Multi-output neural processes

Zinzan Gurney

20 January 2022

This research project is supervised by Prof. Richard Turner and has three aims:

1. Build a conditional neural process (CNP) for handling multi-output data and test on synthetic data.
2. Understand which architectures are good for deploying CNPs on multi-output data on the input side and output side.
3. Establish benchmarks for evaluating CNPs on multi-output data on real-world environmental and healthcare datasets.

This report will explore the motivation behind the project, background on neural processes, current achievements and work, and future directions.

1. Motivation

Deep learning methods have enjoyed enormous success across a range of applications, however, they fall short in certain problems where only small datasets are available and good uncertainty estimation is required. In particular, one area not yet explored is making joint predictions on data with multiple outputs where a number of possible patterns of missingness are present. These kinds of datasets are often encountered in healthcare settings with irregularly sampled time-series and environmental science with off-the-grid spatial data.

For example, in a spatial-data environmental problem, at any given location, we may have access to a number of measurements such as rainfall, temperature or river streamflow. At each location, we may only have access to a subset of observations, for example if the sensor is disabled for a period of time or the measurement is irregularly sampled. This can be seen in fig. 1 where the we do not have access to the flooding sensor, however, could we exploit the rough relationship between the two variables, that high rainfall increases the probability of a river flooding, to make joint predictions over both outputs. In short, a flexible model is needed to handle robustly a number of possible patterns in which observations can be missing.

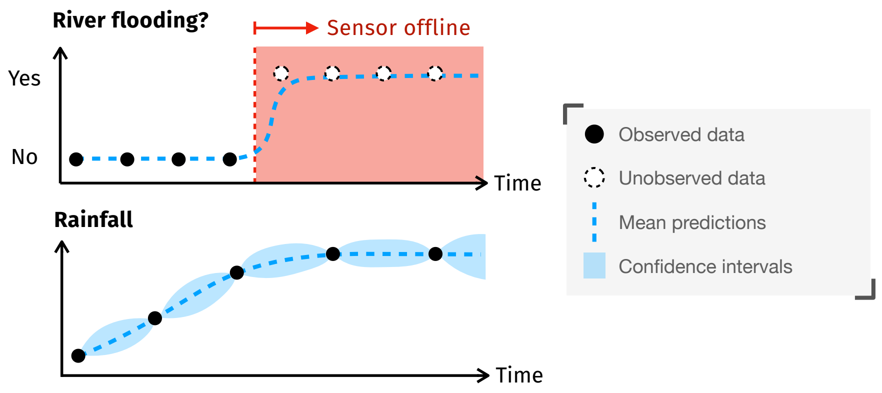


Figure 1: Toy example of a binary and continuous output joint prediction problem

Current approaches in making predictions on multi-output data with deep learning approaches often naively combine the losses for each output like $L_{\text{total}} = \sum_i w_i L_i$ and thus model performance is highly sensitive to the selection of these weights, as shown in Kendall et al. In the case of two outputs, if the weighting between the two is incorrect, we can end up with the model fitting only one output. The problem of tuning these weight hyper-parameters only becomes more challenging when more than two outputs are modelled, due to combinatorial possibilities in attempting to cross-validate, and we will not be able to perform this validation process on small datasets.

Neural processes, introduced by Garnelo et al. in 2018, are a rich and powerful set of models which produce strong predictions in the small-data regime with good uncertainty estimation. They are a good fit to address these issues highlighted above with deep-learning approaches as they learn automatically from a large training meta-dataset the appropriate way to weight and combine multiple modalities. However, they have not yet been extended to handle multi-output data, which is the central motivation for this project.

2. Background on neural processes

Neural processes tackle issues of working in a small-data regime with good uncertainty estimation by meta-learning a mapping from datasets to a predictive distribution. They draw inspiration from ideas of deep learning which is excellent at approximating complex functions as well as from probabilistic approaches which use prior knowledge to perform inference in small-data regimes. To contrast the standard supervised learning approach with neural processes: the former are trained from scratch on a single dataset to learn a function that can be used as a predictor on unseen data; the latter are trained on a set of related datasets and incorporate uncertainty through learning a distribution over predictions.

