Measuring water stage level using machine learning and neural networks

Rodrigo Morales A01632834, Jessica Nicole Copado Leal A01637876, Carlos Estrada Ceballos A01638214, Andrés Olvera Rodríguez A01638129

Advanced artificial intelligence for data science.

When site visits are biased toward base flow or fair weather circumstances, time-lapse images of streams and rivers can offer additional qualitative information about hydrologic conditions at stream gauges. Quantitative data from still images from ground cameras are also abundant and can improve monitoring of current flow. Considering a time lapse photograph. It can be useful to fill data gaps when sensors malfunction and/or when funding for monitoring projects is interrupted. Machine learning models that employ scalar image features that could be programmatically computed to fill data gaps in flow meter records were built and evaluated. Automated analysis of time-lapse images from a single camera at a single location. A ground-based fixed camera was used to capture time-lapse photos as part of a watershed documentary imaging project.

rivers | hydrologic | photograph | models

Introduction

Being able to measure, predict and understand how a body of water behaves is a very important task in both hydrology and engineering. This way we can better manage water consumption in industry and cities, we can predict whether there will be a flood or even a drought, we can build better bridges taking into account the change of water throughout the year, and much more. With this objective in mind, work began on a project to create a model capable of predicting both the height and flow of different rivers.

The information with which we started working with comes directly from sensors placed in the water, and data captured by cameras placed in strategic locations so that they could capture the change in the water and the surroundings. The database obtained by these means is a combination of numerical measurements and images, both of which were used to create a deeper model. All the data used was taken from a study conducted by a research group from the University of Nebraska dedicated to solve hydrology issues.

Problem description. The problematic here is to find a way to predict the state of water (stage and discharge) at any time in the future using ML, sensor data and photographs of water bodies. By creating multiple ML models with various results, we can decide which is more effective to forecast how water behaves in other locations so that this model can be replicated to be able to solve distinct hydrological questions such as water distribution, flood and erosion management, etc.

Research question. Numerous crucial hydrological concerns, including those regarding floods and droughts, agricultural sustainability, global climate change, and water distribution, must be addressed. These questions depend on water level (stage) and streamflow (discharge). When sensors malfunction or are unable to operate due to a lack of funds, methods have been devised to fill in the gaps in sensor data. Some of these methods include cameras that cost little money and offer a variety of data that is not possible with regularly used sensors.

The research question here would be: Is a huge data set of photos sufficiently detailed to replace significant (or minor) gaps in data on the discharge and stage without manually indicating the data?

Work development

In order to begin the stage of analysis and characterization of the data, we employed the ELT (Extract, Load, and Transform) method after downloading the data. To build the most fitting model, we experimented with different variables, and tasks were carried out to clean, visualize, analyze, and transform data. The steps taken were to: create models, separate data by seasons, and to use images to create different models.

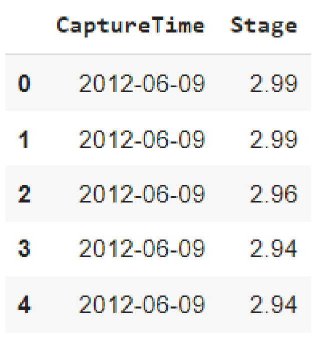
- Understand the research paper by reading the information explored by the research group from the University of Nebraska mentioned in their study.
- Analyze the dataset which included features that were combined with tage and discharge data from the United States Geological Survey (USGS) from more than 40,000 daylight pictures acquired at hourly intervals between 2012 and 2019.
- Data preprocessing.
- Modelation of ML and time series models.
- Create a Convolutional Neural Network (CNN) cropping some parts of the images like the edges or focusing on the white water. Also splitting data by years and seasons.
- Results and interpretations.

Dataset. Two sources provided information regarding the North Platte River State Line Weir site. From 2012 to 2019, the USGS retrieved the real-time stream stage sensor measurements and discharge computations for every fifteen minutes. The Platte Basin Timelapse gave daylight photographs for the same time period on an hourly basis. The total number of daytime photos in the dataset was around 40,000 recordings with 59 colums.

Methodology. In order to understand the problem in greater depth, it was decided to analyze the data to show its behavior over time and how the different seasons of the year affect our dependent variable. Since the data was measured throughout the year for several years, and therefore has a temporal order, we decided besides the regression model, add a a time series analysis.

