Reflections of Bikes and Washing Machines:

Echo State Network Performance Compared to Dominantly Used Classifiers

James Daus

Colgate University

Abstract

Echo State Networks (ESNs) are an emerging architecture for machine learning based on reservoir computing, with the unique qualities of data dimension reduction and variable temporal memory, and the additional benefit of very little required training. Preliminary results from research on the subject have been promising, especially on chaotic time series data, though no experiments have been done in the important field of network traffic classification (Bianchi et al. 2018). Our findings indicate and further support the claim of ESN effectiveness and versatility on a variety of time series data, and offer suggestions for future research into ESN performance on IoT and network data, such as giving special attention to tweaking the spectral radius of the reservoir to fit the data, and to the weight distributions within the reservoir connections.

1. Introduction

Reservoir Computing is the process of using a simple classifier trained on the output of a physical or digital dynamical recursively-connected reservoir to achieve complex results with lower training cost, and has been a known system for over two decades. (Schrauwen, Verstraeten, and Van Campenhout 2007). One such architecture, known as an Echo State Network (ESN), utilizes an achievable property in certain reservoir organizations in which data is stored in moving patterns within the network, known as a reflection (Jaeger 2003). These reflections enable even a very simple classifier to be informed by both temporally close and distant data, saving on the costs associated with other complex recurrent neural network architectures that can achieve this result. However, there has been limited research and usage of this architecture, and its performance across many difficult time series problems is unknown. Our goal with this study was to further investigate the efficacy of ESNs on complex and simple time series problems.

We selected a simple classification case of identifying the time based on human bike traffic through Fremont Bridge in Seattle, as well as a complex classification problem of identifying IoT machine activity based on raw network packet captures. In each case, the Echo State Network was compared against the following off-the-shelf architectures from Scikit-learn and Keras: Support Vector Machine, Random Forest Classifier, Feedforward Neural Network, 1D Convolutional Neural Network, and Long Short-term Memory. Architectures were evaluated using precision-recall and receiver operating characteristic graphs as well as confusion matrices.

After hyperparameter optimization using a grid search over key hyperparameters for each architecture including the ESN, the performances were roughly the same. For the bike traffic data, each architecture achieved around 93% accuracy once fully trained. This is likely near the maximum accuracy due to variations in the data based on the day of the week, which is not a feature given to the architectures. The ESN was initially far less accurate than the unoptimized off-the-shelf architectures, but after tuning to achieve the proper reflection quality and switching to a trainable sigmoid activation classifier rather than a ridge regressor, similar accuracy was achieved. This meant the ESN was not only capable of the same accuracy as other models, but had the added benefit of far less training cost, due to only needing to train the weights to the sigmoid activation node. The selected experimental network data proved to be too early in production, and thus proper labeling and consistency across various IoT devices was not sufficient to train the architectures beyond always classifying negative. Further research on cleaned and usable network packet data may still provide interesting insights.

2. Related Work

The efficacy of reservoir computing systems for classification on multivariate time series data has been previously shown, and in our study we seek to both confirm this result through the simple bike traffic data and expand upon its validity for other complex cases, such as network packet capture classification (Bianchi et al. 2018). The architecture and construction of our ESN was informed by several papers, such as the original paper Jaeger 2003, a guide to design principles Lukosevicius 2012, and a guide to training Ferreira 2013, though they all have the same general architecture. Lukosevicius was the paper most closely followed, laying out the construction of the ESN as follows:

“1. generate a large random reservoir RNN ... ;

2. run it using the training input u(n) and collect the corresponding reservoir activation states x(n);

3. compute the linear readout weights Wout from the reservoir using linear regression, minimizing the MSE between y(n) and y target(n);

4. use the trained network on new input data u(n) computing y(n) by employing the trained output weights Wout.”

The paper further specifies numerous options for specific construction of the ESN, such as the following parameters and hyperparameters: the number of nodes within the reservoir, the density of connections between nodes, the spectral radius of the reservoir, the input scaling, and the leaking rate within the reservoir. The paper offers guidance on which parameters and hyperparameters are valuable for different applications, and in accordance with the paper and our testing we decided to focus on optimizing only the number of nodes, the spectral radius, and very importantly the method of distribution of the weights, as the other parameters were less impactful on our classifications.

In Schrauwen et al. 2007, the dimensionality reduction into fixed-sized representations quality of ESNs, aided by their recurrent temporal qualities, were shown to be very effective at classifying multivariate time series, which is especially significant due to their much faster training times than other commonly used methods. We decided to look into real example cases of complex time series data, and were motivated by the work presented in Holland 2020, which presented the nPrint package for turning network packet captures into machine learning pipeline usable binary, and tested the efficacy of several off-the-shelf architectures on example cases. The nPrint package has been made available, and so we sought to do further testing on both machine learning applications in the IoT space, and the efficacy of ESNs as a classifier on chaotic time series data. As a baseline, we decided to include similar off-the-shelf machine learning architectures, as is used in the original nPrint paper.

3. Methods

3.1. Data Collection & Preprocessing

Two data sets were selected for the testing of the architectures. Binary classifications were used as the truth labels, in order to achieve similarity across sets. The first set contains motion sensor data of bikes moving over Fremont Bridge in Seattle, with each row consisting of the count of travelers heading East, West, and total, per hour, respectively. This data is already split into three columns, and was imported directly into our pipeline using the csv Python package. The only required processing was to remove the date and time column from the data, so that only bike counts would be considered, as the intention with both sets is to be able to apply them to real time applications.

Labels were selected to be true from 1am to 5am, as this was the most uniquely identifiable timeframe within the data, which would make this labeling system as close to the network packet application as possible. The main cause for this is that other time frames, even the proposed early morning and rush hour increases, had a lack of distinction, thus not being suitable to this type of machine learning application.