To formally explain neural processes, we can take the view of prediction maps from Foong et al. Defining the context set $\mathcal{D}_c := \{(x^{(c)}, y^{(c)})\}_{c=1}^{C}$ and target set $\mathcal{D}_t := \{(x^{(t)}, y^{(t)})\}_{t=1}^{T}$. A neural process is a prediction map π, a function that maps from a context set $\mathcal{D}_c$ and a set of target inputs $\{x^{(t)}\}_{t=1}^{T}$ to a predictive distribution over a set of target outputs $\{y^{(t)}\}_{t=1}^{T}$:

$$\pi(\{y^{(t)}\}_{t=1}^{T} | \{x^{(t)}\}_{t=1}^{T}, \mathcal{D}_c)) \tag{1}$$

In this project, convolutional conditional neural processes (ConvCNP) by Gordon et al. are used as a basic architecture. Conditional neural processes (CNP) are a sub-family of neural processes which employ a factorisation assumption for modelling the predictive distribution: $\pi(\{y^{(t)}\}_{t=1}^{T} | \{x^{(t)}\}_{t=1}^{T}, \mathcal{D}_c)) = \prod_{t=1}^{T} p_\theta(y^{(t)} | x^{(t)}; \mathcal{D}_c)$. ConvCNPs address the issue of failing to generalise outside the training regime in CNPs by introducing an translation-equivariant inductive bias. In other words, if the context set is shifted in input space by T_τ, then the resulting predictions are shifted by T_τ as well.

Although the factorisation assumption of CNPs implies that they are unable to model correlations across input space and thus generate incoherent samples, they are easier to train than (latent) neural processes with a maximum-likelihood procedure and with its ConvCNP variant perform well enough for the purpose of this project.

3. Current achievements and work

3.1 Extending the ConvCNP architecture to handle two outputs

There are different possible ways in which the ConvCNP architecture could be modified to handle multi-output data. We propose the following approach in fig. 2, starting first with two outputs, which we term the *DualConvCNP*.

Firstly, we perform the convolutional deep set encoding $\mathrm{Enc}_\theta(\cdot)$ of the two outputs separately to the representations R_{reg} and R_{reg}. The key modification is in the next step: Rather than passing each representation separately through the convolutional neural network (CNN), we pass the two context set representations as input channels and output predictive parameter representations R_μ, R_{σ^2}, R_ρ jointly. As a result of this architecture, the two predictive distributions on the outputs will be informed by the context sets given by both modalities.