A time series analysis is specifically good in these cases as it would allow us to visualize the information in a different way, being able to draw new conclusions and even observe characteristics of our dataset that are quite important for the development of the project.

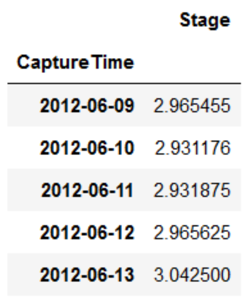
For the time series analysis the only columns that were vital for this, were CaptureTime and Stage 1. CaptureTime tells us in what date the record was made, an important variable for the analysis since it marks the data in time. Stage is our dependent variable that shows the height of the water in meters. In addition to Stage we have the Discharge variable, the other dependent variable, this one we will not be used since in the analysis of the data, determined that the correlation between both is very strong.



	CaptureTime	Stage
0	2012-06-09	2.99
1	2012-06-09	2.99
2	2012-06-09	2.96
3	2012-06-09	2.94
4	2012-06-09	2.94

Fig. 1. Multiple records of the variable Stage in a single day.

The data preprocessing for both the regression and time series analysis was necessary but relatively simple. First the data type of the CaptureTime variable was changed from object to datetime. Next, was to check if there were nan values in the dataset. Then, the creation of the new dataframe with only the CaptureTime and the Stage was created. It is important to mention that the sensors take several measurements in a single day, as can be seen in figure 1, so we have several records of the variable Stage and in order to perform the analysis the relationship between the variables has to be 1:1, that is, for each day there can only be one record of Stage. One way to achieve this is to group all the records per day, although it is great way to build the time series, it's not trustable since we are eliminating important information and there is no real way to determine which of the records in a day is more important than the others. Taking this into account, we decided to obtain an average of all the records in a single day 2, although this also deprives us of relevant data, it is a way to take into account all the data and their weight, without losing them. In addition, columns for month and year were added for further analysis of the data.



	Stage
CaptureTime	
2012-06-09	2.965455
2012-06-10	2.931176
2012-06-11	2.931875
2012-06-12	2.965625
2012-06-13	3.042500

Fig. 2. Dataframe with the dates grouped.

With our new dataset ready, we started with the analysis. The first thing we did was to make a graph where we could see in a more visual way the change of the river height along the years. Looking at the figure 3 we can see that the data has some seasonality, this is due to the change of seasons and temperatures, which directly affects the water body. For example, in the middle between years we can see large peaks, this represent a higher water level, which is due to high temperatures and possible summer rains. On the other hand, we can observe that, at the change of year, during winter, the height of the water decreases and remains constant for a while, due to the low temperatures and possibly the water freezing at this time.

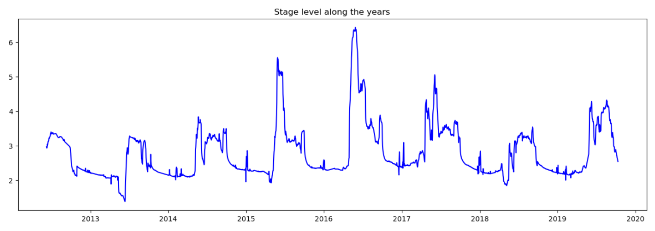


Fig. 3. Seasonality of the data graphically represented.

Season	Mean
Spring	2.6861
Summer	2.6342
Autumn	3.5145
Winter	2.2852

Table 1. Stage mean values per season.

Having observed the seasonal behavior, we decided to assign each month a season of the year, February to May would be spring, June to August summer, September to November autumn and December to January winter. This was done in order to be able to divide our data by seasons and see the characteristics of each one. We determined that the most constant season of the year in terms of water level is winter, although the level is approximately 2.2 meters, in summer we also found some constancy with levels of 3.5 even though they are less. These data seem to coincide with the average levels calculated by season shown in the table 1.

After the data analysis, we went on to perform a KPSS test to determine if the information followed a stationary pattern, which means that the data may change, but its change is constant, it is not affected. We performed this test using the kpss function within the statsmodels library, which gave us results greater than the critical value, so we can deduce that it develops in a stationary manner.

