STUDENT SUMMER INTERNSHIP TECHNICAL REPORT

Automatic Monitoring of Water Seepage in the F-Area 3 Basin Cap

DOE-FIU SCIENCE & TECHNOLOGY WORKFORCE DEVELOPMENT PROGRAM

Date submitted:

December 15, 2023

Principal Investigators:

Aris Duani Rojas (DOE Fellow Student)
Florida International University

Timothy Johnson, Ph.D. (Mentor)
Pacific Northwest National Laboratory

Ravi Gudavalli, Ph.D. (Program Manager)
Florida International University

Leonel Lagos, Ph.D., PMP® (Program Director)
Florida International University

Submitted to:

U.S. Department of Energy
Office of Environmental Management
Under Cooperative Agreement # DE-EM0005213

FIU | Applied Research Center
FLORIDA INTERNATIONAL UNIVERSITY

EXECUTIVE SUMMARY

This research work has been supported by the DOE-FIU Science & Technology Workforce Development Initiative, an innovative program developed by the U.S. Department of Energy's Office of Environmental Management (DOE-EM) and Florida International University's Applied Research Center (FIU-ARC). During the summer of 2023, a DOE Fellow intern, Aris Duani Rojas, spent 10 weeks doing a summer internship at Pacific Northwest National Laboratory under the supervision and guidance of Dr. Timothy Johnson. The intern's project was initiated on June 5, 2023, and continued through August 11, 2023, with the objective of using deep learning algorithms to develop an automatic system to detect abnormal conductivity values in the Savannah River Site's F-Area Basin 3 Cap. The purpose of this system is to understand the seepage of water through the basin cap by analyzing Electrical Resistivity Tomography (ERT) data collected on the site.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Between 1955 to 1988, the F-Area Basins at the U.S. Department of Energy's (DOE) Savannah River Site (SRS) received approximately 1.8 billion gallons of low-level acidic waste from the processing of uranium slugs and irradiated duel in the F-Area Separations Facility. The acidic waste contained a variety of radionuclides and dissolved metals that, after entering the basin, either evaporated or seeped into the underlying soil. This seepage contaminated the groundwater with radionuclides such as plutonium isotopes, cesium-137, strontium-90, uranium isotopes, iodine-129, technetium-99, and tritium. [1]

As part of the remediation efforts, in 1991, the basins were closed by dewatering them, stabilizing the remaining waste, and covering them with a low-permeability, multilayer cap to reduce the rainwater infiltration. In 1997, a pump-treat-and-reinjection system was introduced that trapped untreatable tritium in a continuous loop. In 2004, the pump-treat-and-reinjection system was replaced with a hybrid funnel-and-gate system that aimed to slow the migration of the contaminated groundwater and funnel it through treatment zones in the gates. There have also been many geophysical surveys carried out in the wetlands to map radiological hotspots and monitor seasonal behavior of contaminants. After many years of active treatment, low levels of radionuclide contamination still remain in the groundwater as it exits the treatment zones. [2]

For the active treatments to remain as effective as they currently are, an important objective is to ensure that the different zones of vulnerability remain protected. One zone of vulnerability, and the focus in this report, is the cap on each of the basins. If these caps begin to lose effectiveness and allow rainwater infiltration, they can cause more radionuclides to be mobilized and migrate to the groundwater. As such, to monitor the structural integrity and effectiveness of the caps, a technique known as Electrical Resistivity Tomography (ERT) was used. In July 2022 DOE scientists involved in the Advanced Long-Term Environmental Monitoring Systems (ALTEMIS) Project and Savannah River Nuclear Solutions (SRNS) Area Completion Project (ACP) worked collaboratively to install an ERT array at the F-Area Hazardous Waste Management Facility (FHWMF) along the cap of F-Area Basin 3. The goal of ERT is to monitor the seepage of water through the clay layer in the cap by measuring its electrical conductivity (and how it changes over time.) [3]

During the summer of 2023, DOE Fellow Aris Duani Rojas, with the support of his mentor, Dr. Timothy Johnson, developed an automatic system that can determine if there were any abnormally high conductivity values, where those values were, and how severe they were given the conductivity values obtained using ERT. This can then be translated into determining if there was any rainwater seepage and when and where it occurred. These results are automatically converted into 3D visualizations that a human user can move, pan, and zoom in. A human user can quickly review the results of the system and determine whether there has been any seepage or not. They can then open the visualization if they think further analysis is needed. This eliminates the need for manpower to manually monitor, analyze, and visualize the conductivity values every day to

determine if there are weaknesses in the cap, to significantly reduce overall monitoring cost and risk.