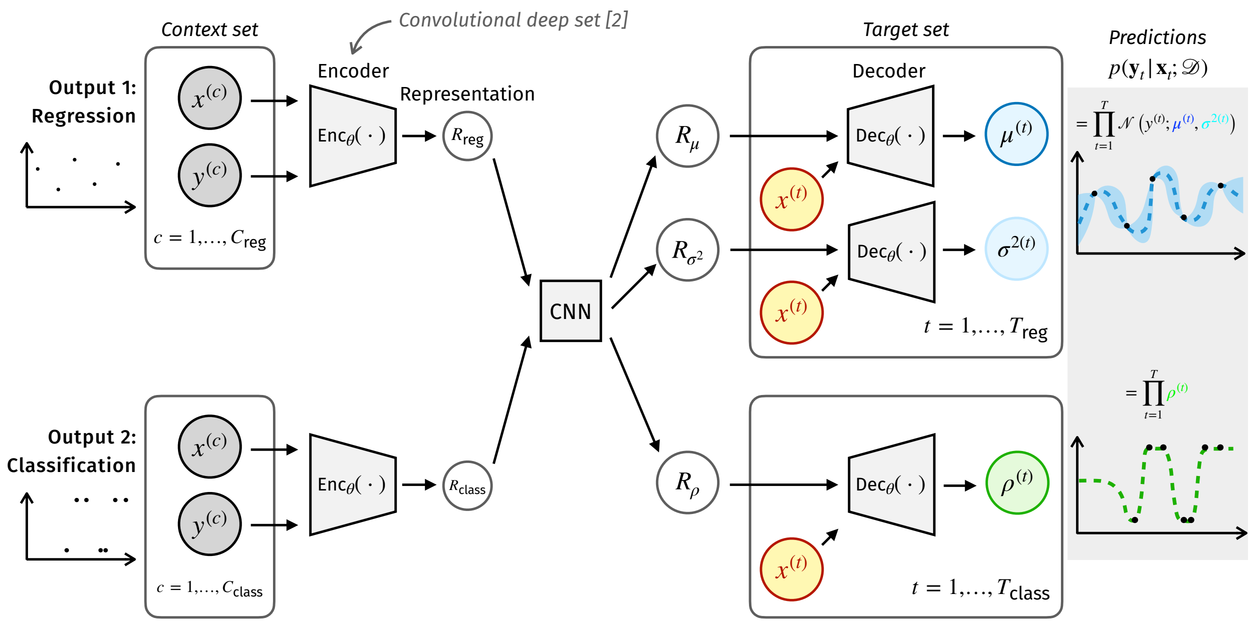

Context set
Convolutional deep set [2]
Target set
Predictions
$p(\mathbf{y}_t | \mathbf{x}_t; \mathscr{D})$
Output 1: Regression
Output 2: Classification
Encoder
Representation
Decoder
$x^{(c)}$
$y^{(c)}$
$\mathrm{Enc}_\theta(\cdot)$
R_{reg}
R_{class}
CNN
R_μ
R_{σ^2}
R_ρ
$x^{(t)}$
$\mathrm{Dec}_\theta(\cdot)$
$\mu^{(t)}$
$\sigma^{2(t)}$
$\rho^{(t)}$
$c = 1,\ldots,C_{\mathrm{reg}}$
$c = 1,\ldots,C_{\mathrm{class}}$
$t = 1,\ldots,T_{\mathrm{reg}}$
$t = 1,\ldots,T_{\mathrm{class}}$
$= \prod_{t=1}^{T} \mathcal{N}(y^{(t)}; \mu^{(t)}, \sigma^{2(t)})$
$= \prod_{t=1}^{T} \rho^{(t)}$

Figure 2: Computational architecture of the DualConvCNP

It is to be noted that although we have specified this new computational architecture in terms of a regression and classification output, this approach could be simply extended to more than two outputs as well as any permutation of regression or classification outputs.

3.2 Initial benchmark of performance of the DualConvCNP

We evaluate the performance of the DualConvCNP on two outputs against two single ConvCNPs, trained on each output separately.

To initially assess the DualConvCNP architecture, a synthetic meta-dataset $\mathcal{D} := \{\mathcal{D}_i\}_{i=1}^{N_{\text{tasks}}}$ was generated using the following procedure for each task D_i, composed of a continuous output $\mathbf{y}_i^{(1)} \in \mathbb{R}$ and binary output $\mathbf{y}_i^{(2)} \in \{0, 1\}$ over an fixed input range $x \in [-2, 2]$:

1. A function $f_i(x)$ is sampled from a Gaussian Process (GP) with a squared exponential (SE) kernel with length-scale $l = 0.25$.
2. To induce a relation between the two outputs, we threshold the function $f_i(x)$ to give a binary step function $g_i(x)$ as follows:

$$g_i(x) = \begin{cases} 1 & f_i(x) \geq 0 \\ 0 & f_i(x) < 0 \end{cases}$$

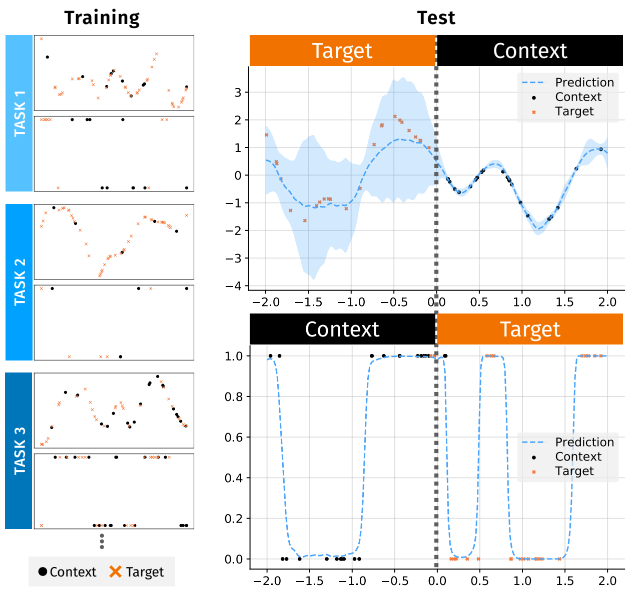
3. To generate the context and target sets for the two outputs, we then randomly subsample from the two functions respectively.

