Technical Documentation

WAVE2WEB Hackathon

RESERVOIR WATCH

Walter Samuel

Saumya Srivastava

Sagar Garg

Priya Shejule

Chamarthi ***Sarat*** Chandra

Sumana Borbora

Manash Bhuyan

Indirapriyadarshini Jagirupu

Jayatu Bhuyan

Bernard Ernest

Table of contents

1. Problem statement

The forecast of near real-time water availability in reservoirs helps in effective water resources management, flood control, hydropower generation, irrigation, and drought mitigation. Artificial intelligence and deep learning techniques can be used for forecasting reservoir water availability based on historical water availability, hydrological modelling, hydro-meteorological data, and water demand. This predictive modelling exercise yields meaningful results when it is complemented with human understanding of the physical processes in the river basin. This is achieved by combining predictive and hydrological modelling. Appropriate selection criteria can be used to arrive at the best performing model combination. The visualization and dissemination of the results of this deep tech exercise are done on an interactive dashboard. Other important hydro-meteorological datasets can also be visualised on the dashboard to help stakeholders in making informed decisions.

2. Study area

Cauvery river originates in the Western Ghats of India at Coorg district in Karnataka, India. It traverses 800 km and drains an area of 82,000 square kilometres before it flows into the Bay of Bengal. The Upper Cauvery has four major reservoirs, namely - Hemavathy, Harangi, Kabini and Krisharajasagara. Water from these reservoirs accounts for 80% of the water supply to Bengaluru city. The reservoirs mainly depend on the south-west monsoon rainfall causing a high discharge during the months of June - September while October - May is considered a lean period. The topography (digital elevation model), the land use, land cover type, soil type and sub-basins are shown in the figure below.

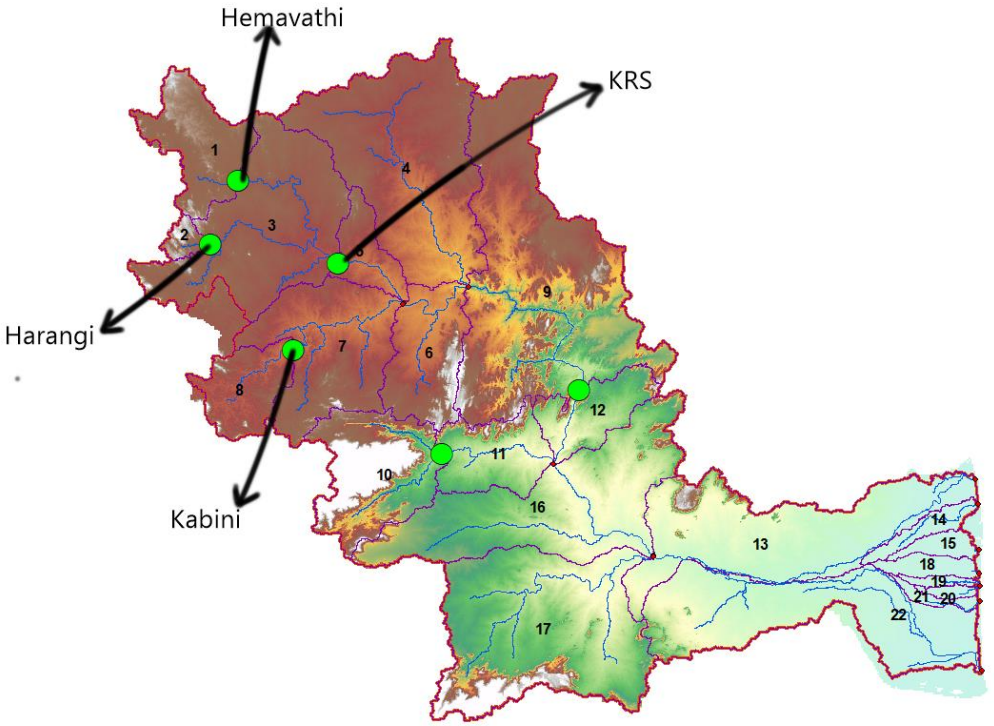


Figure 1. Cauvery River basin

3. Methodology

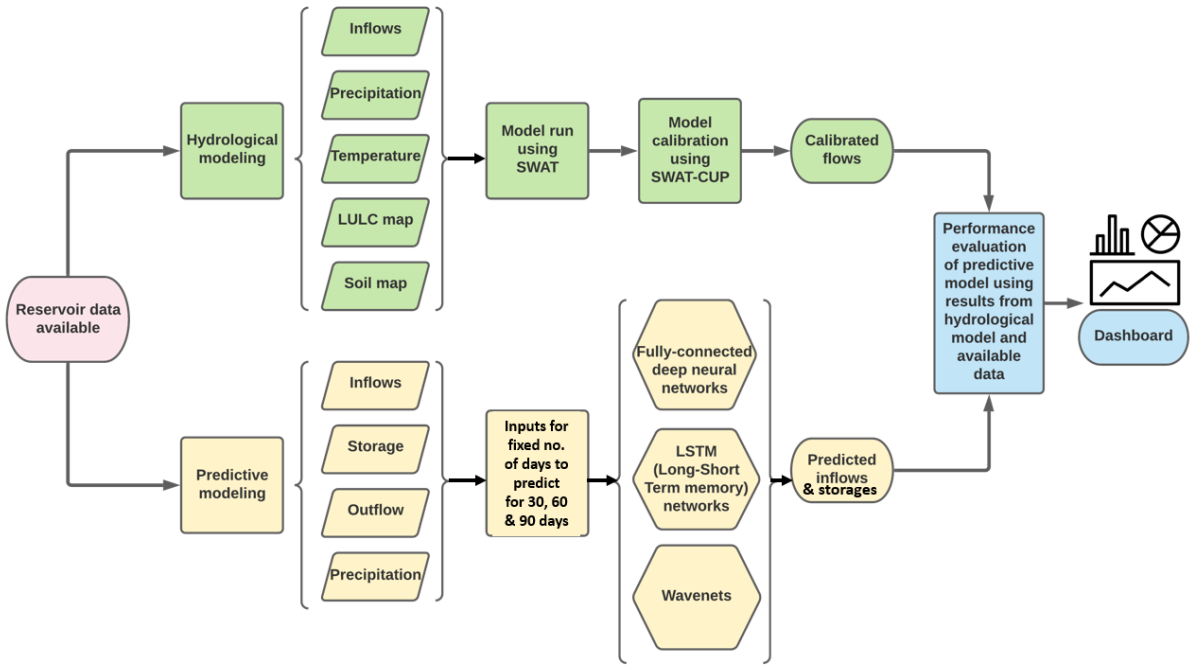


Figure 3. Flowchart for our methodology

4. Hydrological modelling

4.1 Need for hydrological modelling

It is important to impose physical constraints into ML models and couple them to process-based models because of the inability of ML models alone to produce physically consistent results. Reservoir inflows are dependent on the rainfall occurring upstream of the reservoir, soil moisture, evapotranspiration, groundwater availability and some other catchment characteristics. The transformation of rainfall to runoff is a highly complex process which can only be understood by adopting a hydrological model to simulate the behaviour of the river basin.