2. RESEARCH DESCRIPTION

This research problem can be defined as anomaly detection in the conductivity values obtained from ERT measurements collected at the SRS along the cap of F-Area Basin 3. The data available included 729 files, each containing the conductivity values for 1,465,082 points in an XYZ space. Since there is currently no rainwater seepage through the cap, the 729 files of data can be considered as baseline data. Timeseries data were also available which included 32,142 rows of temperature and rainfall readings in 15-minute intervals.

A popular and effective deep learning algorithm to detect anomalies is the autoencoder. It has two parts: an encoder and a decoder. The encoder works by taking an input X and mapping it to an embedding tensor Z. The decoder works by taking the embedding tensor Z and mapping it to an output Y. There are constraints placed on the shape of the embedding tensor Z and the output tensor Y. First, the shape of the embedding tensor Z must be less than the shape of the input X. Second, the shape of the output Y must be equal to the shape of the input X. The first condition forces the Deep Neural Network (DNN) to learn the patterns in the data when compressing the input X to embedding Z, otherwise the DNN could just become an identity function. The second condition is to allow the DNN to compare the reconstructed output Y with the original input X since they both have the same shape.

The concept is that the baseline data X is fed as input to the DNN where it gets compressed and decompressed into the output Y. The DNN can then evaluate the difference between X and Y and learn how to better compress and decompress the baseline inputs. When anomalous data is given to the DNN as input, the compression and decompression will give an output that is different from the input. This is because the DNN does not know how to compress and decompress anomalous data. The difference between the input and the output will aid in determining if there is an anomaly, where it is, and how severe it is, which is all the information required.

A popular and effective deep learning layer used for time-series data analysis is the Long Short-Term Memory (LSTM) layer. The main idea behind the LSTM layer is that it outputs both an output value and a memory value for itself. This way the layer remembers what happened previously, though how much it remembers is determined by weights within the layer. The temperature and rainfall data of the past N timesteps before the conductivity values were obtained can be processed through multiple of these layers, and their output can be combined with the embedding tensor Z. This combination will change the embedding tensor given to the decoder, which will allow the DNN to learn the impact of the temperature and rainfall values over time on the conductivity values measured by ERT.

Train, Validation, Test Split

The 729 files of available data were split into 70% training, 20% validation, and 10% testing. When dividing the dataset, it is important that the training, validation, and testing sets have a similar distribution as that of the population. The reason why this is important is to ensure that the model can generalize well, and the measure of its performance is accurate. If the train and/or validation distributions are too different from the test and/or population distribution, then what the model learns from training and validation will not carry over for testing or prediction of future data. Beyond that, during development, the performance of the model will seem better than what it truly is due to the difference in distributions.

An approximation of the mean and standard deviation for each point of the true dataset was therefore calculated by sampling a few of the 729 files at random and calculating the mean for each point. That result was considered one sample. The process was then repeated until hundreds of samples were acquired, and then the mean and standard deviation for each point of those hundreds of samples was computed. This mean and standard deviation approximates the true mean and standard deviation of the population since the Parent Distribution Mean and the Sampling Distribution Mean are the same, and the standard deviation of the Parent Distribution divided by the square root of the sample size is equal to the standard deviation of the Sampling Distribution. Whether the conductivity values follow a normal distribution or not is of no significance since by the Central Limit Theorem, as the sample size increases, the sampling distribution converges to the same mean and standard error as the sampling distribution for normal distributions. With this approximation of the population distribution, the train, validation, and test sets were split in such a way that their mean was within one population standard deviation of the population mean.

Preprocessing

Since the conductivity values were near zero (thousandths place), the temperature values in the tens place, and the rainfall values in the tenths place, a standard scaler was used for preprocessing and converting the data to the same distribution with mean of 0 and standard deviation of 1.

The use of Principal Component Analysis (PCA) was considered as a dimensionality reduction technique given that each input has over 1 million values and most of them are highly correlated with each other. However, the issue with PCA, is that slight changes to the input of PCA can lead to large changes in the output vector. For example, let the PCA operation applied on an input X be $f(X)$, then $f(X) = Z$. If the conductivity values are increased around a specific place to get $\overline{X}$, then $f(\overline{X}) = \overline{Z}$. Even though X and $\overline{X}$ differ only on one area, the usage of PCA can and will lead to a Z and $\overline{Z}$ that differ everywhere. The vice versa is true, if Z and $\overline{Z}$ differ in one area, the inverse of PCA can and will lead to an X and $\overline{X}$ that differ everywhere. This conflicts with our goal of finding the exact locations of the anomalies because if the input has an anomaly in one place, PCA can give an output that has an anomaly everywhere; or if the prediction of the model has a slight error somewhere, the inverse of PCA can lead to an output that is different from the input everywhere, so it may seem like there is an anomaly everywhere.