The second selected dataset was the Iotlab_devices dataset. Iotlab_devices is a dataset collected by a network traffic acquisition pipeline integrated with the DeepLens project in the IoTLab within the John Crerar library at UChicago. The collected data has traces of network traffic (in .pcap and .csv) associated with human interactions with IoT devices in a lab setting. This data took much more processing, as it was in the raw network packet form, and so an outside software named nPrint was used instead (Holland, Jordan 2020). nPrint separated each packet into individual binary features, which could then be fed into a network after being imported and labeled using the csv library.

Due to the sparsity of the packet capture data, we devised a system to transform the data into regular intervals of data, providing a temporal context for the architectures. The timeframe of the data was divided into regular intervals, and any step without a packet was given a packet entirely of zeros, while any step with a packet remained unchanged. To ensure no packets were skipped, the timestep was smaller than the minimum time between packets. Labels for the data were not included, but rather were created based on supplied activation timings from the lab. We chose to label all timesteps during activations plus a two second buffer as true, as the IoT devices often had a brief delay before sending pertinent data, and little to no additional unrelated data was sent. This enabled our architectures to identify when a device was sending packets due to activation, rather than its normal procedure.

In both cases, the data was split into 80-10-10 randomized subsets, which were for training, testing of gridsearch models, and validation of the selected model, respectively. This enabled us to have well trained models while still avoiding using any data twice in testing and validation, so our selected model's graphs are based on entirely unseen data.

3.2. Model selection & Training

The ESN was constructed by hand, rather than an off-the-shelf implementation, and rather than a node-object approach, the underlying math was expedited by storing all the node values (R), input weights (Win), and internal connection weights (Wr) in matrices. R was a one dimensional matrix containing a value for each node, which were randomly set upon creation by a random uniform distribution, and changed dynamically as inputs entered the system (see Figure 1). Win was a matrix of nodes by features shape, so for every node, each feature had a unique weight chosen by a random uniform distribution. Wr was a matrix of nodes by nodes shape, which contained the weights of every node's value to each other node, and importantly was constructed with a normal distribution of scale one divided by the number of nodes.Wr also had its spectral radius tuned using a hyperparameter, which consisted of finding the absolute eigenvalue of the matrix and dividing the entire matrix by it, before finally multiplying that matrix by the chosen tuned spectral radius. This ensured no value expansion to infinity as time progressed, and allowed for tuning of how long values were stored in memory (see Figure 2). A hyperparameter grid search of the ESN resulted in an optimal tuned spectral radius of 0.1, and a node count of 50 for the bike data and 512 for the pcap data.

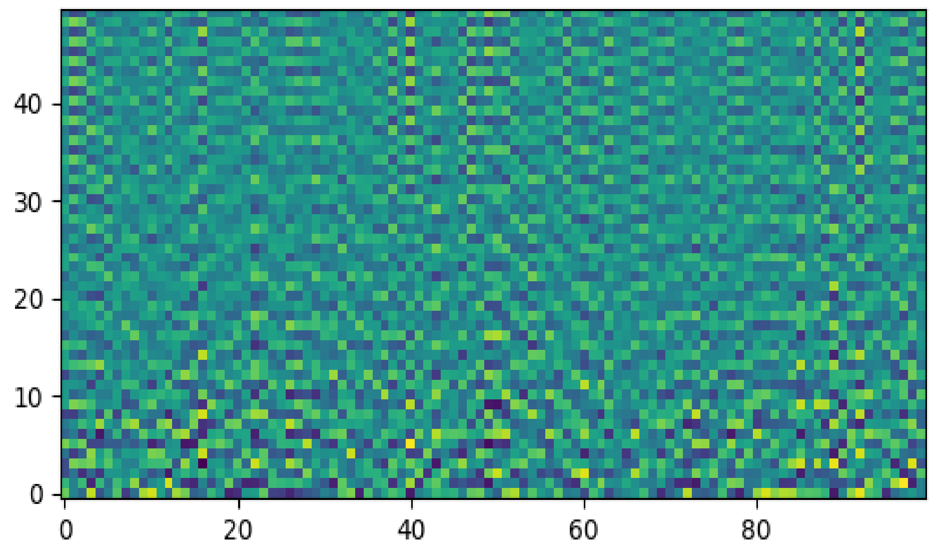

Figure 1: ESN activations for per node (X-axis) over time (Y-axis), Spectral Radius=1, Nodes=100

An initial input of all 1's was entered, then followed by 49 sets of zeroed features

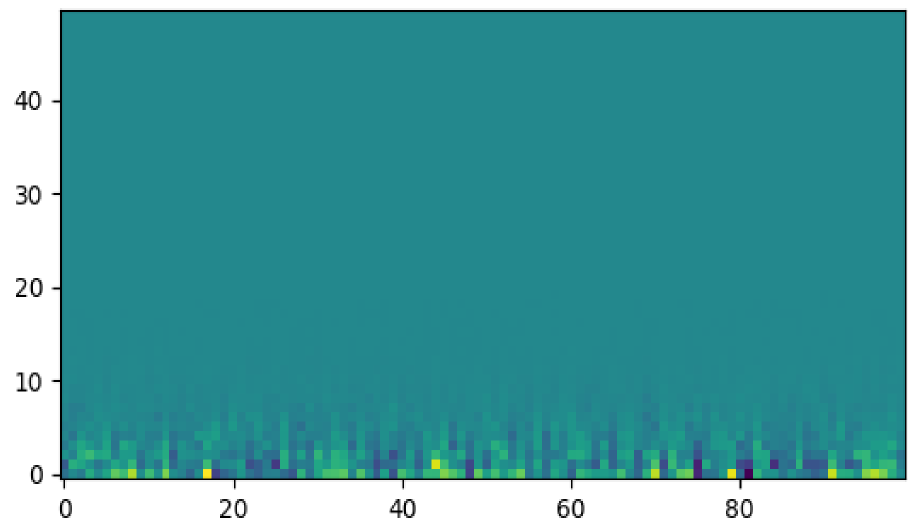

Figure 2: ESN activations per node (X-axis) over time (Y-axis), Spectral Radius=0.75

