Supplementary Material

1 MODEL ARCHITECTURES AND TRAINING

1.1 Multilayer Perceptron (MLP)

The MLP considered for the water level prediction task receives as input 25×25 pixels water level fields flattened to vectors $\vec{x} \in \mathbb{R}^{25 \cdot 25}$. Those vectors are passed through a hidden layer with 100 neurons and a rectified linear unit (ReLU) nonlinear activation function to an output layer with one neuron whose linear activation is the prediction of the network. The target $y \in \mathbb{R}$ is the water level at the target location normalized to have a mean of 0 and a standard deviation of 1 on the training set.

We trained the network for 150 epochs using the Adam optimizer with the learning rate and weight decay parameters both set to 1e-3, a batch size of 256 and the mean squared error as loss function. We implemented this using the deep learning library Pytorch (Paszke et al., 2019).

1.2 Convolutional Neural Networks (CNNs)

The input to the CNNs are geopotential fields at 500, 850 and 1000 hPa. For each pressure level, we normalized the geopotential fields to have a mean of 0 and a standard deviation of 1 on the training set.

CNN1 is a shallow CNN with two convolutional layers, each consisting of 24 kernels of size 3 and rectified linear units (ReLU) nonlinear activation functions. The convolutional layers are followed by a fully-connected layer connecting the second convolutional layer to two output neurons. CNN2 is a commonly employed, much deeper CNN architecture called resnet18 (He et al., 2016) for which the last fully-connected layer was adapted to have only two output neurons to fit our prediction task.

The networks predict a rain event for a given input if the value of the second output neuron is higher than the value of the first output neuron. To obtain a 2-dimensional vector with no rain and rain probabilities which sum to one, a softmax function is applied

$$P(\text{rain event})_i = \frac{\exp(out_i)}{\exp(out_1) + \exp(out_2)}, \tag{S1}$$

where out_i is the activation of the i-th output neuron. Note that for the saliency maps, we consider the derivative of the output of the second output neuron before the application of the softmax layer. Applying the saliency maps to the first output neuron yields very similar results. The target vector for an input $\vec{x}$ is a 2-dimensional vector indicating whether a rain event occurred ($\vec{y} = (0, 1)$) or not ($\vec{y} = (1, 0)$).

We trained both networks for 60 epochs using the Adam optimizer with a learning rate of 1e-3, a batch size of 1000, the CrossEntropyLoss-criterion and the ReduceLROnPlateau scheduler with patience parameter set to 6. We implemented this using the deep learning library Pytorch (Paszke et al., 2019).

Note that for CNN1 the number of neurons in the fully-connected layer has to be adapted when the size of the input region is changed to 80×80 pixels. This is not the case for the resnet18 model, as for this model, the output of the last convolutional layer is always pooled to a fixed size.

REFERENCES

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. doi:10.1109/CVPR.2016.90

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc.). 8026–8037