With all this information, the next step was to check what type of prediction model was this; MA, ARMA, ARIMA, etc. It would be necessary to test and plot the autocovariance, autocorrelation and partial autocorrelation. These graphs and metrics allow us to determine how the data of a variable is related within itself. By observing the graphs in figure 4, we were able to determine that the model that best fit our data would be an ARIMA model. This model has both the auto-regression of the AR model and the moving average of the MA model, which makes it deeper and capable of working with more complex data.

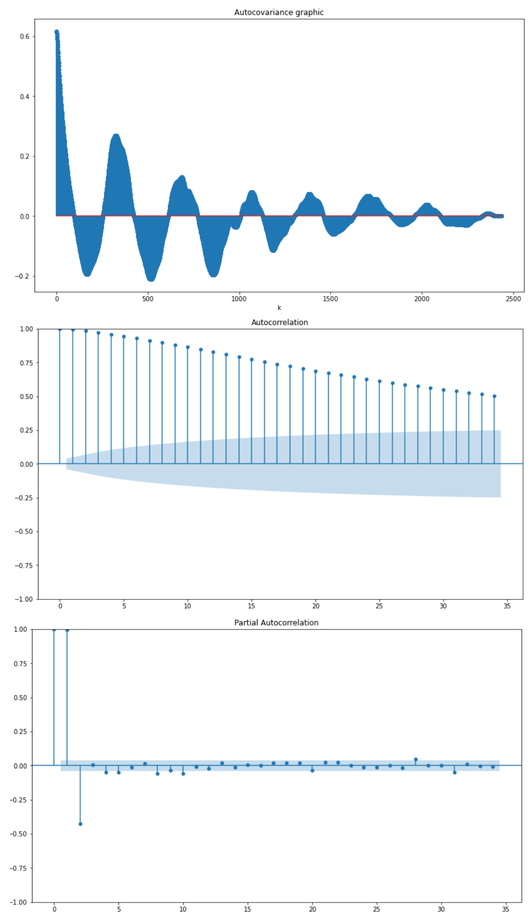


Fig. 4.
1. Autocovariance plot
2. Autocorrelation plot
3. Partial autocorrelation plot

Results

A. Regression model. We decided to make two approaches for the data with the same procedure. The first approach consisted in splitting the data using the train test split function: 70% of the data for training and 30% for testing, this data split was made randomly. For the second approach, the dataframe was splitted into two new frames, the training dataframe that contained the records from year 2012 to 2017 and the testing dataframe from year 2018 to 2019. Once the data was splitted, the next step was to remove the residuals with high influence using three procedures:

- Studentized residuals
- High leverage
- Cook's distance 5

For each residual elimination, there was an OLS (ordinary least squares) model built to compare them with each other, also there was one model built deleting all three procedures at the same time. For the first approach we have table 2 and for the second approach we have table 3.

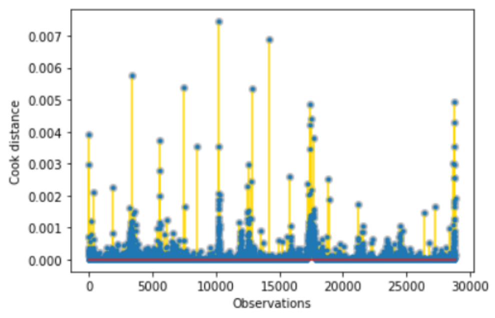


Fig. 5. High cook's distance residuals in second approach.

Procedure	$r^2 score$
Studentized residuals	0.788
High Leverage	0.699
Cook's distance	0.663
All procedures	0.802

Table 2. First approach OLS $r^2 tablescores$

Procedure	$r^2 score$
Studentized residuals	0.801
High Leverage	0.711
Cook's distance	0.797
All procedures	0.741

Table 3. Second approach OLS $r^2 tablescores$

For the first approach, the model with the best $r^2 score$ was the with all the procedures deleted at the same time with a value of 0.802, as for the second approach, the best model was only deleting the studentized residuals with a value of 0.801. Once this was notices, the final step was to build different ML regression models to see which one will throw the best score. For both approaches, were implemented Decision Tree Regressor, Random Forest Regressor, Support Vecotr Regressor, MLP Regressor, and of course the OLS. As seen on tables 4 and 5, we have the $r^2 score$, the mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE) for each ML model.