An initial input of all 1's was entered, then followed by 49 sets of zeroed features

Several additional off-the-shelf models from both Keras and Scikit-learn were used for comparison. From Scikit-learn, a Random Forest Classifier (RFC) and a Support Vector Machine (SVM) were used. The RFC was implemented with entirely default settings from the library, as the Random Forest expands to meet the size demands of the data. The SVM was implemented similarly, with an additional standard scaler from Scikit-learn which was used to make the features usable to the SVM, as it is negatively impacted by improper scaling of data, such as too large values. The following neural network architectures were implemented through Keras, with the following important hyperparameters, which were selected through several test runs across a grid search:

- Feed Forward (FF):
    - Two Dense Layers of (50/512) Nodes (Bike/pcap)
- 1D CNN:
    - Two Convolutional 1D Layers of (5/24) Kernels (Bike/pcap)
    - One Dense Layers of (50/512) Nodes (Bike/pcap)
- Long Short-term Memory (LSTM):
    - One LSTM Layer of (32/128) Nodes (Bike/pcap)
    - One LSTM Layer of (16/64) Nodes (Bike/pcap)

Each architecture contained a dropout layer before the output of value 0.1, a binary sigmoid classifier, and were trained using the adam optimizer, a binary cross-entropy loss function, and the metric of accuracy.

These specific architectures were selected because the FF created a baseline for the neural networks, and the 1D CNN and LSTM represented two very common methods for analyzing complex time series data, such as our network packet data. Each model was trained using 15 epochs in the final evaluations, as they had long plateaued by this point in prior testing.

3.3. Evaluation

Architectures were evaluated using precision-recall (PR) and receiver operating characteristic (ROC) graphs as well as confusion matrices (CM). The PR curve was used to visually summarize the actual positive rate versus the positive predictive value, which makes it particularly valuable in this case, as both data sets have largely imbalanced data, with a much smaller number of the binary classifications being positive. The ROC graph was used to graphically represent the true positive versus false positive rate. Finally, the CMs were used to easily count and compare the amount of each, which when combined with the graphs was of vital importance to evaluation, as these measured ensured the models were not simply achieving a high accuracy by always classifying negative, as due to the small number of positive labels, this would create a very high accuracy.

Results

Measurement	ESN	LSTM	1DCNN	FFNN	RFC	SVM
Accuracy on Bike Data (~83% False Labels)	93.32%	92.82%	93.26%	93.16%	92.95%	93.04%
Accuracy on IoT Data (~85% False Labels)	85.73%	86.02%	83.90%	82.91%	83.90%	84.11%

Table 1: Accuracies on validation set across architectures for each data set, with true/false rates of labels

The following plots depict the ROC curves for the ESN, 1D CNN, and RFC on subsets of the network data, creating a representative sampling across the architectures:

