

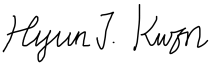
**J.N. Andrews Honors Program
Andrews University**

**HONS 497
Honors Thesis**

**Improving Ethanol Fermentation Estimation with Generative Data Augmentation in a Machine
Learning-driven Soft Sensor**

**Joseph Shiu
April 19, 2024**

**Advisor (Primary): Dr. Hyun Kwon
Advisor (Secondary): Dr. Roy Villafane**

Primary Advisor Signature: 
Department: School of Engineering

Abstract

While the use of Machine Learning through autoencoders and adversarial networks to augment data for computer vision (CV) applications is well known in industry and literature, its use in tabular time-series data for regression problems has only recently gained traction. This project aims to employ deep learning techniques to generate synthetic data. Given the expensive analytical processes to measure the ethanol concentration during fermentation, this data is used to augment the existing training data for a regression model which predicts the ethanol from variables easier to measure. Ultimately, we find that synthetic VAE data increases on average both the robustness and predictive power of a regression model to serve as a soft sensor to monitor the fermentation process.

Background

As industries seek to reduce their dependence on fossil fuels, biofuels are a prominent contender to provide for the globe's growing energy needs. The US Energy Information Administration [1] reports that total ethanol production in the United States has been increasing since 1981, and in 2021 its volume produced at 17.5 billion gallons per year. While ethanol is a greener alternative, it does not come without drawbacks. Ethanol is commonly mixed with gasoline at varying levels of concentration, but doing so reduces fuel efficiency, up to 27% less energy per gallon in E85 fuels [2]. In addition, ethanol production is subject to high cost and challenges associated with low or impure yields, motivating the development of various technologies to increase efficiency and reduce waste. Of particular interest in this model is the very high gravity (VHG) intensified fermentation technique, which yields higher ethanol concentrations compared to

conventional techniques. To effectively evaluate and monitor the fermentation process, it is necessary to have access to real-time information about the ethanol concentrations. Unfortunately, these measurements are costly and take hours to develop, so it is impractical to measure them directly.

Previous work by Kwon et al. [3] has demonstrated that it is possible to predict ethanol concentrations using auxiliary variables using tools from process analytical technology (PAT) that are far easier to measure. Specifically, electrical capacitance, redox potential, pH, and temperature were used as features in a feedforward neural network (FNN) regression model to prototype a soft sensor. After imputing missing data as well as corrupt data from improperly calibrated sensors, they were able to achieve satisfactory preliminary results ($R^2 = 0.97$). While data imputation is a standard practice in preparing data for the training step in machine learning, this may not be the best approach. In practice, sensor data is frequently noisy, and sensors may occasionally malfunction entirely. For a soft sensor to be resistant to such failure when predicting in real time, its underlying predictive model must be robust to outliers or corrupt data, and missing data. Because of the difficult nature of predicting the output variables of ethanol and substrate concentrations, a significant challenge in this modelling process is the scarcity of data. We hypothesize that synthetically increasing the dataset will help to alleviate these issues. This thesis is a continuation of Kwon's work and aims to leverage novel machine learning architectures with synthetic data generation to expand the size of the training set to ultimately improve the inferential power of the model. Specifically, we seek to discover whether generating synthetic data increases the robustness and predictive power of a

regression model for predicting ethanol concentrations.