The conductivity values represent tetrahedrons from a mesh in 3D space, which will be referred to as points for simplicity. The 1,465,082 conductivity values form 410,227 points with their own x, y, z, and conductivity values. Since the points exist in XYZ space, it is possible to rearrange them into a 3D tensor, so that during training the spatial correlation between points provides more information to the DNN about the patterns in the data. However, the points in the 3D space are sparse. When those points are converted to a 3D tensor, they do not fill the entire tensor, but leave most of it empty. For example, if there are 3 points at (0,1,2), (1,2,0), and (2,0,1), it would take a 3 x 3 x 3 tensor to represent those points in space, but those points occupy only 3 out of 27 spaces in the 3D tensor. This is an extreme example, but what occurred during preprocessing would require hundreds of terabytes to allocate one sample of points with 4 bytes per conductivity value.

To resolve this issue, the points in 3D space were compressed by reindexing on every axis. The first axis is kept the same, so for a given value X from the first axis, the coordinates in the second axis are reset to start from 0. Subsequently, for a given value Y in the second axis, the coordinates in the third axis are reset to start at 0. For the previous example (0,1,2) would be converted to (0,0,0), (1,2,0) would be converted to (1,0,0) and (2,0,1) would become (2,0,0). If there was another point (0,2,2), it would be (0,1,0). This greatly compresses the sparse points while maintaining the spatial correlation. In the actual implementation, the 3D tensor was arranged in a ZYX format because it was required to properly use the 3D Convolutional Layers on that tensor.

The use of graphs (or meshes) to represent the points in the 3D space was investigated, but it was not deemed to be any better than the compressed tensor approach due to some considerations discussed in the next section.

Training

During development, there were three main layers considered when building the Autoencoder architecture. The first one used Linear layers and was referred to as Linear Autoencoder. The second used 3D Convolutional and 3D Convolutional Transpose layers and was referred to as Convolutional Autoencoder. The last one used Graph Convolutional layers and Graph Convolutional Transpose layers and was referred to as Graph Autoencoder.

After experimenting with the Linear Autoencoder, it was found that, like PCA, it did not do a good enough job of preserving the location of the anomalies. Changes to the input of the Linear Autoencoder in one area led to changes in the output of the Linear Autoencoder in most of the conductivity points. It would predict most of the conductivity values are anomalous even though most of them had remained normal when the input was changed.

The issue with the Graph Autoencoder, is that Graph Convolutional Layers and similar approaches using GNNs would compress and decompress the graph, so the reconstruction is not just about the conductivity values, but the x, y, and z values as well. That is not helpful since prediction of the x, y, and z values on top of the conductivity values introduces unnecessary sources of error. To avoid compressing the x, y, z, the graph would have to become fixed, so applying the convolutional layers without reducing the size of the graph would lead to the risk of the Graph Autoencoder

becoming an identity function. In addition, if the size of the fixed graph were to be reduced, then it would be almost the same concept as that of the Convolutional Autoencoder. Due to these considerations, the representation of points in 3D space as a graph and the subsequent use of a Graph Autoencoder was not deemed to have an advantage over the compressed tensor and the Convolutional Autoencoder. At best it would give similar results, but it seemed likely that the Graph Autoencoder would do worse.

The selected architecture was therefore the Convolutional Autoencoder, which is more capable of determining the presence of anomalies, where they are, and how severe they are, than the other layers explored. As previously mentioned, the rainfall and temperature data were handled through LSTM layers.

Finding the best DNN architecture for a given dataset is not straightforward, but rather requires a mix of trial-and-error and intuition. As such, most of the architecture was parameterized, so parts of it like the number of 3D Convolutional layers, size of the 3D kernel, stride, padding, up sampling mode, number of LSTM layers, LSTM steps, nodes per LSTM layer, batch size, epochs, learning rate, weight decay, etc. could be easily changed. The activation function for the 3D Convolutional and 3D Convolutional Transpose layers was the Hyperbolic Tangent (tanh) function, and the activation function the LSTM layers was the Rectified Linear Unit (ReLU) function. The loss function of the network was Mean Squared Error (MSE) where the model prediction being off for a given conductivity value is highly punished.

Using the parametrized architecture, a Random Search algorithm was run through 50 combinations of parameters where the DNNs were trained and their training and validation MSE per epoch were recorded. The architecture with the best training and validation MSE was then chosen as the best architecture, and the weights for the epoch that had the best training and validation MSE were chosen as the best weights for that architecture.

Prediction

For a given sample, once the model compresses and decompresses it, the output will have the same shape as the input, thus it is possible to calculate the difference between the output and input to gain a measure of how well the model did. If the model learned the correct patterns in the data, it is extremely successful in giving an output close to the input when the data is normal (small difference), and an output not too close to the input when the data is anomalous (large difference.) However, the difference is not a good measure of the performance because a difference of 0.001 could be good or bad depending on the scale of the data. As such, when predicting, the percent error between the input and output is calculated to gain a better sense of how significant the difference truly is. Once there is a percent error for every conductivity value in each sample, the question remains: What is a good threshold? That is, it is not known how much percent error on a normal sample is normal and how much percent error on an anomalous sample is anomalous.