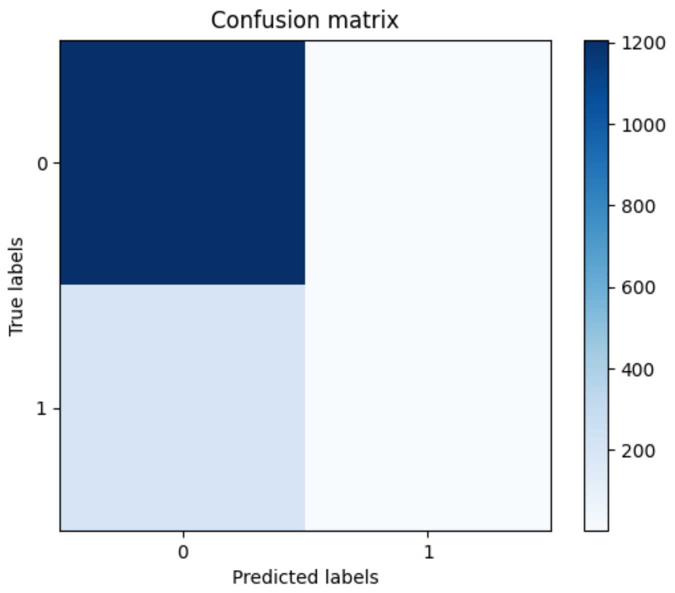

Figure 3: Confusion matrix of ESN on pcap data

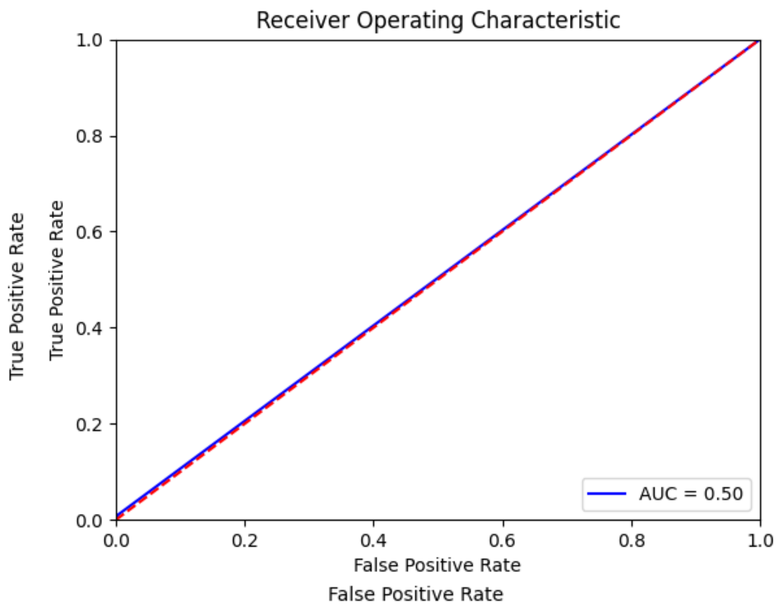

Figure 4: ROC curve of ESN on pcap data

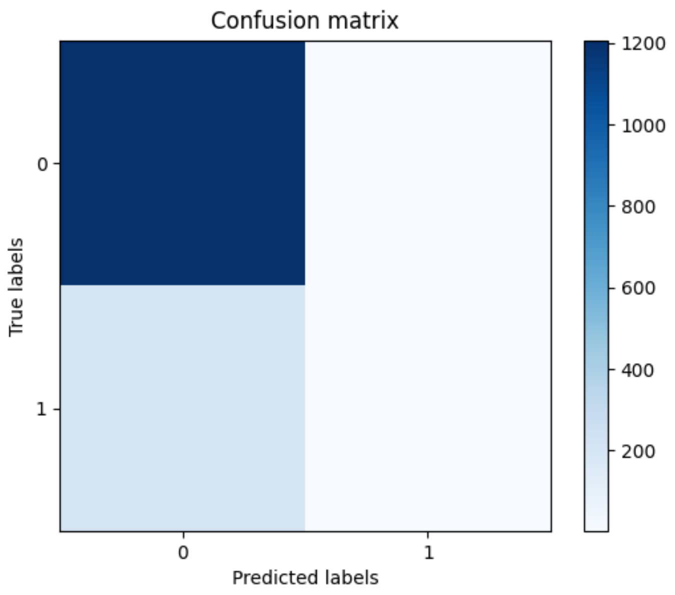

Figure 5: Confusion matrix of 1D CNN on pcap data

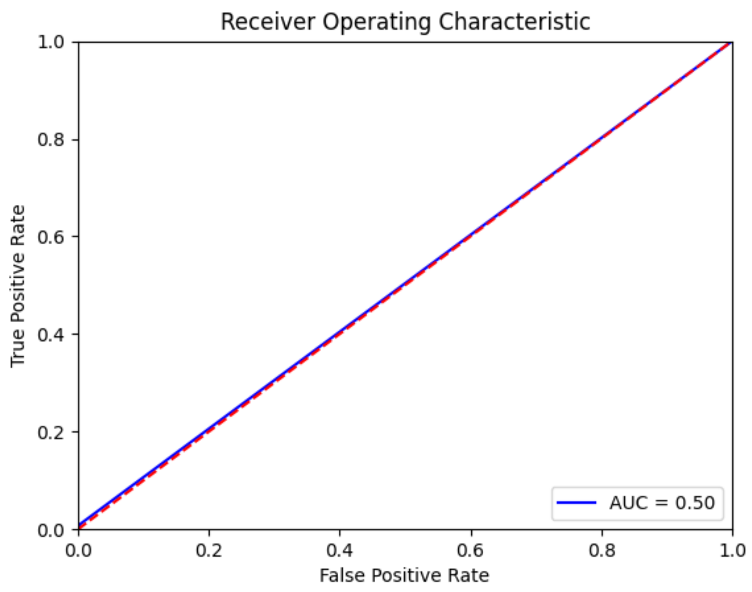

Figure 6: ROC curve of 1D CNN on pcap data

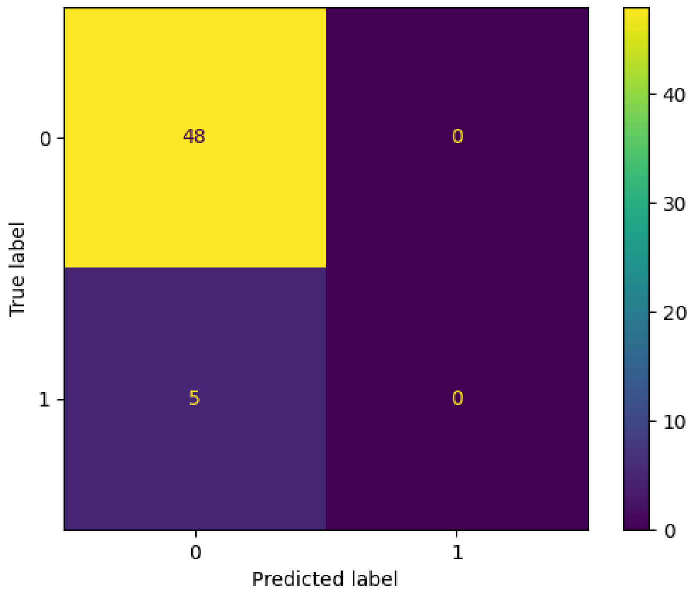

Figure 7: Confusion matrix of RFC on pcap data

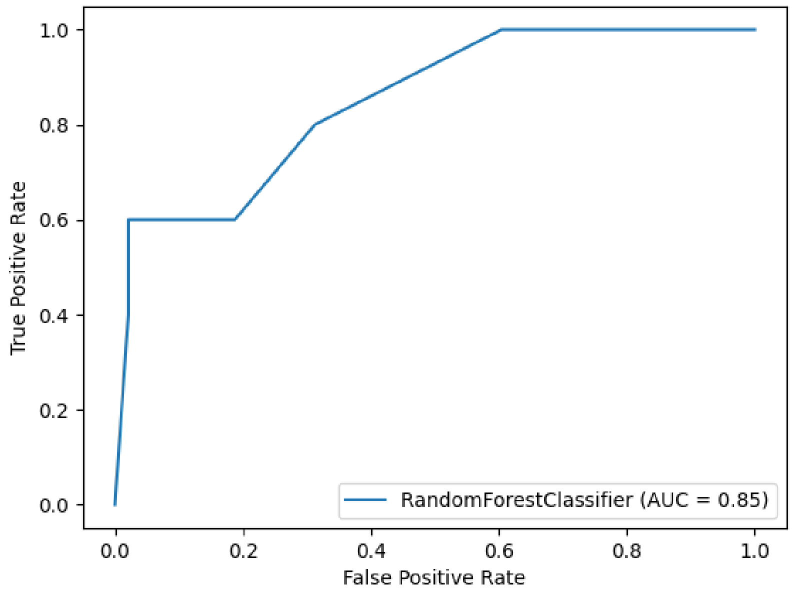

Figure 8: ROC curve of RFC on pcap data

The following plots depict the ROC curves, PR curves, and CMs for the ESN, 1D CNN, and RFC, on the bike data, creating a representative sampling across the architectures.