4.2 Details of the model used

We use a semi-distributed hydrological model, Soil and Water Assessment Tool (SWAT), to simulate the hydrological processes in the Cauvery River basin. In this model, the entire river basin is divided into smaller sub basins and each subbasin consists of Hydrological Response Units (HRUs). It can simulate various rainfall-runoff processes as shown in the diagram below.

Details about the model setup and the input and output files can be found at https://drive.google.com/drive/folders/1Z8Cff8hQNjiEutE76atbjrNk2J5_oQoz?usp=sharing.

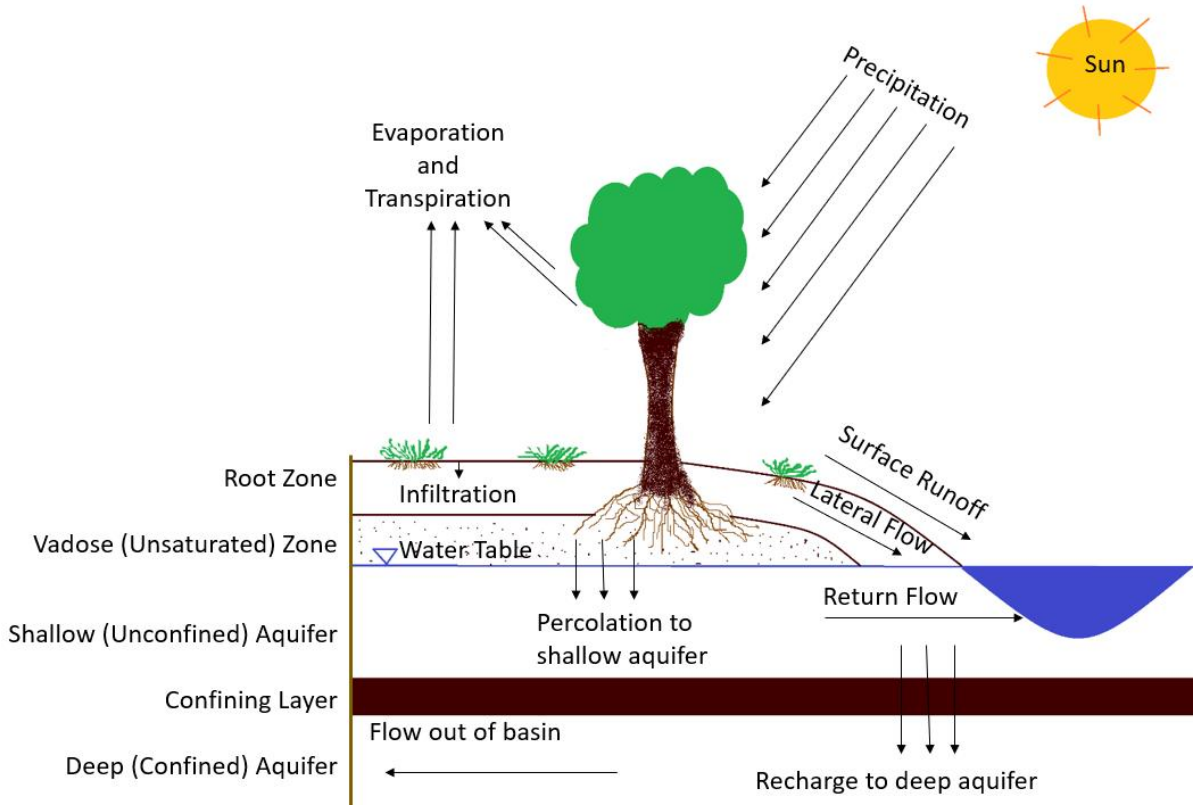


Figure 3. Hydrological processes modelled in SWAT

4.3 Data used in hydrological modelling

Type	Data	Source
Static	DEM	Shuttle Radar Topography Mission (90 m resolution)
Static	LULC	ESA (European Space Agency) GlobCover 2009 (Global Land Cover Map) (300 m resolution)
Static	SOIL	FAO-UNESCO Soil Map of the World (5 arc-minute/ 0.083 degrees resolution)
Dynamic	Precipitation	IMD (0.25 degree) (subbasin averaged)
Dynamic	Temperature	IMD (1 degree by 1 degree)
Dynamic	Relative Humidity	Simulated in SWAT (CFSR data 0.3 degrees)
Dynamic	Solar Radiation & Wind	Simulated in SWAT (CFSR data 0.3 degrees)
Dynamic	Streamflow	Observed daily streamflow data at 4 reservoirs

Table 1. Data used in the study

4.4 Calibration and validation of the hydrological model

We have used SWAT- Calibration and Uncertainty Program (SWAT-CUP) for the calibration and validation of the model. Multi-site calibration is done at 4 locations for inflow to the reservoirs in the Upper Cauvery Basin – Harangi, Hemavathi, Kabini and KRS. The following parameter ranges were selected based on literature review.

Tables below show the calibrated parameter ranges for Hemavathi, Hanrangi, Kabini and KRS reservoir basins.