The idea of a single static threshold applied for all points can be quickly discarded. There are some points for which the model's output matches the input extremely closely and others for which it is

not so close. To apply a single static threshold that is too low runs the risk of marking the points in which the output is not too close to the input as anomalous even when they are normal. Yet to apply a single static threshold that is too high runs the risk of marking the anomalous points as normal points, especially for the points where the percent error is low, thus a threshold is needed for every point.

It is possible to use Tolerance Intervals to bind, with α confidence, β proportion of the population. Finding the Tolerance Interval for each percent error between the input and output in the training set will give an estimate of the thresholds that determine what is a normal percent error for each point in a sample. One issue, though, is that the conductivity values are not normally distributed in the training data. Applying either a Shapiro-Wilk Test or a Kolmogorov-Smirnov Test will result in the vast majority of the points failing those normal distribution tests.

There exists a way to calculate Tolerance Intervals using bootstrapping [4]. The algorithm used is known as Content Corrected k-Factor Tolerance Limits Based on Bootstrap. The main idea behind the algorithm is to use sampling with replacement to obtain an interval that contains at least β proportion of the population with approximate confidence α.

Since extremely low conductivity values are not of interest due to the seepage of water being correlated with increased conductivity values, only the upper Tolerance interval was calculated with 99% confidence that the interval contains 99% of the population.

Thus, the prediction process for a given sample is as follows: input the sample to the Convolutional Autoencoder and obtain an output, calculate the percent error between the input and the output, and adjust each percent error based on the corresponding tolerance interval by subtracting them and turning all negative values to zero. The result is a prediction of where the anomalies are (non-zero points) and how severe they are where a bigger percent error means a bigger discrepancy between the conductivity values expected by the model with the tolerance intervals and what the input conductivity values were.

Visualization

For humans to better understand where the anomalies are and their severity, a visualization script was implemented. The visualization script takes in the final percent error result from prediction and the x, y, and z coordinates and maps those percent errors to a 3D space with different color intensities. The specific colors as well as the thresholds where one color turns to the next can be user-defined. Lastly, the 3D space is saved as a webpage that the user can double click, render the 3D space in the browser, pan, zoom, and move around in it. This means that no packages or extra software are needed to analyze the results beyond a web browser.

3. RESULTS AND ANALYSIS

The results of using random search for hyperparameter optimization during training are shown below.

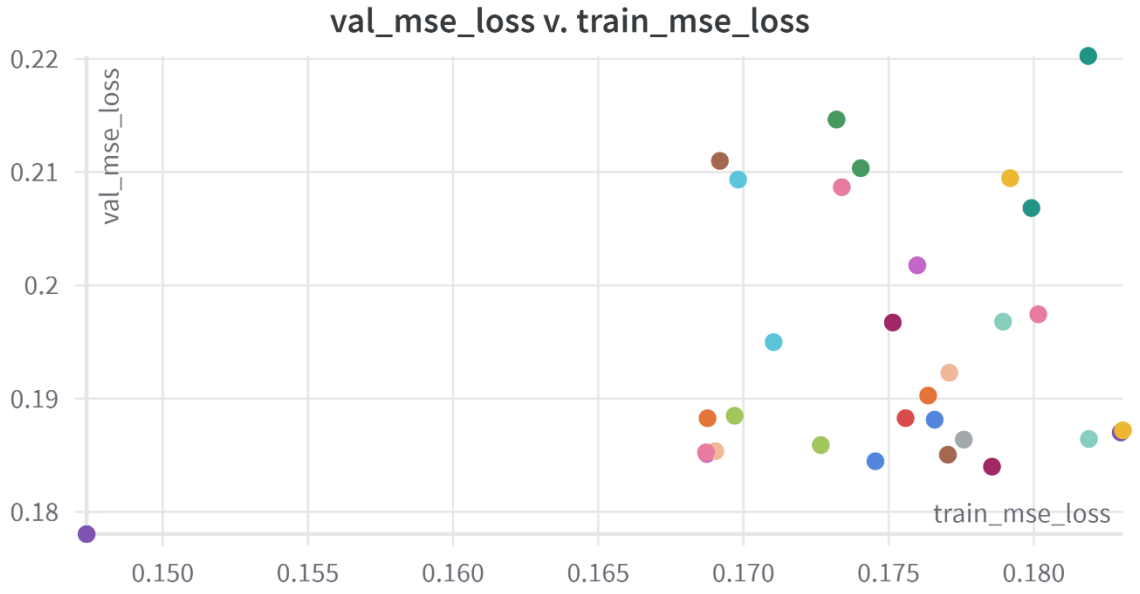

Figure 1. The MSE of the validation set for each run compared to the MSE of the training set.