ML Regressor	$r^2 score$	MSE	MAE	RMSE
OLS	0.802	0.084	0.217	0.290
Decision Tree	0.826	0.074	0.159	0.272
Random Forest	0.912	0.037	0.109	0.193
SVR	0.305	0.295	0.397	0.543
MLP	-1091.109	464.041	19.270	21.542

Table 4. First approach ML accuracy and errors.

ML Regressor	$r^2 score$	MSE	MAE	RMSE
OLS	0.801	0.206	0.341	0.454
Decision Tree	0.344	0.253	0.341	0.503
Random Forest	0.552	0.173	0.290	0.416
SVR	0.059	0.363	0.460	0.602
MLP	-1637.818	632.177	16.035	25.143

Table 5. Second approach ML accuracy and errors.

As seen on the tables, the ML model with the highest $r^2 score$ was the Random Forest Regressor with a value of 0.912 and low error values for the first approach. On the other hand, for the second approach, the best model was the OLS with a value of 0.802 also with low value errors. So, the approach that best predicts the water stage level was the first one.

B. Time series model.

In order to check the data interpolation, an ARIMA (Autoregressive integrated moving average) model was built. It is an statistical model that uses data variations and regressions with the final purpose to find patterns for it to make a future prediction. The model has certain parameters that change the behavior of the model itself:

- **p:** Number of lag observations included in the model.
- **q:** Size of the moving average window
- **d:** Number of times that the raw observations are differenced.

It is important to choose the right value for each hyperparameter so thay can throw good predictions, so the auto-arima function was implemented, resulting in p = 3, q = 1 and d = 2.

Once the values of the hyperparameters are set, the ARIMA model can be build. Looking at figure 6 you can see that both lines are practically on top of each other. The model threw coefficients close to 0.05 which is the confidence level the model sets automatically, this means the model is trustworthy.

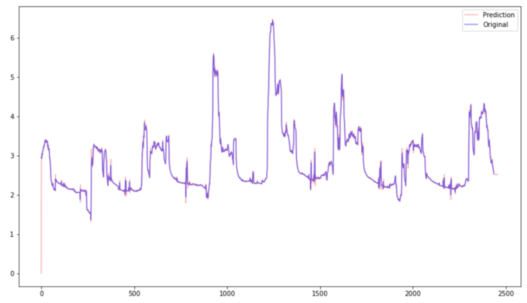


Fig. 6. Overlapping graphs of prediction model and actual data.

For a final step with the time series analysis, a function was applied to try predicting the year 2020. Prophet, is a library for forecasting time series based on an additive model where non-linear trends are fitted yearly, weekly, and daily seasonality. In order to predict the year 2020, inside the function you can set the amount of periods you want to predict in this case 365. Figure 7, might be a little hard to comprehend, but will explain it. The black dots are the original recordings from the dataframe, the blue line is the model's prediction and the blue shaded area is the confidence level the prediction has. The image shows the 2020 prediction, but Prophet can predict the time series weekly, monthly, or even all together. You might think, it is a really great approach to predict the stage level unfortunately, due to the fact there are missing values, the model had a score of 0.61. These was really disappoitning, but it is a great to predict the stage level over the years. If the dataframe had all the dates, maybe the value will rise.

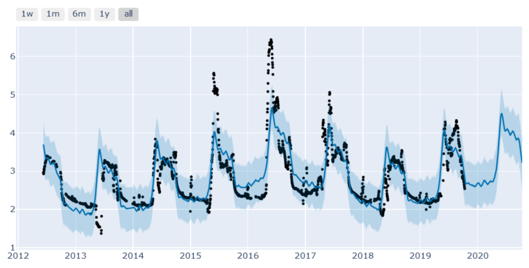


Fig. 7. Prophet model accuracy plotted.

Convolutional Neural Networks. These are composed of neurons with teachable biases and weights. Each neuron receives numerous inputs, weights them, passes the total through an activation function, and then produces an output. They were created specifically to work with images and videos. They use photographs as inputs, extract and learn the image's attributes, then categorize the images using the learnt features. These neurons build the foundation for automatic recognition by learning how to translate input impulses into equivalent output signals.