Parameter Name	Minimum Value	Maximum Value	Definition
CN2*	-0.29	0.04	SCS runoff curve number for moisture condition II
ALPHA_BF	0.37	1.12	Base flow alpha factor
GW_DELAY	0	240	Ground water delay
GWQMN	0.78	2.36	Threshold depth of water in the shallow aquifer for return flow to occur (mm)

Table 2. Calibrated parameters for calibration of SWAT model for Hemavathi and Harangi reservoirs

Parameter Name	Minimum Value	Maximum Value	Definition
CN2*	-0.15	0.01	SCS runoff curve number for moisture condition II
ALPHA_BF	0.32	0.96	Base flow alpha factor
GW_DELAY	0	247	Ground water delay
GWQMN	0	1.30	Threshold depth of water in the shallow aquifer for return flow to occur (mm)
SOL_K	228	609	Soil conductivity (mm/hour)
SOL_AWC	0	0.66	Available water capacity
ESCO	0.08	0.69	Soil evaporation compensation factor
EPCO	0.09	0.70	Plant evaporation compensation factor

Table 3. Calibrated parameters for calibration of SWAT model for Kabini reservoir

Parameter Name	Minimum Value	Maximum Value	Definition
CN2	17	66	SCS runoff curve number for moisture condition II
ALPHA_BF	0	0.05	Base flow alpha factor
GW_DELAY	48.65	213.35	Ground water delay
GWQMN	4140	4835	Threshold depth of water in the shallow aquifer for return flow to occur (mm)
GW_REVAP	0.12	0.19	Ground water revap coefficient
REVAPMN	16.24	56.16	Threshold depth of water in the shallow aquifer for revap to occur (mm)
ESCO	0.52	0.88	Soil evaporation compensation factor
SOL_AWC	0.20	0.55	Available water capacity

Table 4. Calibrated parameters for calibration of SWAT model for KRS reservoir

4.5 Hydrological model output

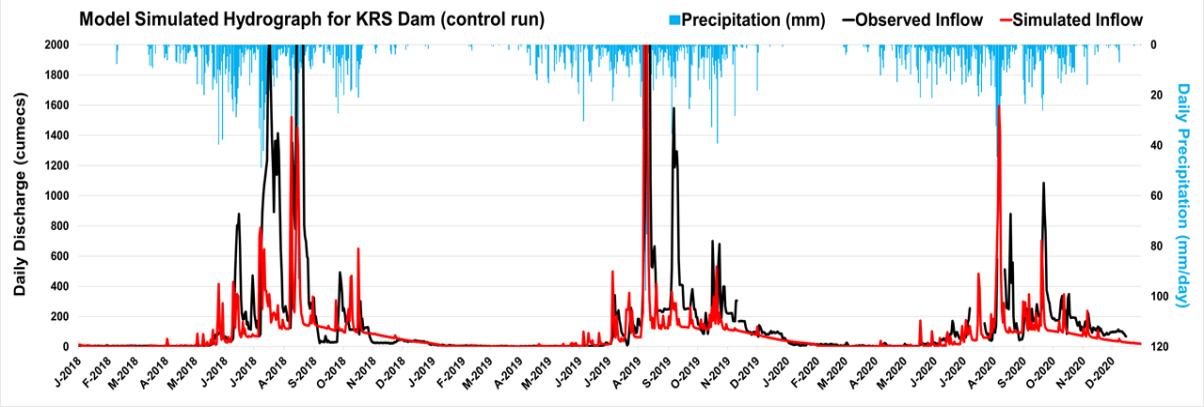


Figure 4. Output from the calibrated and validated SWAT model. Daily precipitation (in blue) and inflow (observed in black and simulated in red) to the KRS reservoir.

The calibrated and validated model parameter sets are then fed back into the SWAT model to get realistic simulations of rainfall-runoff processes in the basin. The outputs from this final SWAT model are used in training the machine learning models and are also shown on the dashboard.

5. Link between hydrological and machine learning model

The integration of a hydrological model with machine learning models considering long-term dependencies is recommended in discharge forecasting in many scientific works. When our machine learning model used inputs from our hydrological model, the results were better as compared to the results by using machine learning alone.

6. Machine learning model

We present our work to build a machine learning model that *learns* from data without domain-knowledge and forecast basin's **storage capacity, and inflow to the basin.**

6.1 Code

Our code is available as a simple Jupyter notebook, with helpful comments and plots included, and a partial list of our many trials and things that didn't work. We implement everything in simple modules, making it very user-friendly. All you need to do is get the input files, change the first block: *Configuration* and run your desired model. https://github.com/AQRITY/RESERVOIRWATCH/blob/main/final_machinelearning_code.ipynb

6.2 Data

Observed data included Reservoir specific hydrological variables, Meteorological Variables. **Simulated variables for all** these observations were also included through our **SWAT model – the semi-distributed Hydrological Model we built** as part of the competition.