We test the performance of the DualConvCNP on an extreme task where the continuous output only has access to context points in the range $x \in [0, 2]$ and the binary output only has access to context points in the range $x \in [-2, 0]$. In tbl. 1, we see the greater performance over the independent single predictions from a standard ConvCNP.

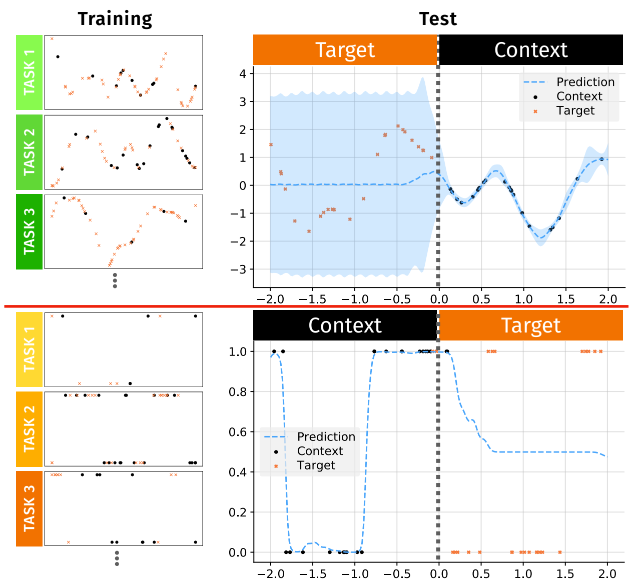
Visually, it is clear from fig. 3a, how predictions over the target range, particularly for the binary output, are informed by the DualConvCNP being able to learn the hyper-structure from the meta-dataset of tasks and then apply this thresholding relation to make joint predictions. This is in contrast to fig. 3b where predictions away from the context datapoints are completely uncertain.

Table 1: Losses over 64 evaluation tasks after 20 epochs

	Regression	Classification
Baseline	1,016	125
Single ConvCNP	526	54
Dual ConvCNP	502	31



(a) DualConvCNP predictions



(b) Independent single ConvCNP predictions

Figure 3: Training and testing the Dual ConvCNP against independent separate predictions on 1D synthetic data

4. Future directions

The future directions of the work centre around three tasks: working with time-shifts, deploying on the CAMELS dataset and developing the computational architecture. The timeline for the unfoldment of these tasks is summarised in fig. 4. The tasks with the same colour indicate dependencies for the following key tasks: building the basic DualConvCNP, exploring causal filter synthetic data, building a GNP architecture, testing on CAMELS dataset, testing on MIMIC-IV dataset.

4.1 Evaluating performance on a time-shift

Beyond the simple scenario of the thresholding relation explored above, the question that emerges is natural: what kinds of hyper-structure across outputs can the DualConvCNP learn? For example in the toy example in fig. 1, we might expected a time delay between a period of high rain and subsequent flooding. Thus, we modified the synthetic data to induce a time delay τ in one output, as shown in fig. 5a.

We note that initial results look promising. In sections 2 and 3 without any context points in the classification output, we can see that the boundary between the two sections, shows the DualConvCNP has learned the time shift of $\tau = 0.5$. Moreover, in section 1, the classification predictions become uncertain as the DualConvCNP has learned that the corresponding regression outputs yields no information. The next steps will be to produce quantitative measures of performance against the independent predictions as well as to investigate if the DualConvCNP can learn non-constant time-shifts in the training meta-dataset.