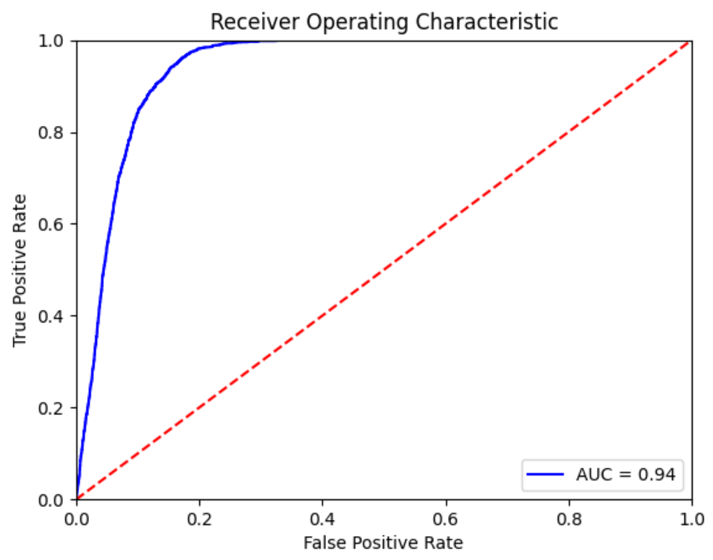

Figure 9: ROC Curve of ESN on bike data

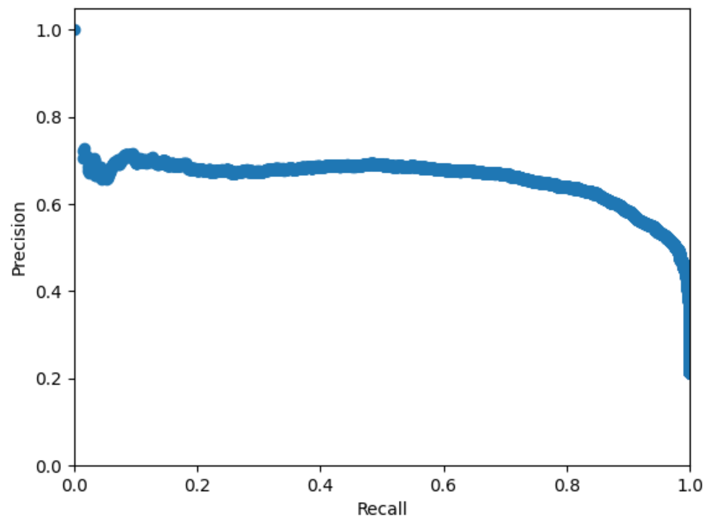

Figure 10: PR curve of ESN on bike data

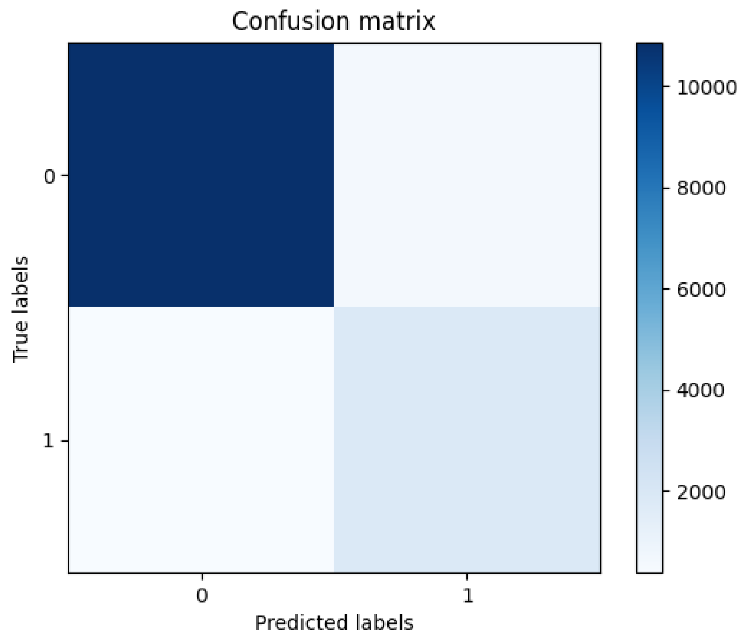

Figure 11: Confusion Matrix of ESN on bike data

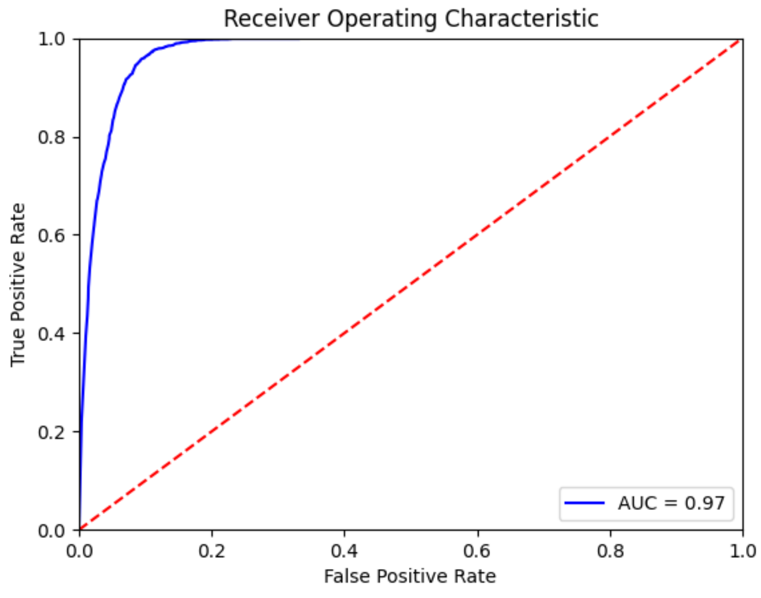

Figure 12: ROC Curve of 1D CNN on bike data

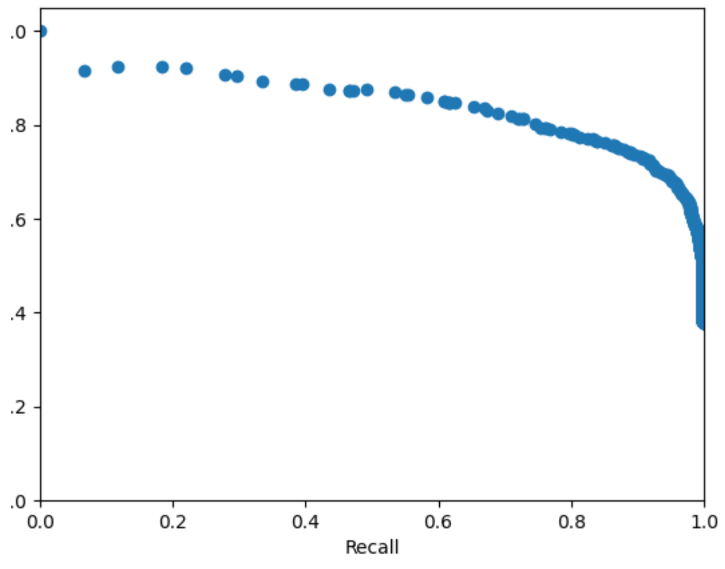

Figure 13: PR curve of 1D CNN on bike data

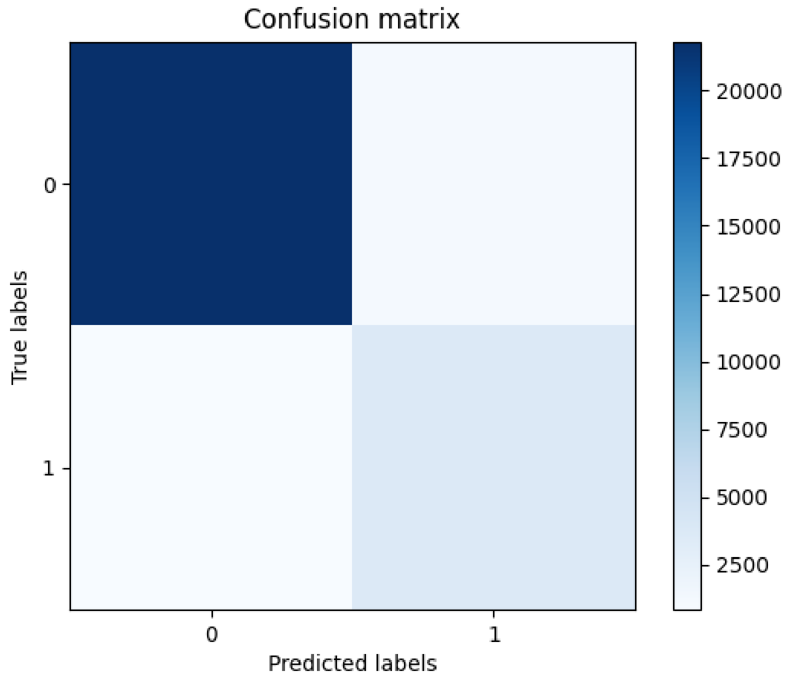

Figure 14: Confusion Matrix of 1D CNN on bike data