These results show that there is one run that has the lowest mean squared error (MSE) loss on both the training and validation sets. This run is therefore considered the best run amongst the set of 31 runs. The runs were stopped at 31 instead of the planned 50 as the best run was found early and none of the 20+ runs after that were close to that one run.

This run describes the architecture that has the best learning capabilities, and thus the best hyperparameters for the Convolutional Autoencoder architecture. These best hyperparameters found included 5 convolutional layers with a 4 x 4 x 1 kernel and a stride of 2, 3 Long Short-Term Memory (LSTM) layers with 71 nodes within each layer, 152 timesteps in the LSTM layer input (with 15-minute intervals between each timestep), a batch size of 43, and 85 epochs of training time. There was also a LSTM dropout layer of about 0.2 between each LSTM layer. Lastly, the learning rate was about 0.005 and the weight decay around $2 * 10^{-7}$.

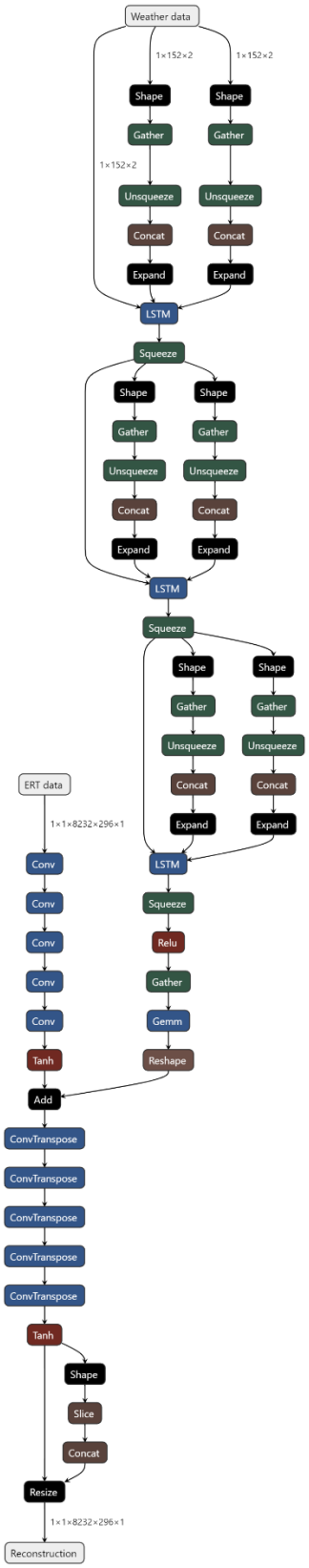

Figure 2. A visualization of the architecture that had the lowest training and validation loss.

To evaluate the architecture (and the corresponding weights that resulted in the lowest training and validation loss), there are two assessments required: first, assessment of the performance of the model on normal data and second, assessment of the performance of the model on anomalous data.

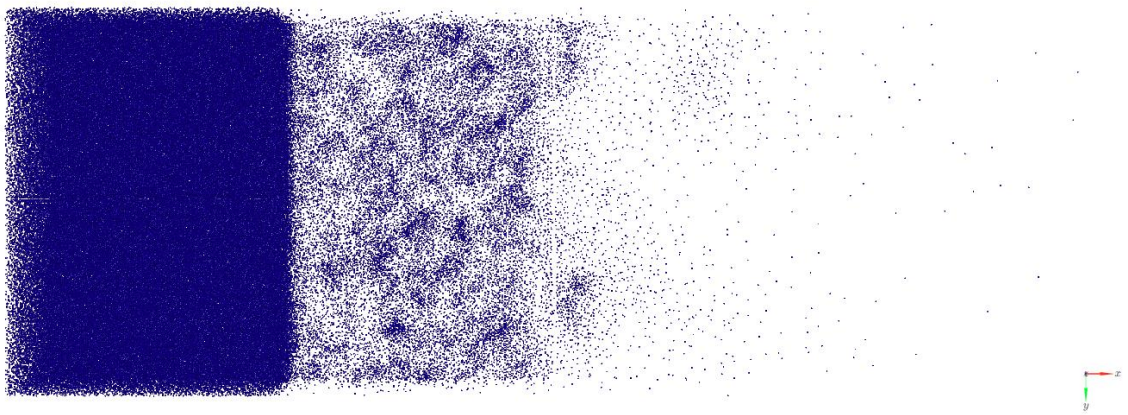

Figure 3. Reconstruction error of the Convolutional Autoencoder on a normal sample.

Figure 3 shows a bottom view of the 3D space, where the z-axis is facing away from the screen. The blue points are tetrahedrons where the model with tolerance intervals has a reconstruction error less than $1 * 10^{-6}$.