6.3 Data Exploration

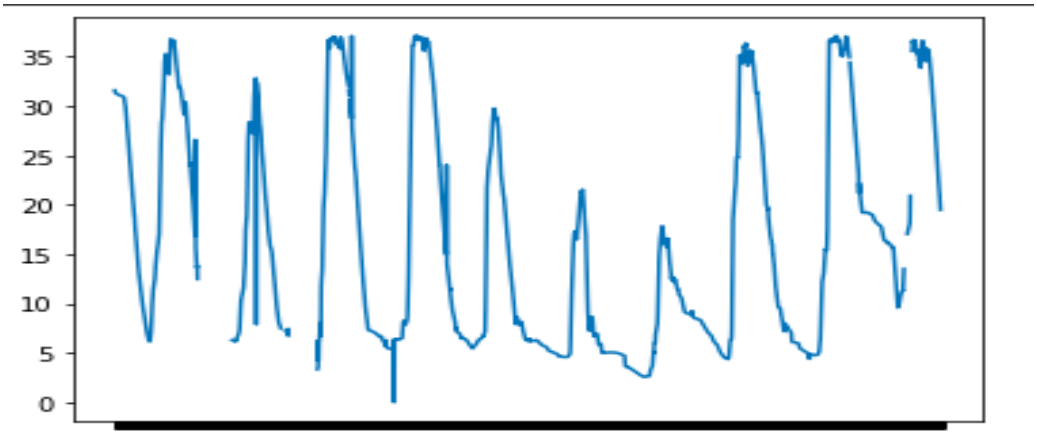


Figure 5. Plot of storage vs date

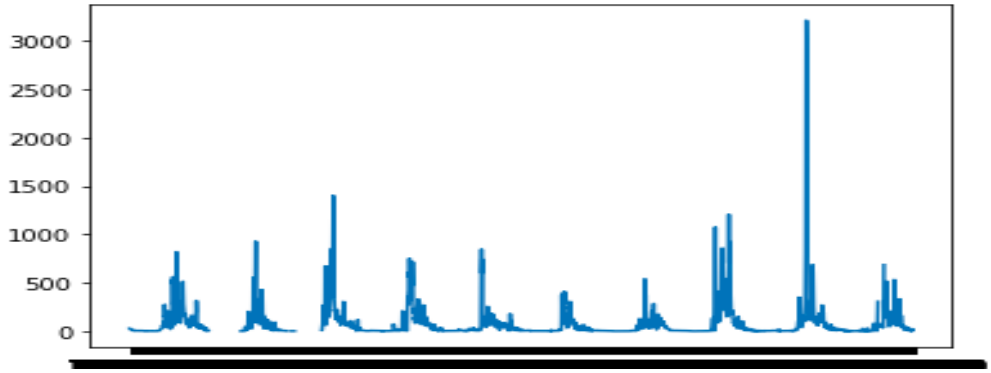


Figure 6. Plot of inflow vs date

Around 10% of data is missing, mostly in continuous chunks around 2013. Storage is easier to predict with a relatively stable frequency, but Inflow data clearly shows seasonal spikes (~1 per year), predicting which correctly is of extreme importance.
We attempt a few strategies to handle these issues.

- Error-Trend-Seasonality Decomposition: It decomposes the time-series into separate graphs, easier to understand the Error, Trend and Seasonality in the data. As a simple baseline method, one can simply remove the 1. seasonality, 2. train a model on the simpler data and 3.add the seasonality back.
- Normalized Nash Sutcliffe loss (*explained later)*
- Monte-Carlo Uncertainty Estimate (*explained later)*
- Missing data: We tried data imputation methods that keep in mind the strong seasonality. We refer https://www.kaggle.com/juejuewang/handle-missing-values-in-time-series-for-beginners for quick testing and https://www.nature.com/articles/s41598-018-24271-9 for a deeper understanding. However, again the simpler approach worked better- simply removing the data.

6.4 Feature Addition

We add sinusoidal functions – 'Day Sin', 'Day Cos', 'Month Sin', 'Month Cos', converting date to a better, smoother feature (since data has cyclic tendencies yearly, it is better to show that Dec-2015 is as similar to Jan-2016 as March-2016 is to April-2016. Source: https://stats.stackexchange.com/questions/311494/best-practice-for-encoding-datetime-in-machine-learning

6.5 Feature Selection

Neural networks work better when there are fewer features, especially with small data. We follow the general philosophy of Occam's Razor. There is no need to over-compliment a model, with many features and variables and algorithmic supplements when a simpler model would work. To reduce number of input features, we:

1. Find out the **Correlation Matrix** of all variables (except the dependent one). If two features share high correlation, then it is better to drop one of them.
2. Then we implement a more concrete method using **Random Forests**. RFs are a tried-and-tested machine learning algorithm that follows a random tree data structure at its core. Each input is passed through a node, and then depending on a threshold it splits into the left or right category. This provides a natural way to learn **Feature Importance.** We train a small random forest, find the most important features and remove all other features.
   We finally now have clean data.

6.6 Data processing

We normalize the data – subtracting mean and dividing by standard deviation of training set, as is common for neural networks.

6.7 Data Split

Train-validation-test set is split 70%-20%-10%. For the test-set, the prediction labels are completely from the new 10% data, although the initial few input labels are for the preceding dates from the validation set. This is for fair comparison among models, so that each gets tested on the same time-range.

6.8 Cross-Validation

We tried *BlockingTimeSeriesSplit()* method, which is a better approach than sklearn's *TimeSeriesSplit(),* since it prevents data leakage among the models. Ultimately, we did not implement it. Read more: https://goldinlocks.github.io/Time-Series-Cross-Validation/

6.9 Windowing

Our setup is based on https://www.tensorflow.org/tutorials/structured_data/time_series.
The *WindowGenerator()* class takes in input as

- INPUT_WIDTH: past input days
- LABEL_WIDTH: days to predict (30, 60 or 90)

- SHIFT: predict time-steps ahead

and creates an object window. The data is then further split into inputs and labels using a *split()* function. We also add a *plot()* function. Finally we convert it to *tf.Data.Datasets* to be inputted into the neural network.
The input is of the shape [*Batch Size, Time-Steps, Features].*
For multi-time-step prediction, it looks like the following figure.

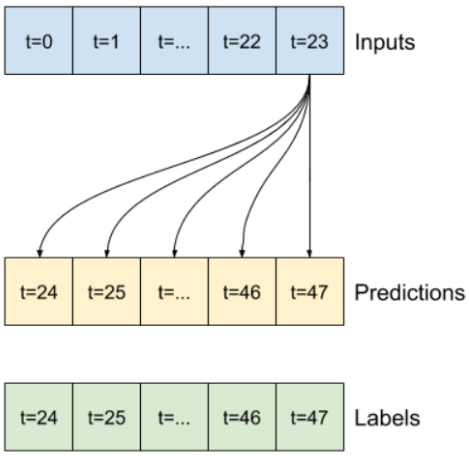


Figure 7. Multi time-step prediction model
(Figure taken from https://www.tensorflow.org/tutorials/structured_data/time_series)

6.10 Models

We try 3 different architectures: Fully Connected Dense Neural Networks, Recurrent Neural Networks and WaveNets. Finally, after much experimentation, **we decided on Recurrent Neural Networks – specifically the Long-Short Term Memory architecture**.

6.10.1 Fully Connected Dense Neural Networks

The first architecture is a fully connected deep neural network- one of the most basic deep neural network architectures in which there are hidden layers and in each hidden layer each neuron or unit is connected to each neuron in the next layer making the model having many parameters. In our implementation, we have 3 hidden layers. We also make small improvements using a convolutional layer making the model flexible to all input shapes.

6.10.2 Long-Short Term Memory Networks (LSTMs)

The advantage is that they can process much longer sequences than simple RNNs using these special Gates which try and understand what information to keep and how long is that information needed. We could have also tried other variants of LSTMs such as GRUs and many other architectures out there, but research has shown that all perform approximately similar. https://arxiv.org/abs/1503.04069.
Instead, we focused our energies on building a basic LSTM and then improving the model by making it more complex and robust, adding dropout layers to prevent overfitting, including convolution so that we can help detect even longer patterns than hundreds of time-steps, including layer normalisation to stabilize gradients, and many other small changes for which we encourage you to look at our scripts.