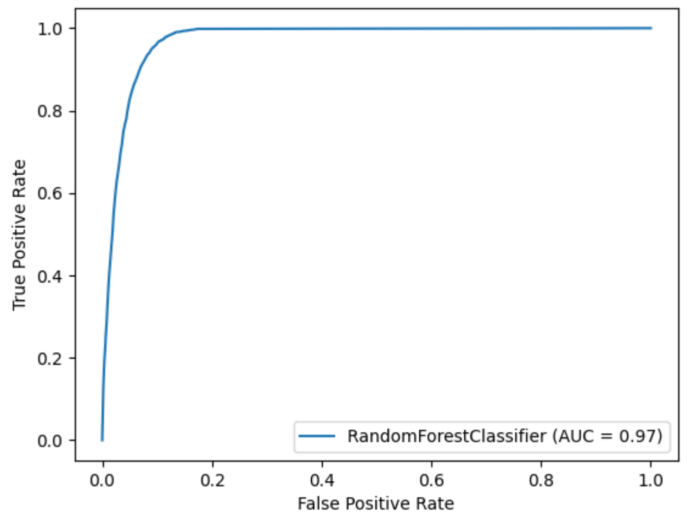

Figure 15: ROC Curve of ESN on bike data

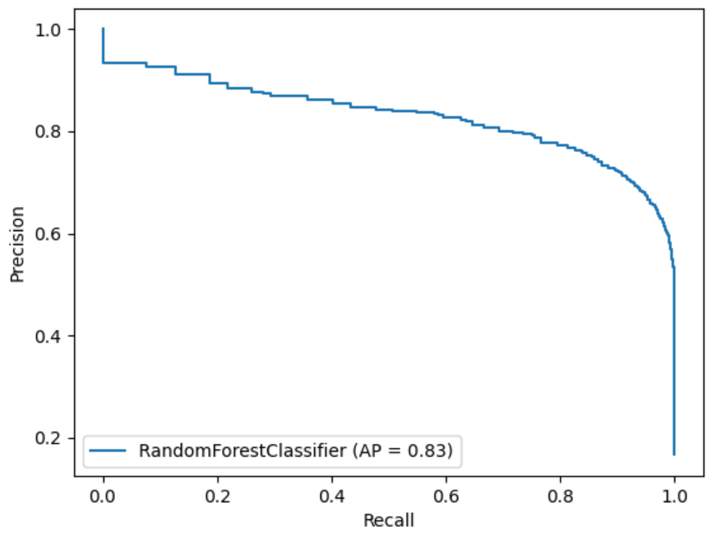

Figure 16: PR curve of ESN on bike data

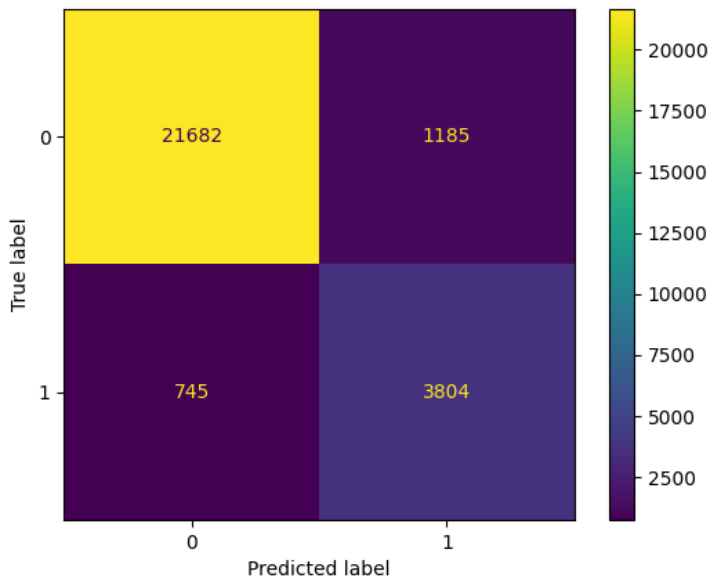

Figure 17: Confusion Matrix of ESN on bike data

The following plots depict the values within the nodes of the ESN while classifying the bike data, which is what is fed into the trainable sigmoid activation classifier. They are shown at the spectral radii 0.1 and 