Literature Review

The use of autoencoders and adversarial networks to augment data for computer vision (CV) applications is well known in industry and literature. However, its use in tabular data for regression problems has been studied in a growing number of contexts in the literature. Demir et al. [4] proposed the use of autoencoders (AEs), variational autoencoders (VAEs), and generative adversarial networks (GANs) to aid in expanding multivariate time series data on electricity demand forecasting for a regressive model. They found that while naïve approaches such as gaussian jittering and scaling produced little benefit, the AEs, VAEs, and GANs were able to reduce mean absolute error (MAE) by 2-3%. They noted that VAE and GAN architectures described in the text produced the best and most reliable results. Work by Papadopoulos and Karalis [5] demonstrated the use of VAEs to augment data from clinical studies in contexts where increasing the number of participants has high cost and ethical considerations. They reported that the VAE data improved the statistical power of their analyses, particularly noting utility in datasets with a high level of variance. Similar work has also been performed by a previous JN Andrews Scholar, Kim [6], who used VAEs on data for a biosensor. Kim's work produced mixed results, showing consistent improvements in the predictive model error when using VAE-generated data for one target variable, but a weaker, more erratic trend in the other target variable studied.

To our knowledge, however, these techniques have not been used on ethanol fermentation data, and it is the goal of this project to investigate their effectiveness.

Methodology

We aimed to implement these techniques on ethanol fermentation data and retrain predictive models to evaluate performance with synthetic data. Our methodology consisted of three primary stages. In the first stage, we trained generative models on the original data to generate synthetic data. In the second stage, we trained competing regression models with identical architectures, one on original data, one on synthetic data, and one on a 50/50 blend of original and synthetic data. In the final stage, we tested all three models on original data to evaluate their performance.

A. Generation Stage

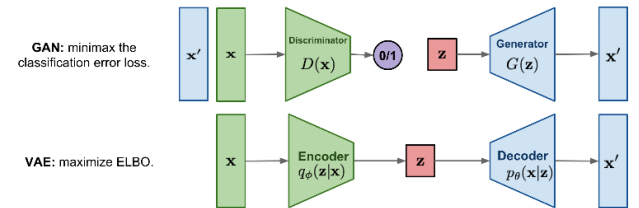


Figure 1: A schematic diagram of generative models [7].

We began with the original dataset, which consists of 11 individual experiments, each containing about 600 timesteps, spanning 10 hours. Because the sensors were considered noisy at the beginning, we chose to eliminate early samples, retaining the last 500 timesteps for each experiment. Each experiment contained several variables, but the ones we retained for this project were pH, redox, capacitance, and temperature, which comprise the 4 regression inputs, and ethanol concentration, the regression output. One experiment had a failed redox sensor, resulting in no data for that variable in the entire experiment, and this was imputed using a KNN imputation technique via the scikit-learn library. In addition, another experiment had a deviating capacitance sensor that was likely uncalibrated and gave readings that were consistently far from that of all the other

experiments. We will return to the discussion of this noisy capacitance data momentarily.

Using the open-source Time Series Generative Modelling (TSGM) library [8] built on the TensorFlow framework for Python, running in a Google Colab environment, we created gaussian jitter, VAE, and GAN models. Having trained each of these models, we generated 100 virtual experiments.

A VAE is a modification of the standard AE that allows for the generation of new data. An AE consists of two components, an encoder and decoder. The encoder compresses a high-dimensionality space into a latent space, while the decoder performs a reconstruction into the original vector space [9]. In a VAE, random sampling is introduced to the decoding process such that new rather than identical data are generated. In this case, points are randomly sampled from the latent space then fed to the decoder.

The encoder yields the posterior distribution $q(z|x)$, or the conditional probability of latent variable z given the evidence or real data x . In other words, the encoder produces a probable point in the latent space based on the real data. The decoder does the opposite, yielding the likelihood distribution $p(x|z)$, or the conditional probability of some data x given the latent variable z .

The objective function of the VAE is known as the Evidence Lower Bound (ELBO), which the training process seeks to maximize. The ELBO represents the lower bound of the log-likelihood function of the data. It consists of two terms. First is the reconstruction term, which is the log-likelihood of the reconstructed data at the decoder, expressed as an expectation. The second is the regularization term, which helps to prevent overfitting. It is given by the Kullback-Leibler (KL) divergence, an asymmetric measure of

distance between distributions. In this case, the KL divergence is taken between the posterior distribution $q(z|x)$ and the assumed prior distribution $p(z)$. The standard normal is typically used for the prior distribution.