6.10.3 WaveNets

Our third architecture WaveNets is an interesting addition, especially for time series since we do not know if anyone has applied this method successfully ever. The advantage of WaveNets which originally came from the audio domain is that we can process much longer time sequences than it is possible for a LSTM and do it much faster. They are made only by convolutional layers so there is no recurrent unit in these. They are just layers with increasing dilation (or field of view) so the first layer looks at two timestamps, the second level looks at the previous 4 time steps, the third layer looks at 8 time-steps and so on. We implement a small model of the original paper by only using 4+4 convolutional layers, but expanding it will improve performance, as we plan to do in future.

6.11 Final Model Architecture: Long-Short Term Memory

The simple architecture is:
Input > Convolutional Layer > (1 or 3) LSTM layer > Dense Layer > Output
The convolution layer helps make the model flexible for any input shape. Also as seen in WaveNets, it allows for a larger field of view. The LSTM layer has **dropout and recurrent dropout** applied, and in some implementations –has a **Layer-Norm** layer. The Time-Distributed Dense layer (fully connected layer) allows us to reshape the output into desired length. We learn of this implementation through the highly recommended book: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/
For smaller input widths, we prefer a simpler 1-LSTM layer network and for others, a 3-LSTM layer network. We have tried dozens of combinations including very deep networks, but they seem to overfit on our small data.

6.12 Monte-Carlo Ensemble Model

We convert our single deterministic predictions into probability distribution functions to help find an uncertainty estimate. Machine Learning models often suffer from overconfident predictions, uncertainty in observations and in the natural process being modelled. We apply Monte-Carlo Dropout technique (https://arxiv.org/abs/1506.02142) which learns a predictive posterior distribution for an observation instead of a single deterministic point. The method states that dropout applied during training to NNs is in fact a Bayesian approximation to Gaussian process - a probabilistic model that builds a distribution of functions. In practice, we simply keep dropout switched ‘on’ during test-time. Each prediction is different, coming from a Monte-Carlo sample function from the space of possible functions. We take 50 samples, build an ensemble, and find mean and variance.

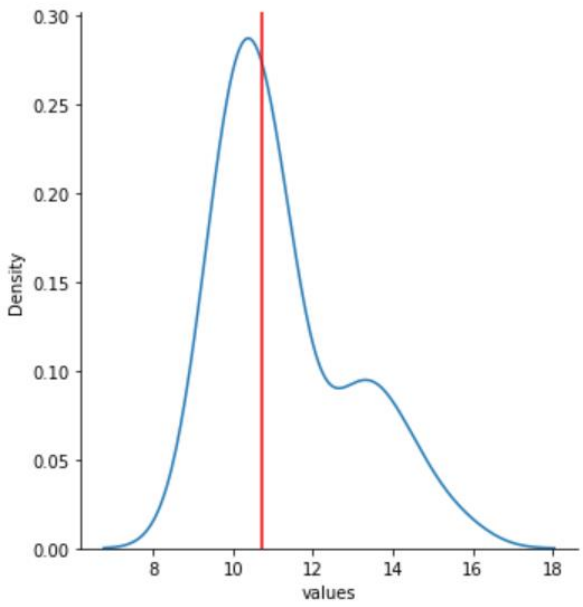


Figure 8. Sample output from Monte-Carlo Dropout

Here is a sample output from Monte-Carlo Dropout. Red line shows the deterministic model prediction. Blue curve shows the density plot when the model was run 50 times.

6.13 Model components and hyper-parameters tuning

We tried 100s of combinations to reach our final conclusions. It is an iterative process that takes enormous time and effort, and we encourage you to look at some of the partial results in the notebook to get an understanding of what didn't work and why.

As explained, we build a simple LSTM. The optimizer is ADAM. To reduce overfitting and increase model generalization, we include Early Stopping Patience, L1 and L2 regularization, LayerNorm and dropouts. The codes are run on Google Colab GPU, with final model runs of 500 epochs, with a patience of 40 epochs.

6.13.1 Loss Functions

We attempt three different loss functions: Mean Absolute Error, Nash Sutcliffe Loss and Root Mean Squared Error. The Nash Sutcliffe Loss is a popular evaluation in Hydrology, defined as one minus the ratio of the error variance of the modelled time-series divided by the variance of the observed time-series.

$$NSE = 1 - \frac{\sum_{t=1}^{T} (Q_m^t - Q_o^t)^2}{\sum_{t=1}^{T} \left(Q_o^t - \overline{Q}_o\right)^2}$$

It ranges from {-infinity to 1}, but since that might cause problems for a neural network, we convert it to the range {0 to 1} by using a normalized NSE. (*Our own implementation*, read on https://en.wikipedia.org/wiki/Nash%E2%80%93Sutcliffe_model_efficiency_coefficient)

$$NNSE = \frac{1}{2 - NSE}$$

6.13.2 Learning Rate

We attempt the new research of 1-Cycle Learning Rate: https://arxiv.org/abs/1506.01186. We find the minimum and maximum of the cycle through a Learning Rate *Finder()* function: https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html. We also attempt power scheduling learning rate, but Cyclic LR is better.

6.13.3 Batch Size

We attempted with size 16, 32, 128 and 256; finally, 256 was chosen. The larger batch size provides less variance among the epochs and the loss falls more smoothly.

6.13.4 Hidden Layers

We attempt with 1-LSTM hidden and 3-LSTM hidden layers. For smaller input/output width, a smaller network is preferred.

6.13.5 Dropout Rate

Range tested from 0.1-0.5. The dropout rate has a direct effect also on the final ensemble model; hence we must be careful on balancing variance of the final density plot and the sharpness of the predictions.

6.13.6 Input Width

How many days of past do you need to predict the future? For variables such as inflow, with ~1 spike per year, it becomes necessary to have a longer input – we took 365 days. However, we must note that performance of an LSTM suffers because of this, since in general they don't have enough memory to remember such long sequences. So, we face a trade-off again. For storage, with more monthly and quarterly patterns, an input width of 90 or 120 days was enough. **Results for storage are with 90-day input for 30-day output. 120-day input for 60 day or 90-day output for Harangi, Hemavathi and Kabini. Results for storage for KRS are with 60-day input for 30-day output, 120-day input for 60 day or 90-day output.**

6.14 Results

Sample results for KRS dam are shown here.