0.75, for a visualization of how the data impacts the reservoir at different values. This provides an explanation for why the reservoir performs better on the bike data at a lower spectral radius, as the immediate values are not hindered by additional noise.

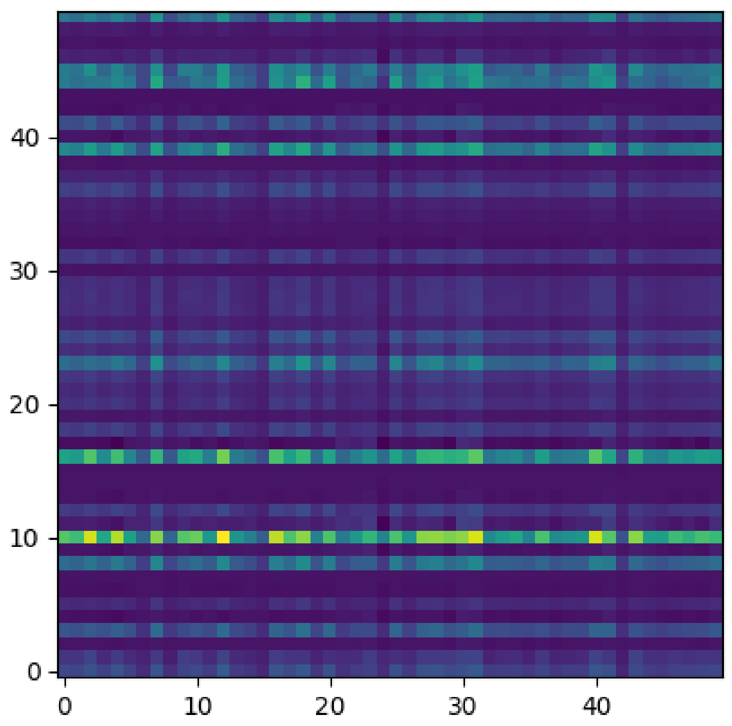

Figure 18: ESN activations per node (X-axis) over time (Y-axis), Spectral Radius=0.1

This the first 50 entries of bike data, thus spanning just over four days, including a weekend (bright spot)