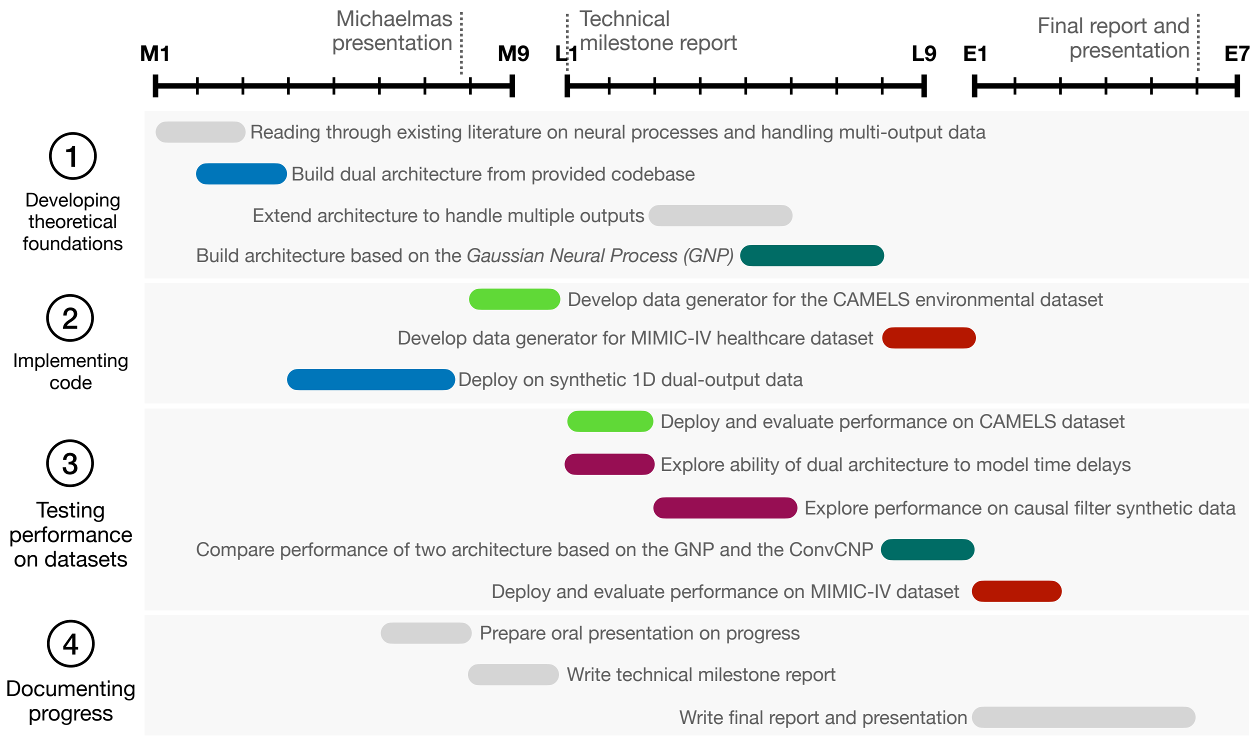
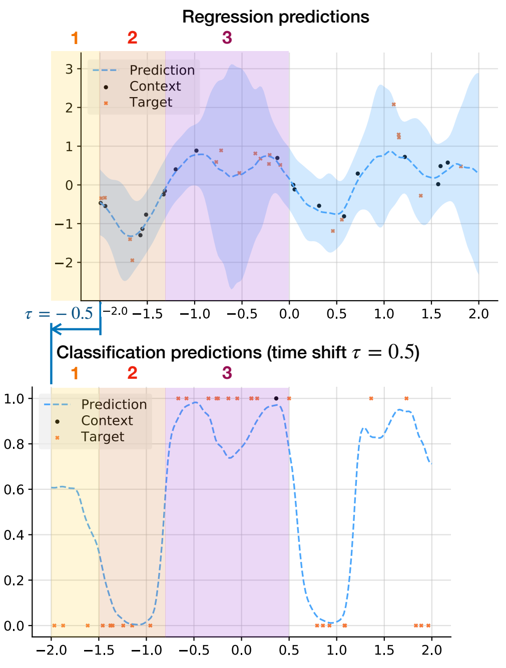
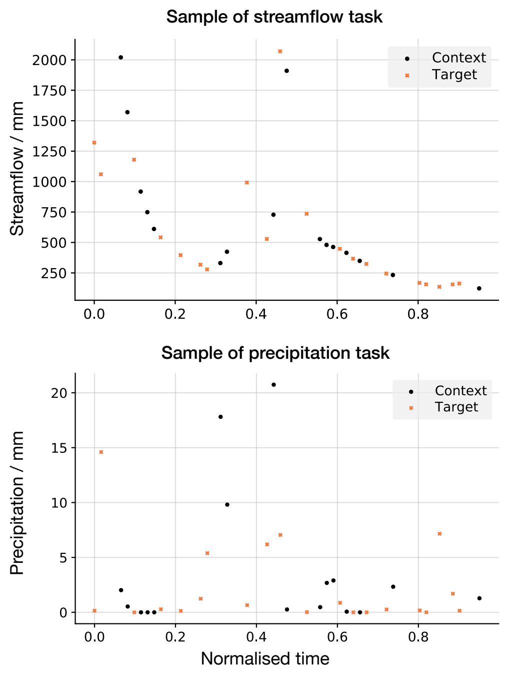