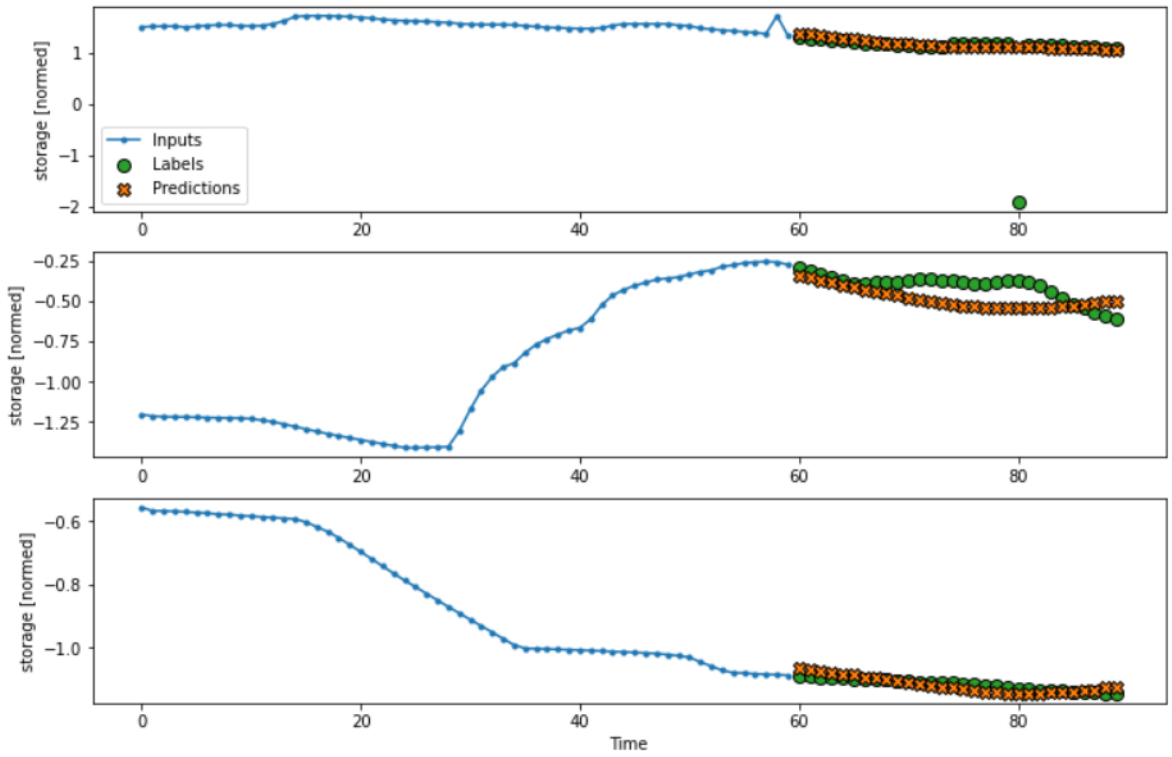


Figure 9. Sample results for KRS dam. Input: Past 60 days. Output: Next 30 days.

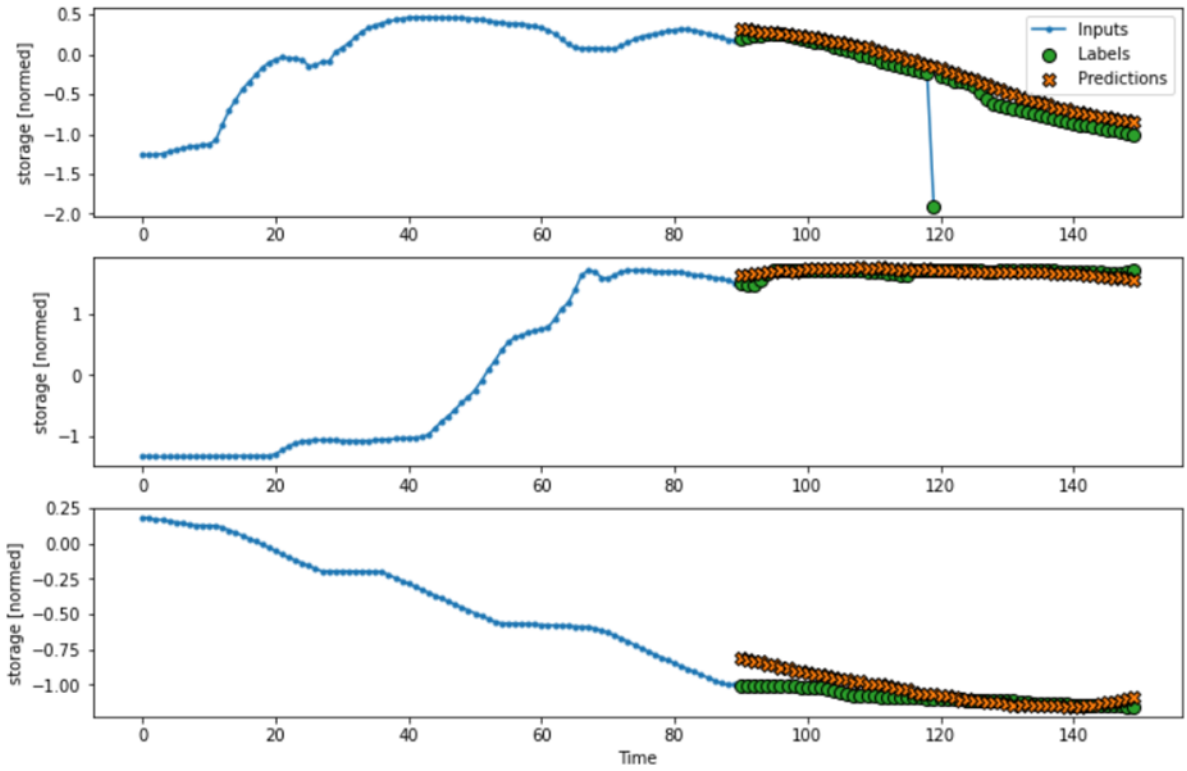


Figure 10. Sample results for KRS dam. Input: Past 120 days. Output: Next 60 days.

