau} h_{N_\tau},h''}[x'',x_{N_\tau},y'',y_{N_\tau}]\chi_{s_{N_\tau} h_{N_\tau},k}[x_{N_\tau},x,y_{N_\tau},y,N_\tau,\tau] \tag{83}$$

If we define the product

$$\boxed{\Xi_{h''k}[x'',x,y'',y,\tau] = \tilde{\chi}_{s_{N_\tau} h_{N_\tau},h''}[x'',x_{N_\tau},y'',y_{N_\tau}]\chi_{s_{N_\tau} h_{N_\tau},k}[x_{N_\tau},x,y_{N_\tau},y,N_\tau,\tau]} \tag{84}$$

we can write

$$\boxed{\omega_{rk}[x,y,\tau] = \frac{\partial \phi_r}{\partial \tilde{H}_{h''}[x'',y'']}\Xi_{h''k}[x'',x,y'',y,\tau]} \tag{85}$$

The remaining task is to find the derivative of the fully connected laysER with respect to the decoder's output. For this purpose we define gain the output variables of the nested ReLU functions as individual variables, i.e.

$$\rho^1_a = \mathscr{W}^1_{ah}[x,y]\tilde{H}_h[x,y] \tag{86}$$
$$\rho^2_b = \mathscr{W}^2_{ba}\mathrm{ReLU}\left(\rho^1_a\right) \tag{87}$$
$$\phi_r = \mathscr{W}^3_{rb}\mathrm{ReLU}\left(\rho^2_b\right) \tag{88}$$

With these definitions we can write after using the chain rule

$$\frac{\partial\phi_r}{\partial\tilde{H}_{h''}[x'',y'']} = \frac{\partial\phi_r}{\partial\rho^2_{b'}}\frac{\partial\rho^2_{b'}}{\partial\rho^1_{a'}}\frac{\partial\rho^1_{a'}}{\partial\tilde{H}_{h''}[x'',y'']} \tag{89}$$

The individual derivatives can be easily carried out by noting $\partial\mathrm{ReLU}(x)/\partial x = \theta(x)$ which is the Heaviside step function

$$\frac{\partial\phi_r}{\partial\rho^2_{b'}} = \mathscr{W}^3_{rb'}\theta\left(\rho^2_{b'}\right) \tag{90}$$
$$\frac{\partial\rho^2_{b'}}{\partial\rho^1_{a'}} = \mathscr{W}^2_{b'a'}\theta\left(\rho^1_{a'}\right) \tag{91}$$
$$\frac{\partial\rho^1_{a'}}{\partial\tilde{H}_{h''}[x'',y'']} = \mathscr{W}^1_{a'h''}[x'',y''] \tag{92}$$

$$\frac{\partial\phi_r}{\partial\tilde{H}_{h''}[x'',y'']} = \mathscr{W}^3_{rb'}\theta\left(\rho^2_{b'}\right)\cdot\mathscr{W}^2_{b'a'}\theta\left(\rho^1_{a'}\right)\cdot\mathscr{W}^1_{a'h''}[x'',y''] = \Omega_{rh''}[x'',y''] \tag{93}$$

As the final result we can then write

$$\omega_{rk}[x,y,\tau] = \Omega_{rh''}[x'',y'']\cdot\Xi_{h''k}[x'',x,y'',y,\tau] \tag{94}$$

2.1 Recipe

Evaluating the result from the right to the left suggests the following recipe:

- take a specific reference sequence $\{X_k[x,y,\tau]\}$ and go over $\tau\in[1,N_\tau]$ in ascending order
  - calculate for each τ
  $$\chi_{s_\tau h_\tau,k}[x_\tau,x,y_\tau,y,\tau,\tau] = \lambda_{s_\tau h_\tau,k}[x_\tau,x,y_\tau,y,\tau]$$
  where λ is a function of the current gates $g_{h_\tau}[x_\tau,y_\tau,\tau]$, the last cell state $C_{h_\tau}[x_\tau,y_\tau,\tau-1]$ and the parameters $\mathscr{M}^g_{h_\tau k}[x_\tau-x,y_\tau-y]$,

    expense: $N_k(2N_hN_x^2N_y^2)$

  - go then over $n\in[\tau,N_\tau-1]$ in ascending order
    * for each n calculate the tensor $\Lambda_{s_{n+1}s_n,h_{n+1}h_n}[x_{n+1},x_n,y_{n+1},y_j,n+1]$ as a function of the input gates and hidden states of the $n+1$-th and n-th ConvLSTM cells and the parameters $\mathscr{M}^g_{h_\tau k}[x_\tau-x,y_\tau-y]$ and $\mathscr{N}^g_{h_\tau k}[x_\tau-x,y_\tau-y]$ and perform the summation over s_n,h_n,x_n,y_n for each $k,s_{n+1},h_{n+1},x,x_{n+1},y,y_{n+1}$
    $$\chi_{s_{n+1}h_{n+1},k}[x_{n+1},x,y_{n+1},y,n+1,\tau] = \Lambda_{s_{n+1}s_n,h_{n+1}h_n}[x_{n+1},x_n,y_{n+1},y_n,n+1]\chi_{s_nh_n,k}[x_n,x,y_n,y,n,\tau]$$

      expense: $N_k(2N_hN_x^2N_y^2)(2N_hN_xN_y)$

  - the final result is $\chi_{s_{N_\tau}h_{N_\tau},k}[x_{N_\tau},x,y_{N_\tau},y,N_\tau,\tau]$

    accumulated expense: $N_k(2N_hN_x^2N_y^2)+(N_\tau-\tau)N_k(2N_hN_x^2N_y^2)(2N_hN_xN_y)$