Figure 4: Gantt chart of major tasks and milestones for the whole project from October 2021 to June 2022



(a) Dual ConvCNP predictions on time-shifted $\tau = -0.5$ 1D synthetic data



(b) Sample of a two-output task generated from the CAMELS dataset

Figure 5: Data for future work

4.2 Deploying on the CAMELS environmental dataset

As a first evaluation of the performance of the DualConvCNP on real-world datasets, we turned to the CAMELS dataset (Newman et al.) which is composed of hydrometeorological time series gathered over a period of 30 years across 671 basin in the United States. Here, a flexible model able to produce joint predictions over multiple outputs would be essential as current approaches rely on hand-crafted physical models which are not robust in handling missing data and require careful fine-tuning to model dependencies, rather than learning them from the data. As a starting point, we selected two measurements that could potentially have a relation: the mean daily discharge or streamflow of the river basin, and the mean daily precipitation at that location. We might hypothesise that periods of high rainfall may cause a rise in the streamflow of the river; could the DualConvCNP learn this hyper-structure automatically without an expert specifying it?

Before evaluating this question, we first needed to generate a meta-dataset $\mathcal{D} := \{\mathcal{D}_i\}_{i=1}^{N_{\text{tasks}}}$ on which the DualConvCNP can be deployed. The following approach is taken to produce each task $\mathcal{D}_i$: we randomly select a river basin and a date from the 30 year period, we then subsample randomly from a period of 60 days to obtain the context and target sets over a normalised time input range $x \in [0, 1]$. An example of a task generated by this procedure is shown in fig. 5b. The next step will be to deploy the DualConvCNP on this meta-dataset and compare predictions to hydrological models that already exist.

4.3 Developing the computational architecture

In addition, we may also explore how to build in appropriate inductive biases into the architecture. In the CAMELS dataset, we might suggest that the streamflow output could look like a convolution of the precipitation output with a time-delayed, exponentially decaying causal filter. For example, we could imagine that a single high precipitation datapoint would result in a delayed peak in the streamflow data, followed by exponential decay. Thus, we will simulate this by extending the synthetic data generator to produce samples by convolving a GP or sparse distribution with time-varying variance together with a causal filter with a time delay and exponential decay. This exploration will be crucial in comparing the performance of the DualConvCNP to traditional physical models as we can learn the underlying dynamics between the hydrological outputs automatically.

A key task in developing the computational architecture is modelling dependencies on the output side. Since CNPs generate independent *mean-field* predictions, they are unable to produce coherent samples. In the CAMELS dataset for example, the probability assigned by a CNP architecture to the event of high rainfall over multiple consecutive days would be unreasonably low as each input location is modelled as independent. A potential solution is to generate samples in an autoregressive manner, by conditioning each sample on previous data to produce dependent predictions over time. To explore this, we will develop the *Gaussian Neural Process* architecture proposed in Markou et al. to handle multiple outputs.

Finally, we will also explore how to extend the dual architecture to multiple outputs more generally as well as deploying the multi-output ConvCNP on the MIMIC-IV healthcare dataset to evaluate the performance of joint predictions over multiple physiological variables.

5. References

Foong, Andrew Y. K., et al. “Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes.” *Advances in Neural Information Processing Systems*, 2020, p. 12.

Garnelo, Marta, et al. *Conditional Neural Processes.* 2018, p. 10.

Gordon, Jonathan, et al. “Convolutional Conditional Neural Processes.” *International Conference on Learning Representations*, 2020, p. 32, https://openreview.net/forum?id=Skey4eBYPS.

Kendall, Alex, et al. *Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.* 24 Apr. 2018, http://arxiv.org/abs/1705.07115.

Markou, Stratis, et al. *Efficient Gaussian Neural Processes for Regression.* 18 Oct. 2021, http://arxiv.org/abs/2108.09676.

Newman, Andrew, et al. *A Large-Sample Watershed-Scale Hydrometeorological Dataset for the Contiguous USA*. UCAR/NCAR, 2014, pp. approximately 2.5 GB, https://doi.org/10.5065/D6MW2F4D.