The formula for ELBO is shown below [10]. The reconstruction term is the first term shown here. The KL divergence is the second term, and for the objective function it is subtracted since smaller KL divergence values are desirable.

$$\text{ELBO} = E[\log p(x|z)] - D_{KL}(q(z|x)||p(z))$$

Due to time constraints, we were not able to explore GANs to the extent that we did for VAEs. The basic structure of a GAN, however, consists of two neural networks: the generator and the discriminator. Each has a respective loss function that is minimized through backpropagation. The generator receives random input that it uses to generate a plausible sample, which is then passed to the discriminator. The discriminator consists of a binary classifier that determines whether the sample is real or fake, returning this feedback to the generator so that it can update its weights before generating a new sample. This process repeats until convergence, when the generator produces convincing synthetic data such that the discriminator is no longer able to distinguish between real and fake data [11].

B. Training Stage

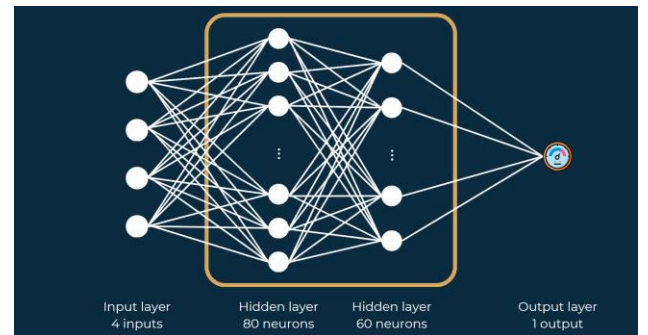


Figure 2: Regression model architecture.

We then developed an architecture for our regression model, opting for a simple feedforward neural network that was found to work well in preliminary testing (Fig. 2).

When designing machine learning models, datasets are typically divided into training and testing sets; however, this process is not as straightforward because our data consisted of 11 distinct time series. Because there is substantial variability between experiments, the model evaluation results were very sensitive to the way the data were split. To alleviate this problem, we settled on a repeated sampling method. In each iteration, we split the data by experiments, keeping entire time series grouped rather than mixing individual datapoints together. We then trained the model on the training set, then tested the models, recording the R^2 and the RMSE scores. This was repeated 100 times, and we then analyzed the resulting distribution of the scores.

To train the first of the three competing models, the original model, we employed the Leave Out One Cross Validation (LOOCV) technique. Of the 11 experiments, 10 were randomly selected to comprise the training set, then the one remaining experiment was designated the testing set. To train the synthetic model, the addition of a second pool of data offers many more degrees of freedom in choosing sets, so we used a bootstrapping method. 20 sets were chosen at random from the 100 sets of synthetic data to be the training set, and one set was chosen at random from the original data. This represents an approximate doubling in the training pool for the synthetic model. To train the blended model, we combined both approaches by randomly selecting 10 experiments from the synthetic set, randomly selecting another 10 experiments from the original set, and choosing the remaining

experiment from the original to be the testing set.

C. Testing Stage

At the completion of each iteration, we calculated the R^2 and the RMSE scores for the model based on the predictions generated for the one original experiment in the testing set.

Note that the R^2 score is the generalized coefficient of determination, which permits negative values. The interpretation is the ratio of variance explained by the regression model to the total variation in the data, where higher values are better up to a maximum of 1. Its formula [12] is given by:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2},$$

where y is the vector of true ethanol concentrations from the testing set, \hat{y} is the vector of predictions from the model, \bar{y} is the mean of the true ethanol values, and n is 500 timesteps. RMSE, or root mean squared error, is an aggregate measure of the errors between real and predicted values. Lowered values are better down to a minimum of 0. Its formula is given by:

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}.$$

We then identified changes in the distributions from the original to synthetic model, particularly noting the change in mean R^2 score as the key indicator of change in predictive power. Secondly, we used change in standard deviation of R^2 scores as the indicator of change in the robustness since a narrower spread of the distribution represents a more consistent model.