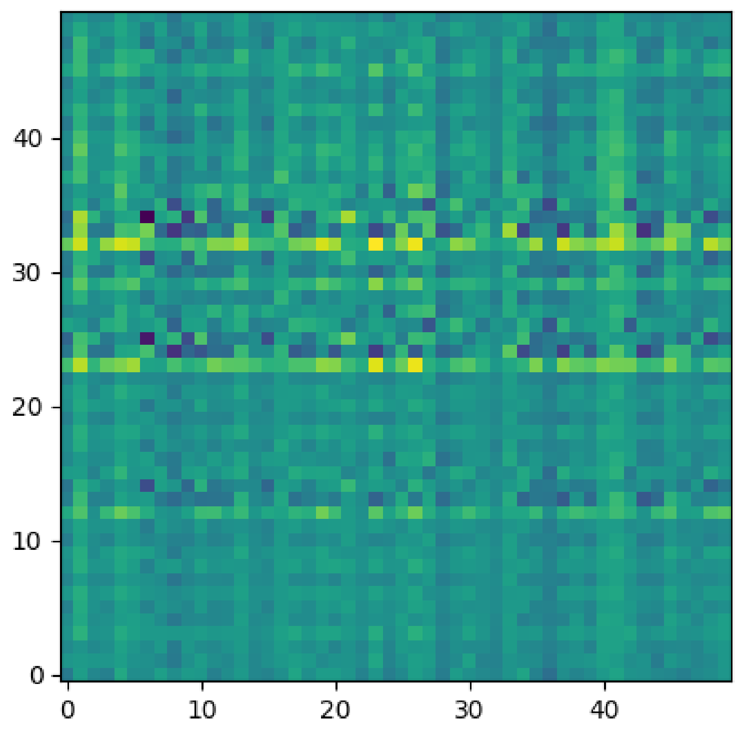

Figure 19: ESN activations per node (X-axis) over time (Y-axis), Spectral Radius=0.75

This is the same data, with only the spectral radius changed

Discussion/Future Work

Through our experiments, we discovered several important takeaways, the first of which was discovered during the construction and testing of our ESN. Initially, we were unable to achieve the reflections described within the reference papers, and dedicated significant testing time on the connection rates, output sparity, input scaling, and internal value scaling, such as using a tanh function on every reservoir value at every timestep. These changes proved inconsequential however, as the change that created the reflection patterns seen above in figures 18 & 19 was to alter the distribution of the weights between reservoir nodes to be a normal distribution with a scale inverse to the number of nodes. In the future, knowing the importance of the scale and distribution of the connections will allow for less time spent on achieving the desired reflection quality. The next takeaway was the impact of the spectral radius once we had a working ESN, and is displayed well above. With a low spectral radius, the data is only very temporarily impactful on the reservoir, which is good for situations like the bike data, in which each hour's classification should only be based on itself and maybe one or two hours before. With the network data however, a much higher spectral radius, tentatively around 0.75, would be important for creating overall spikes in the reservoir when many packets are sent in close succession, which would be important to distinguish between just a few packets sent close together in isolated moments, such as the machine's regular check-ins. This will need to be researched further however, as the network data was not in a state which we could readily use in our pipeline, due to inconsistencies across devices and difficulty finding accurate labelling. In the future, candidate data could be very thoroughly tested on one candidate model--such as an LSTM, which should perform well due to the known versatility on time series data--before being built around. We attempted testing, and had satisfactory initial results, but upon further analysis found the difficulties in achieving accuracy rates more than simply the true/false rate of the labels, and diagnosed a problem across architectures of the best method being an "always false" guess, as seen in the confusion matrices of figures 3, 5, & 7.

The bike data however was a far more successful set, and displayed the efficacy of the ESN on a novel case, in which some short term memory is beneficial. The ESN performed the best on the data of all of the architectures, and had the added benefit of needing to train only a single sigmoid activation classifier, with just one weight for each node's output, thus scaling linearly with the size of the reservoir. This met our expectation that the ESN would be able to perform well on time series data, and further testing is still needed to confirm if the ESN would function on IoT or network data in general. In that testing, it will be important to find a spectral radius that fits the data, as is highlighted in Lukosevicius 2012, and in our own findings that with too high of a spectral radius on the bike data, the accuracy of the ESN was negatively impacted. We still believe this is a very important direction to look into for the potential role ESN can play in the ever-growing machine learning space, as its ability to both simplify complex data and store variable memory scales through its reservoir may prove very useful for the erratic data of IoT devices and network traffic as a whole.

Bibliography

Schrauwen, Benjamin, David Verstraeten, and Jan Van Campenhout. "An overview of reservoir computing: theory, applications, and implementations." Proceedings of the European Symposium on Artificial Neural Networks ESANN 2007, pp. 471-482.

Jaeger, Herbert. “Adaptive Nonlinear System Identification with Echo State Networks.” International University Bremen 2003.

Ferreira, Aida A., Teresa B. Ludermir, and Ronaldo R.B. de Aquino. “An approach to reservoir computing design and training.” Expert Systems with Applications, Volume 40, Issue 10, 2013, Pages 4172-4182, ISSN 0957-4174.

Holland, Jordan and Schmitt, Paul and Feamster, Nick and Mittal, Prateek. nPrint: A Standard Data Representation for Network Traffic Analysis. arXiv preprint 2020

Bianchi, Filippo Maria, Simone Scardapane, Sigurd Løkse, and Robert Jenssen. “Reservoir computing approaches for representation and classification of multivariate time series.” IEEE Transactions on Neural Networks and Learning Systems 2018.

Lukosevicius, Mantas. “A Practical Guide to Applying Echo State Networks.” Springer 2012.