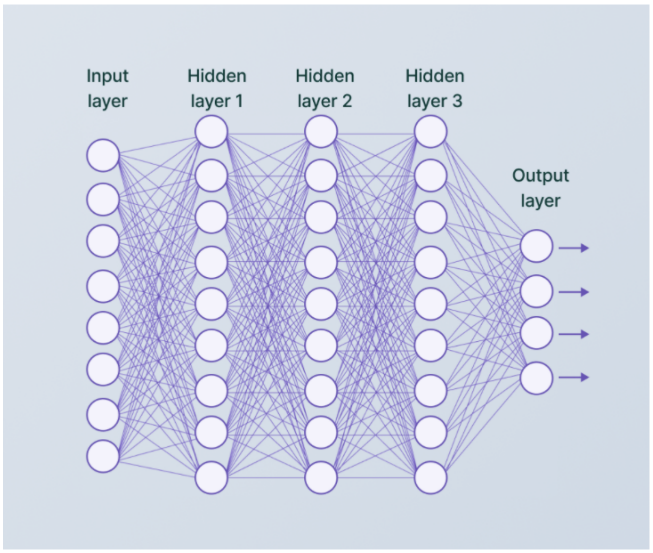


Fig. 8. Convolutional Neural Networks

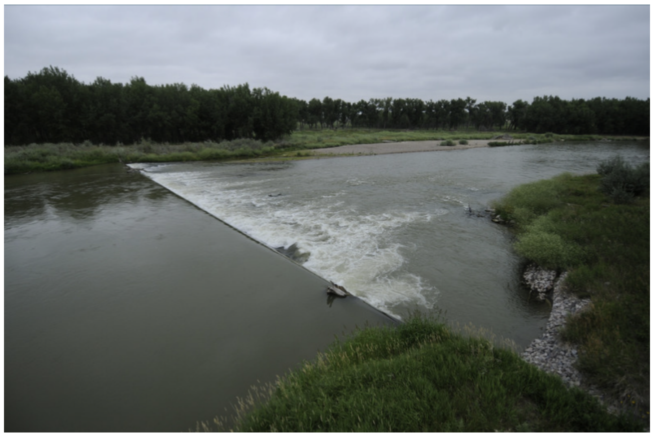


Fig. 9. Original Images

For the neural networks we tried different ways to clean the variables. First we tried with the original images without removing variables or editing the pictures and then we divided the data set into three parts which was the training, testing and validation. The training part had the most data, then the test part had half of the training data and the validation part had the least.

Train	25,235
Test	12,618
Validation	4,206

Table 6. Dataset Separation.

Once we separated the data set we extracted the precision as well as the errors and the errors that we extracted were mean squared error, root mean squared error, and mean absolute error.

MSE	0.123
RMSE	0.35
MAE	0.22

Table 7. Error from original image.

We can see from the data that the error is not huge but it is not so low either, so this model is not as accurate as we wanted it to be.

We then proceeded to edit the images by cutting the edges of the images and separated the datasets once again into training, testing and validation, and still got the same errors as mentioned above.

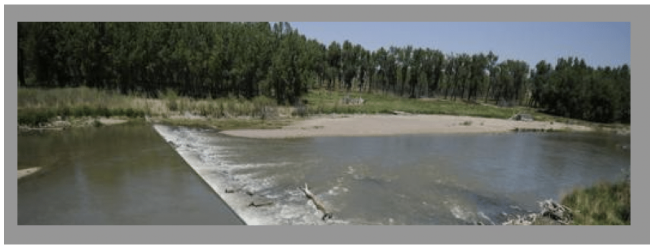


Fig. 10. Cropped Edges from Images

MSE	0.207
RMSE	0.45
MAE	0.313

Table 8. Error from cropped edges.

If we analyze the data we can see that in this way the error is almost doubled, and we obtained an even less accurate model, so removing the edges of the images was not a very good idea.

Finally we decided to crop the image as much as possible to focus on the water foam and separate the data in the same way as previously mentioned. We can see that the error of this model is smaller than cropping the edges of the image, but it is larger than keeping the original image so that this model is also not so accurate. The model that was not so bad is to keep the original images without cropping anything from them.

MSE	0.158
RMSE	0.398
MAE	0.253

Table 9. Error from water foam cropping.