A sample of the normal ERT data can be obtained from the test set. However, to obtain a sample of anomalous data, it must be artificially generated. This can be done by finding the midpoints of all tetrahedrons, choosing one of them as the center, and increasing the conductivity of all points in a D radius by a factor N. As such, it is possible to take a random midpoint somewhere beneath the basin cap and increase the conductivity in a 5-meter radius by 30%.

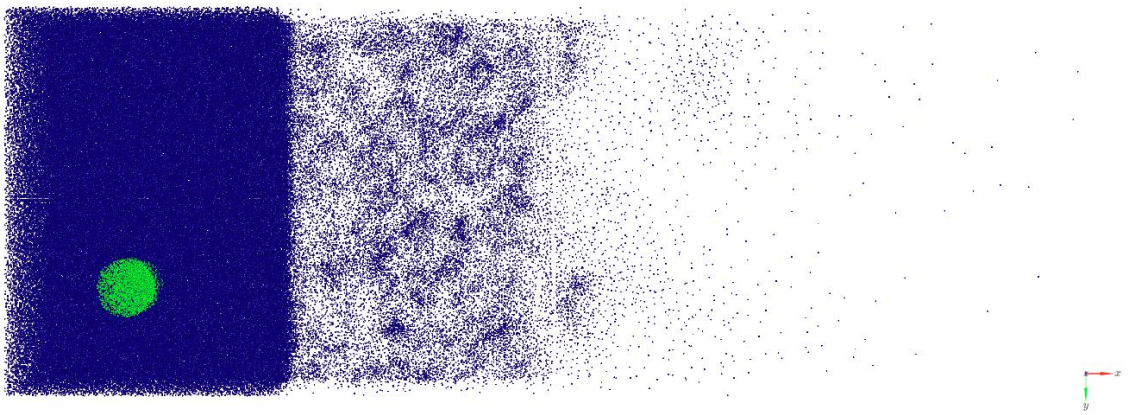

Figure 4. The expected reconstruction error of the Convolutional Autoencoder on an anomalous sample.

The image above shows the results expected if the model can perfectly detect every single point where the conductivity was increased by 30%. The blue color indicates points where there should

be no reconstruction errors and the green color indicates points where there should be reconstruction errors.

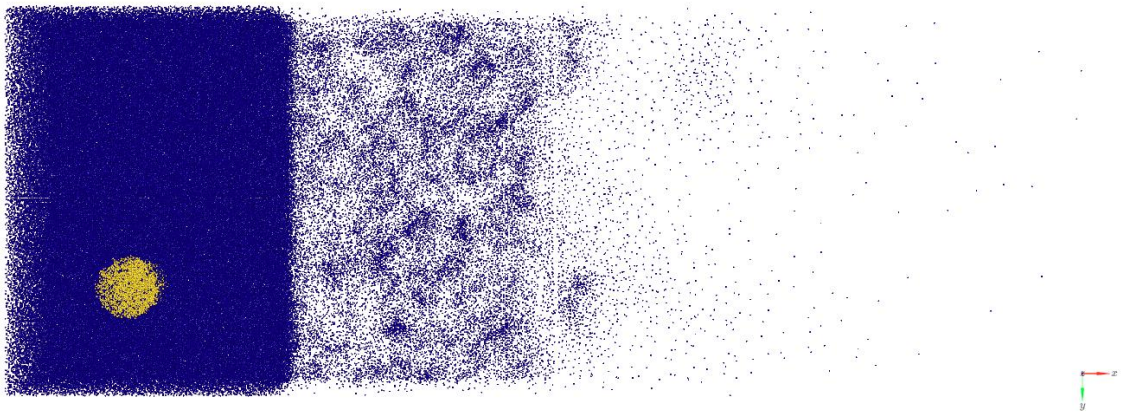

Figure 5. The actual reconstruction error of the Convolutional Autoencoder on an anomalous sample.

The image above shows the reconstruction error after passing the anomalous sample as input to the model, predicting on it, and then applying the tolerance interval thresholding. The blue points indicate a reconstruction error below $1 * 10^{-6}$ and the yellow points indicate a reconstruction error above that.

From these results, it is clear there is a high fidelity between what the model receives as input and what it outputs. When given normal data as input, the model's reconstruction error is near zero. And when given anomalous data as input, the model's reconstruction error is high only on the points that are anomalous. That is, the model's reconstruction error is high only on points whose conductivity values are deemed to be unusually high, and the model's "judgement" gives the desired results since the defined ground truth matches the actual reconstruction error.

However, the model's performance is not the same throughout the predictions across the year. The sample shown above is from October 2022. Samples from months close to October exhibit similar performance where the results are nigh on perfect. On the other hand, results around April 2023, and months close to April, are less outstanding. It seems that with the changes in seasons, there is not insignificant change in the conductivity values that even the model has difficulty adjusting to it.