  - calculate $\tilde{\chi}_{s_{N_\tau}h_{N_\tau},h''}[x'',x_{N_\tau},y'',y_{N_\tau}]$ as function of the decoder gates $\tilde{g}_{h''}[x'',y'']$, the encoder's last cell state $C_{h_{N_\tau}}[x_{N_\tau},y_{N_\tau},N_\tau]$ and the parameters $\tilde{\mathscr{Q}}^g_{h''k}[x''-x_{N_\tau},y''-y_{N_\tau}]\rightarrow$ expense: $2N_h^2N_x^2N_y^2$

- accumulated expense: $N_k(2N_hN_x^2N_y^2)+(N_\tau-\tau)N_k(2N_hN_x^2N_y^2)(2N_hN_xN_y)+2N_h^2N_x^2N_y^2$
- then perform the summation over $s_{N_\tau}, h_{N_\tau}, x_{N_\tau}, y_{N_\tau}$ for each h'', k, x'', x, y'', y

$$\Xi_{h''k}[x'',x,y'',y,\tau]=\tilde{\chi}_{s_{N_\tau}h_{N_\tau},h''}[x'',x_{N_\tau},y'',y_{N_\tau}]\chi_{s_{N_\tau}h_{N_\tau},k}[x_{N_\tau},x,y_{N_\tau},y,N_\tau,\tau]$$

expense: $N_k(N_hN_x^2N_y^2)(2N_hN_xN_y)$

- next calculate $\Omega_{rh''}[x'',y'']$ as a function of the decoder's output $\tilde{H}_{h''}[x'',y'']$ and the parameters $\mathscr{W}^1_{a'h''}[x'',y'']$, $\mathscr{W}^2_{b'a'}$ and $\mathscr{W}^3_{rb'}$

expense: $N_hN_xN_y$

- finally perform the summation over h'', x'', y'' for each k, x, y

$$\omega_{rk}[x,y,\tau]=\Omega_{rh''}[x'',y'']\cdot\Xi_{h''k}[x'',x,y'',y,\tau]$$

expense: $N_k(N_hN_x^2N_y^2)$

accumulated expense:

$$\begin{aligned}N_\tau\Big[&N_k(2N_hN_x^2N_y^2)+(N_\tau-\tau)N_k(2N_hN_x^2N_y^2)(2N_hN_xN_y)\\&+2N_h^2N_x^2N_y^2+N_k(N_hN_x^2N_y^2)(2N_hN_xN_y)+N_hN_xN_y+N_k(N_hN_x^2N_y^2)\Big]\end{aligned} \tag{95}$$

2.2 Choice of the reference sequence $\{X_k[x,y,\tau]\}$

2.2.1 Average over all sequences in the data set

Suppose the number of all sequences with length N_τ in the data set is N_s

$$\bar{A}_r[t]=\frac{1}{N_s}\sum_{\{X_k[x,y,\tau]\}\in\{X_k^t[x,y,\tau]\}}\left(M_r(\{X_k[x,y,\tau]\})+\frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]}(X_k^t[x,y,\tau]-X_k[x,y,\tau])\right) \tag{96}$$

$$=\left(\frac{1}{N_s}\sum_{\{X_k[x,y,\tau]\}\in\{X_k^t[x,y,\tau]\}}M_r(\{X_k[x,y,\tau]\}))-\frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]}X_k[x,y,\tau]\right) \tag{97}$$

$$+\left(\frac{1}{N_s}\sum_{\{X_k[x,y,\tau]\}\in\{X_k^t[x,y,\tau]\}}\frac{\partial M_r(\{X_k[x,y,\tau]\})}{\partial X_k[x,y,\tau]}\right)X_k^t[x,y,\tau] \tag{98}$$

$$=\bar{a}_r+\bar{\omega}_{rk}[x,y,\tau]X_k^t[x,y,\tau] \tag{99}$$

2.2.2 Optimal sequence

Find a specific sequence $\{X_k^*[x,y,\tau]\}$ such that

$$A_r^*[t]=M_r(\{X_k^*[x,y,\tau]\})+\frac{\partial M_r(\{X_k^*[x,y,\tau]\})}{\partial X_k^*[x,y,\tau]}(X_k^t[x,y,\tau]-X_k^*[x,y,\tau]) \tag{100}$$

minimizes the integrated squared error

$$\sum_t(R_r[t]-A_r^*[t])^2 \tag{101}$$