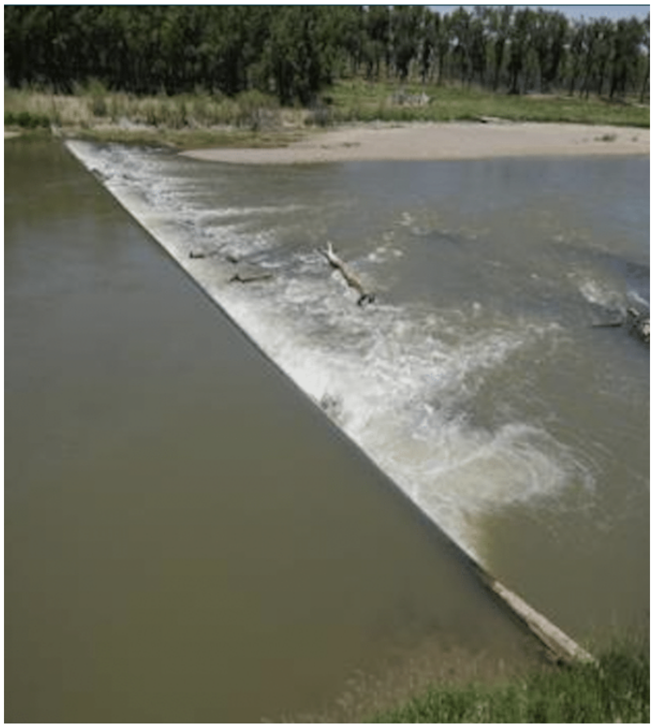


Fig. 11. Cropped Water Foam from Images

The next thing we did was to focus on separating the data by season and by year, instead of editing the images. We decided that this would be a good idea since the water in a river is affected by the seasons as the weather and temperature changes. Let's first focus on separating the images by season no matter what year it was.

Then we decided to separate the data by year to see how much it changed and to visualize the images. The separation of the data was the training set from 2012 to 2016, the test set from 2017 to 2018, and the validation set in the year 2019. This left approximately 21,000 data in the training, 15,000 data in the test and 5,000 data in the validation.

MSE	0.27
RMSE	0.52
MAE	0.34

Table 10. Errors of separating data by year.

When running this data in the Neural Network we found that the error was not as low as we expected so we could conclude that separating the data by year is not as efficient and cropping the images is not as efficient either.
With these results we realize that working with images is not as useful because what happens with these models is that if it were a person, and if we are trying to find out the depth of a river glancing at some images, this can be difficult because it is not as accurate just by seeing a photograph. For a model, it can be an arduous task if there are not enough images.

We decided to look for alternatives to work with these images

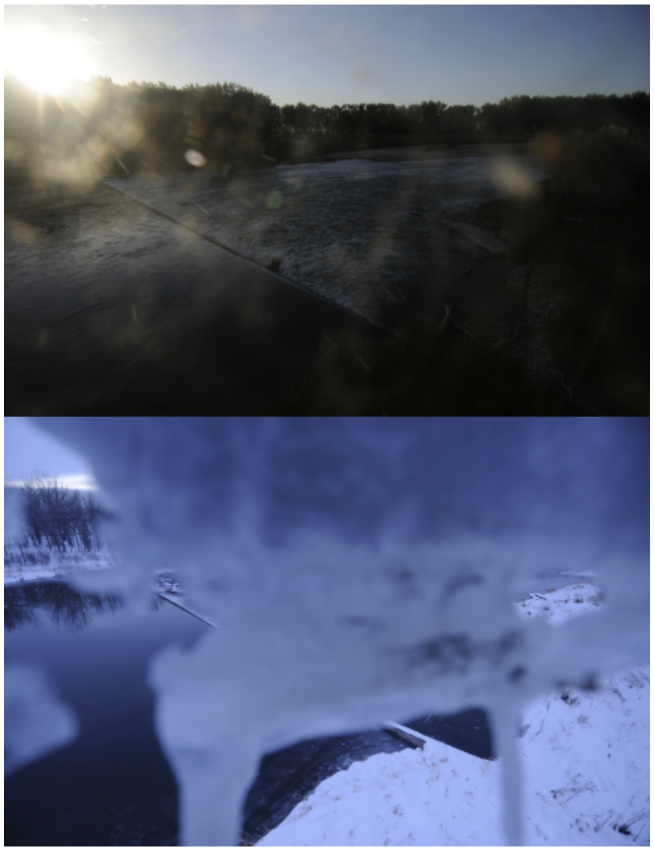


Fig. 12. Images of different seasons

and we realized that there is the segmentation of them. Basically what we try to do with this technique is to separate the elements of the image to leave the water of a single color and make it easier to create a predictive model that can serve us in the future for certain bodies of water that are in different locations, and thus be able to work in hydrology studies to optimize water distribution, predict droughts and floods, etc.