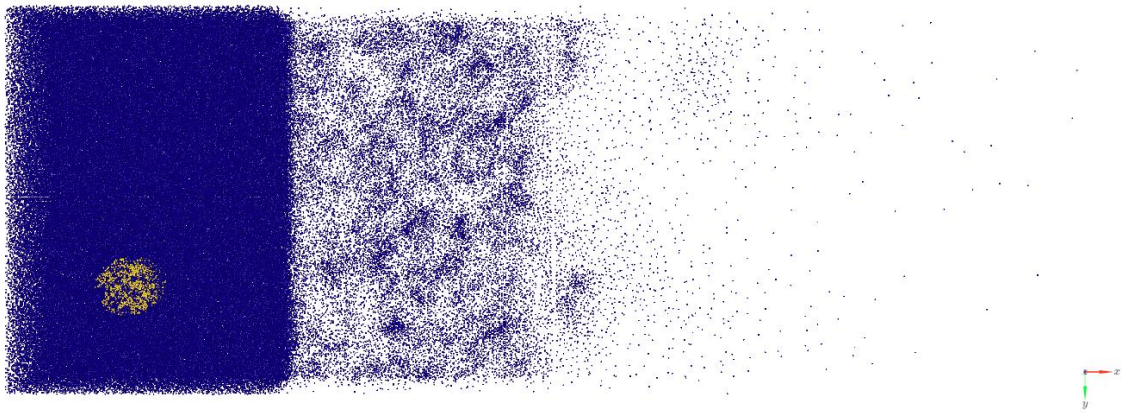

Figure 6. Example of one of the worst results from the month of April 2023.

As seen from figure 6, the actual reconstruction error still matches the trend of the expected reconstruction error. There are less points marked as anomalous, but there are still enough to determine exactly where the anomaly is and how severe it is. Moreover, as the radius and/or magnitude of the conductivity anomalies increases, the performance of the model in April increases. But there is a clear decrease in performance between months like October and April.

A possible cause considered is that there is a difference in the rainfall and temperature between those months. Yet, the model was given rainfall and temperature data to learn from and adjust its predictions. It may be possible that the rainfall and weather signals to the model were not strong enough, or that they are negligible in some seasons and crucial in others, which could cause the model to not properly learn how to use them. It could also be that the conductivity values are lower around April than around October, so a 30% increase in conductivity in April does not seem as much of an anomaly to the system as it does in October due to the way the thresholds are created. Higher conductivity value increases e.g., 60%, do lead to much better results across all samples. However, it is possible that there is another variable not considered that is causing these changes, which can include some soil property changes, noise/measurement errors, or some other unknown not given as input to the model.

Exploring the true cause for the discrepancies between the prediction results in anomalous data depending on the month (or season) and determining how to rectify these discrepancies is a crucial step to further improving the results of the system, a consideration for future work.

4. CONCLUSION

A system to automatically detect the presence, location, and severity of anomalies in ERT data was successfully developed using Convolutional and LSTM layers in an Autoencoder algorithm. The results of the system can be translated to the presence of water below the surface, which will help DOE personnel better assess the structural integrity of the F-Area Basin 3 cap and its effectiveness in preventing water seepage. Also, the system removes the need for a human to analyze the raw ERT data, but they can then instead quickly and easily analyze the visualization of the system's results if needed. This will allow workforce and other resources to be focused on other areas that need the DOE's attention.

In addition, while the current system was built using the F-Area Basin 3 ERT data, it can be effortlessly extended to other caps and similar ERT data monitoring projects. The re-training of the model to find the best architecture and weights, as well as the re-calculation of the tolerance intervals can be done by changing a few configurations and running a few scripts. Besides that, nothing else changes if the cap under monitoring changes. This makes the system adaptable to new areas of interest and/or significant changes to previous areas of interest where previously measured data is made irrelevant.

Thus, an effective tool to monitor the water seepage given ERT data has been created for the F-Area Basin 3 cap and similar places.

5. REFERENCES

1. Washington, D.C.: Interstate Technology and Regulatory Council, Remediation Management of Complex Sites Team. (2017). *Remediation management of complex sites*. Remediation Management of Complex Sites. https://rmcs-1.itrcweb.org/6-13-savannah-river-site-srs-f-area-seepage-basins-groundwater-south-carolina/

2. ALTEMIS Team. (2023). *ALTEMIS Fact Sheet*. ALTEMIS: Advanced Long-Term Environmental Monitoring Systems. https://www.srs.gov/general/srnl/factsheets_2023/altemis-fact-sheet-print.pdf

3. Johnson, T.C., Strickland, C., Knox, H.A., Thomle, J.N., Vermuel, V.R., Ulrich, C., Kneafsey, T.J., Doug, & Blankenship (2019). *EGS Collab Project Electrical Resistivity Tomography Characterization and Monitoring*. Semantic Scholar. https://www.semanticscholar.org/paper/EGS-Collab-Project-Electrical-Resistivity-and-Johnson-Strickland/b1c6879beff5dfe45a77b32303b3a4f1cf8ec769

4. Koski, T. (2011). *Tolerance Intervals with a Computer*. Bayesian statistics for Computer Intensive Methods. https://www.math.kth.se/matstat/gru/sf2955/tolerans.pdf

APPENDIX

Table 1. Train and Validation MSE loss data for each run.