Results and Discussion

The generation of gaussian augmented data was the first viable data created. However, a visual observation of the time series graph (Fig. 3) of the ethanol concentration variable

quickly suggests that the jittering process does not respect the temporal nature of the data. Thus, we chose not to further pursue the gaussian technique and focused our attention on the more sophisticated deep learning techniques, beginning with the VAE.

During the training of the VAE, we noticed that the model did not seem to be converging correctly. Although the reconstruction loss decreased with the number of training epochs, the KL loss increased.

Fig. 4 shows the loss metrics for reconstruction and KL divergence for a training run of 250 epochs. Note that when interpreting loss, as the TSGM package reports, loss is to be minimized, as smaller values are desirable. The increase in the KL loss after about 125 epochs suggests that the model may be overfitting, but repeating the training with fewer epochs seems to exhibit the same behavior (Fig. 5). Although this convergence problem was of concern, we chose to proceed to the regression model training phase, ultimately using VAE data from a model run with 200 epochs and a batch size of 16.

For the GAN data, we used the TimeGAN [13] variant of the GAN that is designed for temporal data. However, looking at the data generated, it appears (Fig. 7) that it is not correctly accounting for the time series data and is producing extremely erratic results. It is not clear why this is the case. Wang and Liu [14] propose another, more sophisticated variant known as the Wasserstein GAN (WGAN); however, it was not implemented in the TSGM package. As it was infeasible to create it from scratch, we did not pursue this option.

Figure 6 displays the histograms for each variable for each of the data sources (original, jittered, VAE, GAN). Figure 7 displays the time series graphs in the same order, minus the jittered data. In the top row, for original data, there are 11 curves corresponding to each of the 11 experiments. In the lower two rows, for VAE and GAN data, there are 100 curves for the virtual experiments generated. We note that visually, the resemblance in distribution and temporality is strongest between the original data and the VAE data. We therefore chose to proceed with the VAE data exclusively into the regression model stage.

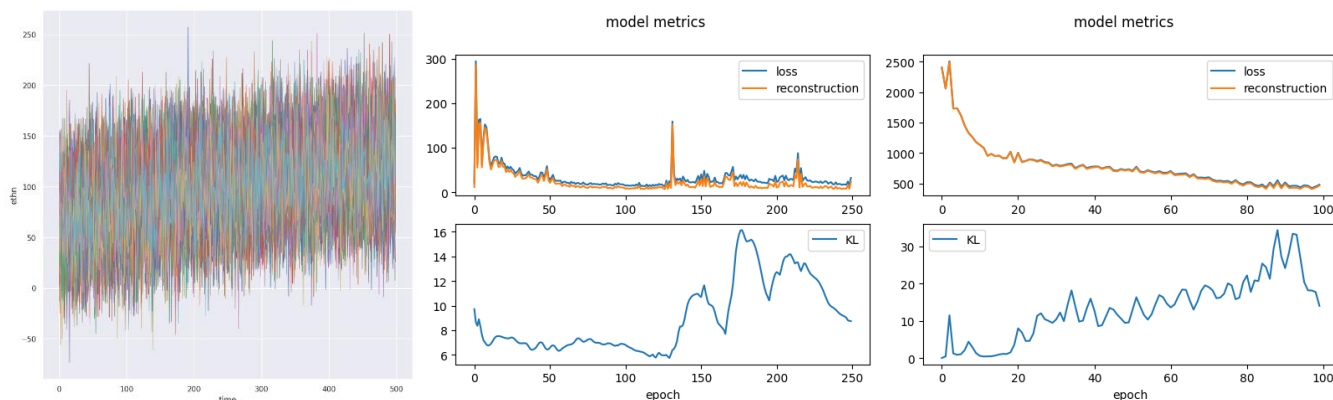


Figure 3 (left): Ethanol v. time graph for jittered data.