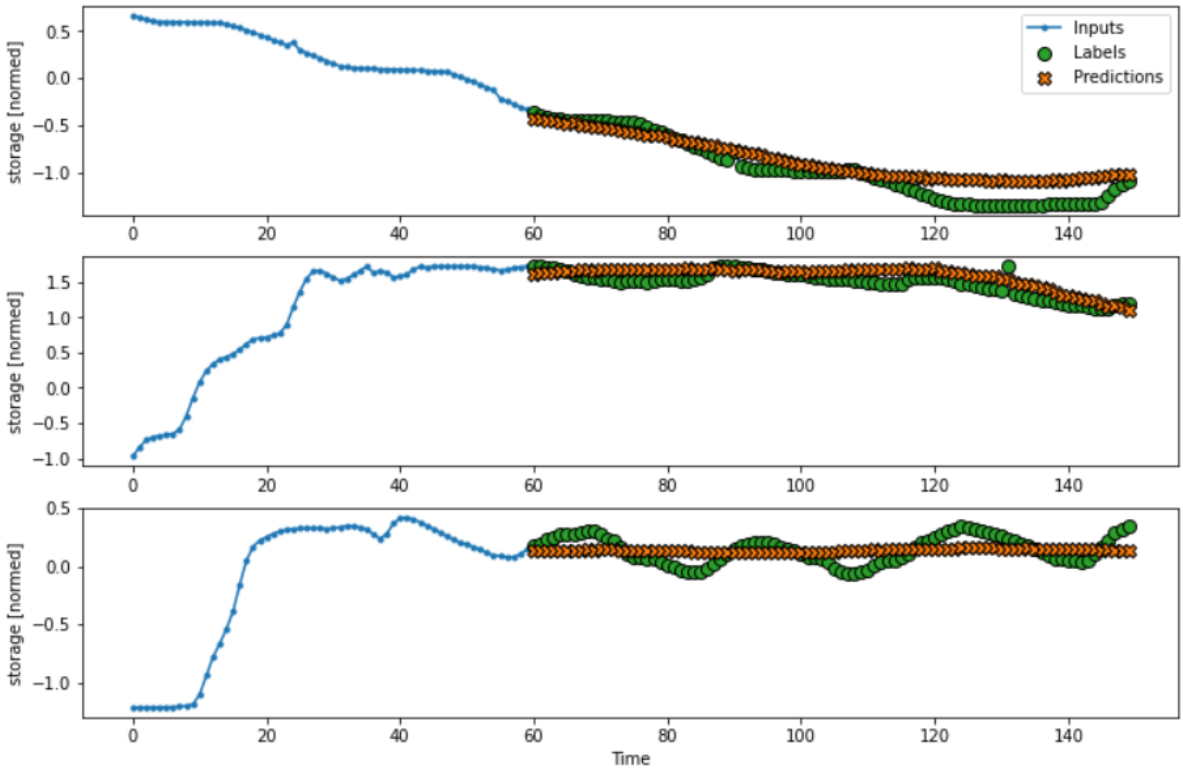


Figure 11. Sample results for KRS dam. Input: Past 120 days. Output: Next 90 days.

6.15 Explainability

Explainability has recently been a big question for machine learning models. We understand the importance of this question to water Management Committees in the real world. We tried (but couldn't finish due to time) at implementing interpretability of our models through SHAP (or SHAPley Additive explanations), which assign an importance value to each input variable showing us how each input affects the output. An easier though less accurate feature importance is available through the random forest method.

7. Interactive Dashboard

There are a couple of factors that we wanted to focus on when designing the interactive dashboard. But we set a brief to begin with i.e. 'To Research & Design a more informative but user-friendly interface for an overview/detailed view dashboard.'

The next step was to identify the keywords of our design process which are that the interface must be:

Simplified, Elegant, Visualized, Cognitive and Bespoke as we like to believe that our solution has to be unique and approached without any prior bias, as they data and its weightage should lay out the design for us.

There were also a couple constraints that we had, it had to be time bound, customizable, programmable, run on any browser for easy accessibility and should offer an overview and detailed view.

Now to set the base parameters we let our focus area be on the output data at first. So, the number of dams and the sample output data was taken into consideration.

	observed data from cwc				simulations from SWAT hydrological model						
date	storage	reservoir level	observed inflow	observed outflow	simulated inflow	simulated outflow	precipitation	simulated evapotranspiration	simulated storage	precipitation	Evapotranspiration
(mm/dd/yyyy)	TMC	ft	cumecs	cumecs	cumecs	cumecs	cubic meter	cubic meter	TMC	mm/day	mm/day
1/1/2011	31.6	2916.07	27.78192	56.66832	68.62	122.5	0	181500	23.35005764	0	
1/2/2011	31.5	2915.95	24.04368	55.224	67.2	64.64	0	134300	23.35005764	0	
1/3/2011	31.39	2915.82	22.14624	55.224	65.69	63.78	0	164400	23.35005764	0	
1/4/2011	31.34	2915.77	19.6824	30.444	64.23	37.23	0	172500	23.4277499	0	
1/5/2011	31.34	2915.77	18.71952	15.576	62.82	45.96	0	160100	23.47365897	0	
1/6/2011	31.33	2915.75	15.97248	18.408	61.41	62.59	0	157100	23.46306457	0	
1/7/2011	31.3	2915.72	16.02912	21.24	59.94	60.5	0	210700	23.45600164	0	
1/8/2011	31.27	2915.69	16.02912	21.24	58.68	57.02	0	143400	23.45600164	0	
1/9/2011	31.27	2915.68	17.36016	16.992	57.3	50.43	0	161800	23.4701275	0	
1/10/2011	31.26	2915.67	17.33184	16.992	55.98	54.07	0	164600	23.4701275	0	

Figure 12. Sample result labels

Now that the parameters have been set, we proceeded to follow an industry standard approach towards product design.