Name	Train MSE Loss	Validation MSE Loss
Run 0	0.175143184	0.196717994
Run 1	0.177040991	0.185042394
Run 2	0.172664078	0.185895031
Run 3	0.177089404	0.192281671
Run 4	0.181873495	0.220255679
Run 5	0.169816657	0.209339124
Run 6	0.179186282	0.209460454
Run 7	0.168737823	0.185114585
Run 8	0.17894202	0.196789542
Run 9	0.168763029	0.188264981
Run 10	0.180153807	0.197442356
Run 11	0.182994021	0.187009071
Run 12	0.174040784	0.210346125
Run 13	0.173386112	0.208668079
Run 14	0.176584507	0.188137872
Run 15	0.177589393	0.186379065
Run 16	0.178560288	0.183991797
Run 17	0.169187198	0.21098793
Run 18	0.169695603	0.188484063
Run 19	0.169030467	0.185351784
Run 20	0.179917443	0.206826912
Run 21	0.171039006	0.194987462
Run 22	0.183068363	0.187193165
Run 23	0.175986097	0.201765934
Run 24	0.181897802	0.18643184
Run 25	0.176360668	0.190267457
Run 26	0.16871109	0.185251365
Run 27	0.147378004	0.178033901
Run 28	0.173205549	0.214638989
Run 29	0.175584171	0.18827203
Run 30	0.174545503	0.184466976

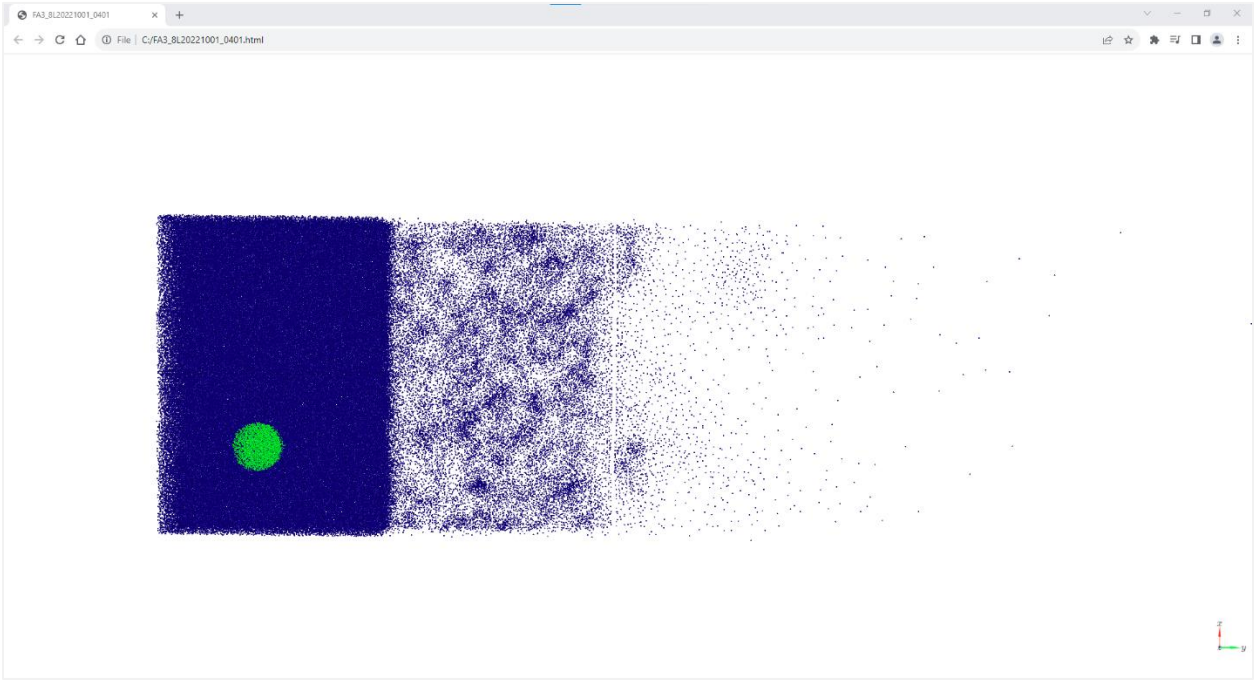

Figure 7. Browser-based 3D visualization that user can load pan, zoom, and move in.