Segmentation. As a final idea to be able to go deeper in the handling of images for the prediction of variables, we tried to perform segmentation on the images to separate, based on the colors of the image, the different parts that compose it, this includes the water zone and the shore, vital parts for the correct analysis of the photographs. To achieve this we imported a pre-trained segmentation model that is available in Github [], we only passed the training images, training annotations and the number of epochs for which it would train.

After that it was simply a matter of passing an image as input and observing the result image. Although we got acceptable results as observed in the image 13, it is still not a model that can be applied to all our images, especially those taken on snowy days because the large amount of white color in the image does not allow the model to discern between the different elements of the scenario.

We believe that segmentation is a good approach to what could be a viable solution to the challenge, it is a matter of having more time to train the model in question, see different ways to separate the pixels, and see how to handle special cases such as snow.

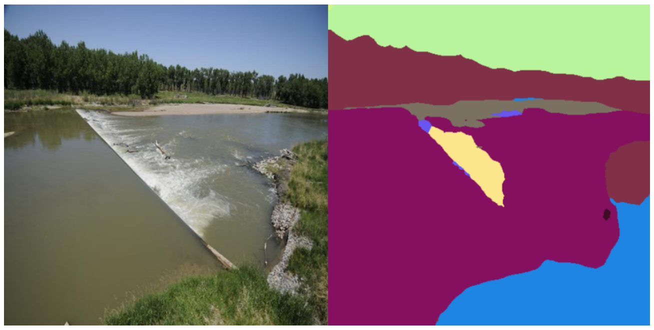


Fig. 13. Resulting image of our segmentation model.

Conclusions

For daily water management, flood forecasting and management, determining compliance with water use agreements, and for the design of reservoirs, water supply systems, and bridges, accurate measurement and modeling of stream stage and discharge are crucial. The project's goal was to fill in data gaps for stream stage by automatically extracting features from image series using machine learning models for one location with a single camera in order to determine whether it would be feasible to apply these techniques to other locations with various image formats, lighting conditions, etc.

Along with finding features that can properly predict stage and discharge in simple situations for a specific location, it's crucial to make sure that the data utilized to develop a model in the training set encompasses the full range of values found in the test set. The model's predictions become more questionable if they are used to forecast stream stage and discharge for values other than those in the trained data set. To determine whether the training set is sufficient to create a good model to close the gap, photographs from the test set can be compared to images from the training set.

Another goal of the project was to create a foundation of software libraries that are well-suited to carry out hydrology-centric research methods in a manner that enables hydrologists to think more about hydrological questions and less about the technical aspects of the research tools. Therefore, it would be ideal for these libraries to be developed further so that they could employ more than just photos and stage and discharge measurements, compute image characteristics, and combine the data for machine learning tasks.

Bibliography

1. Agarwal, A., M.S. de los Angeles, R. Bhatia, I. Cheret, S. Davila-Poblete, M. Falkenmark, F.G. Villarreal, T. Jonch-Clausen, M.A. Kadi, J. Kindler, J. Rees, P. Roberts, P. Rogers, M. Solanes and A. Wright, 2000: Integrated water resources management, TAC Background Paper No. 4, Stockholm, Sweden, Global Water Partnership.
2. Cohn, T.A. and H.F. Lins, 2005: Nature's style: naturally trendy, Geophysical Research Letters, v. 32, no. 23, doi 10.1029/2005GL024476.
3. Dou, G.; Chen, R.; Han, C.; Liu, Z.; Liu, J. Research on Water-Level Recognition Method Based on Image Processing and Convolutional Neural Networks. Water 2022, 14, 1890. https:// doi.org/10.3390/w14121890
4. Koutsoyiannis, D., 2003: Climate change, the Hurst phenomenon, and hydrologic statistics, Hydrological Sciences Journal, v. 48, no. 1, pp. 3-24.
5. Qiu, R., Cai, Z., Chang, Z. et al. A two-stage image process for water level recognition via dual-attention CornerNet and CTransformer. Vis Comput (2022). https://doi.org/10.1007/s00371-022-02501-6