Approach Model -

Phase 1 - Research

• Map past, present & future trends

• Market Analysis

• Competitor study

• Technological advancements

- Scope

- Direction & Integration

• User Study (3 constructive & 2 critical minimum)

- Emotions

- Lifestyle

- Touchpoints

Output - Persona

Phase 2 - Exploration

• Problem areas from persona and user study mapping based on priority

• Solution analysis for the problems

• Identification of core problem areas

• Mock concept discussions for problem solutions based on personal feedback and execution

• Conclusion to chosen priorities & direction

Output - Flowchart structure

Phase 3 - Conceptualization

• Screens requirements

• Assets Requirements

• Back-end discussions for scope of features, animations & transitions and their executions

Output - Wireframes & Theme/Visual concept

Phase 4 - Prototyping

• Assets creation

• Screens finalizations

• Screen flow & logic for

- Major & Minor Interactions

- Animations

- Transitions

- Linking

Output - Prototype

Phase 5 - Data collection

• Cross check & Validate projections

• Identify Errors and Flow obstacles

• Reiterate affected areas

Output - Changes if required

Phase 6 - Deployment

• Convert to html, CSS and java script

• Create a DY Graph chart to observe and render the data outputs

• Project Map coordinates and animation trajectories

• Create Database of Text dialogues like Trivia etc

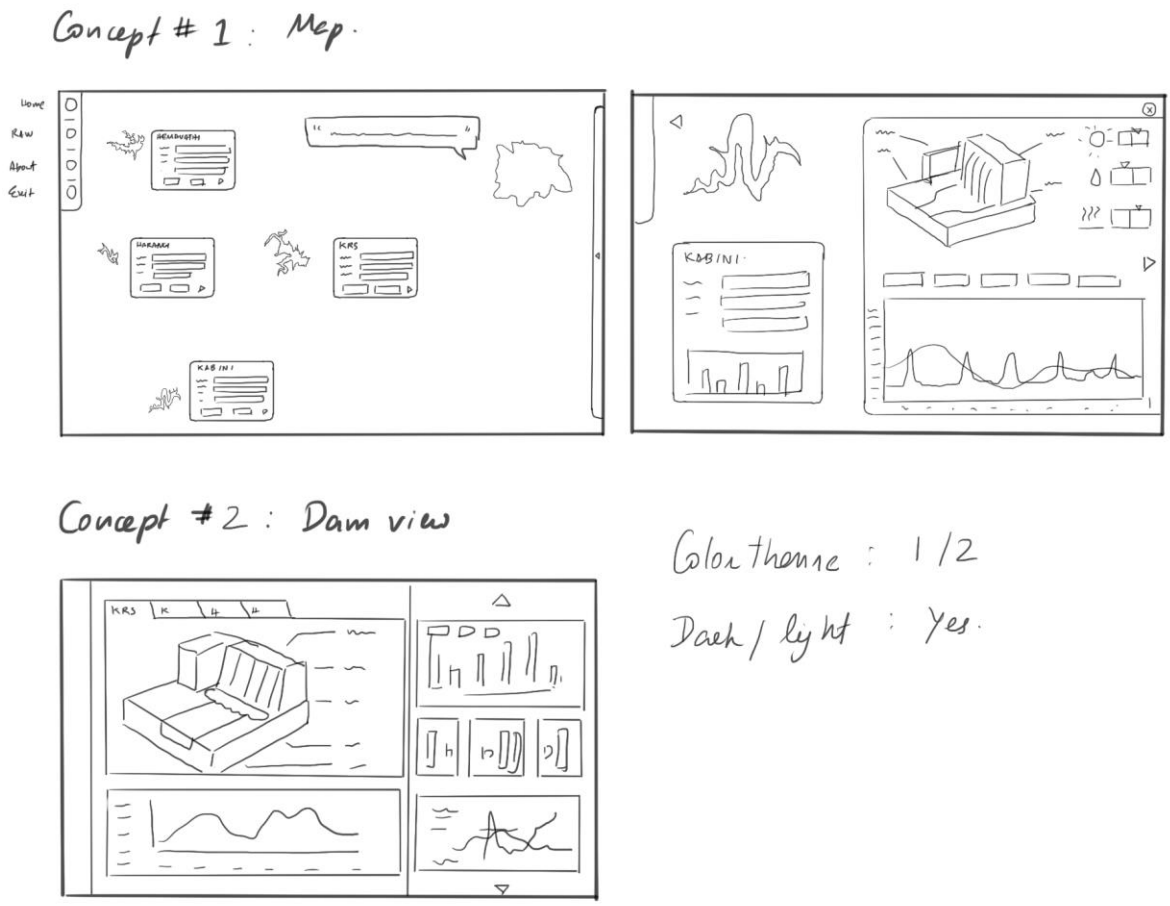
• Integrate all Image Assets

• Integrate Maps and Graphs

• Host Instance at AWS

Output - Live Webapp

The next crucial understanding was the conference of the team to discuss the basic wireframe sketch for the layout.



Key features

- Map Overview of all 4 Reservoirs
- Primary Output Overview floats on all Reservoirs in Map View
- Detailed Reservoir View
- Reservoir Isometric Visual
- Visual Thresholds
- Super User View
- Customizable Super User View
- Light & Dark mode
- Theme
- NLP Summarizer for Overview
- Interactive Output for Reservoir Visuals

After which we made mock-ups of different colour and layout themes. Out of the 6 that were created under 4 categories - Aqua light, Aqua Dark, Simple Light and Colour Dark. We went with Aqua light as it seems to be the friendliest to new users.

Our final layout had to convey these output data into a cognitive and easy to grasp interface that should be so intuitive that a first-time user should be able to use the dashboard without any prior training.

For this we had to first come up with a narrative. Ideally, anyone should know where these reservoirs are and how do they affect life in human settlements. A map overview is the most ancient and proven method to improve a sense of awareness about a space, hence we decided to allow the users an overview of the different reservoirs. This allowed for two things, one being that the nearest capital settlements were visible and a toggle-based view of the river network can allow the users to visually understand how the basin is interlinked with the settlements. The second aspect is that on one page we can show the major attributes of all the reservoirs within the screen area for the present date.

Also in the overview page, we wanted to convey two other things. One being a dialogue that grasps the predicted data and come up with formulative insights that give us a peek into what to expect or what some major changes in data means for the areas around the basin. Another feature was to have a floating section that will have some trivia and facts about reservoirs and how they function and make an impact on human lives.

Upon clicking any of the reservoirs, we now enter a detailed view where a zoomed in map view along with general data about the reservoir is displayed along with the real-time data over a basic visual render of the reservoir. This is where we have the output graph that is based on a variable timeline and hover cursor activated date/output identifier. We can navigate the graph based on output by selecting whether we like to peruse the observed data for different parameters or the simulated hydrological model. Presently the Inflow and Storage options allow us to display our prediction model with the start date and number of days to predict. This begins once that has been selected and we click on the Forecast Button. Now the Output Graph is generated along with the standard deviation and the performance metrics are displayed. We can navigate to the other dams from this view or simple go back to the map overview and select the desired reservoir and observe the different data or predict using our model.