Figure 4 (center): VAE training metrics for 250 epochs.

Figure 5 (right): VAE training metrics for 100 epochs.

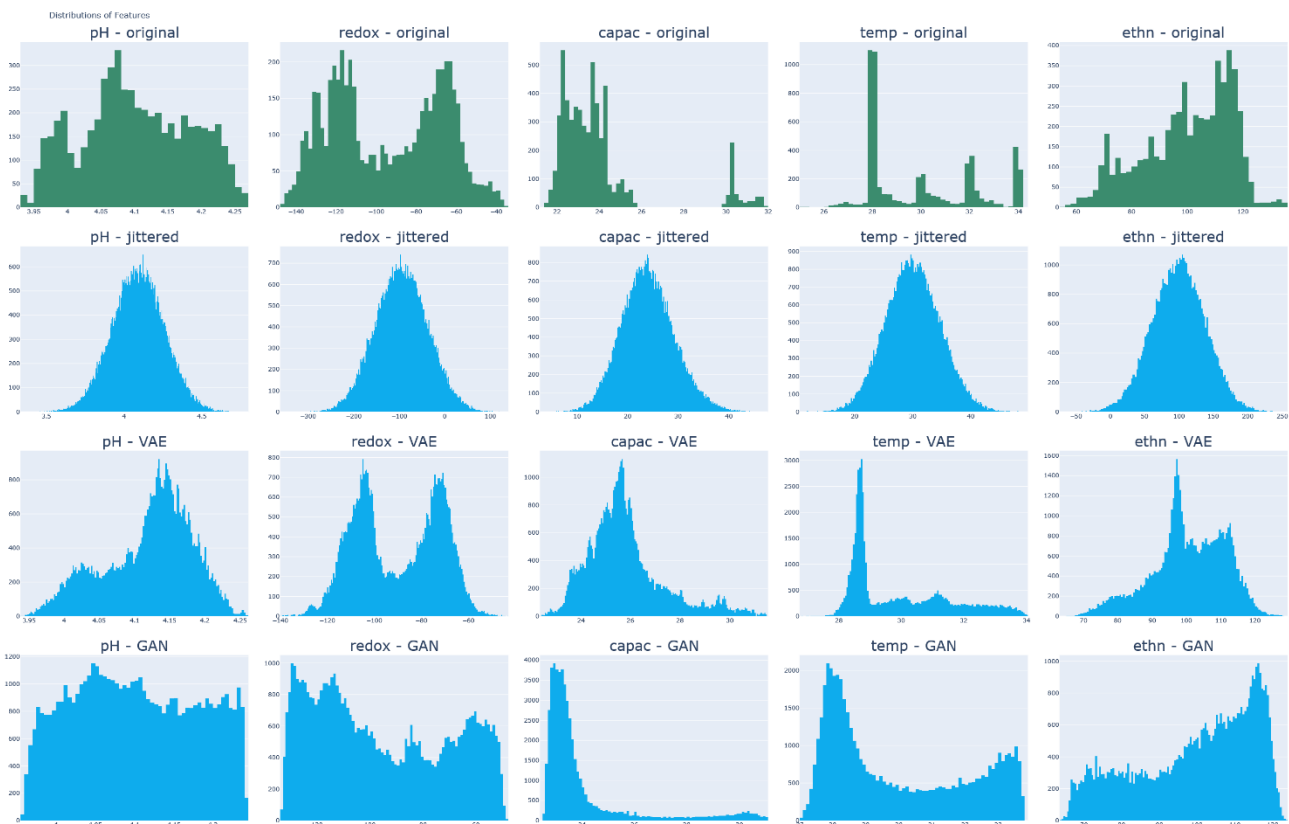


Figure 6: Histograms of features.

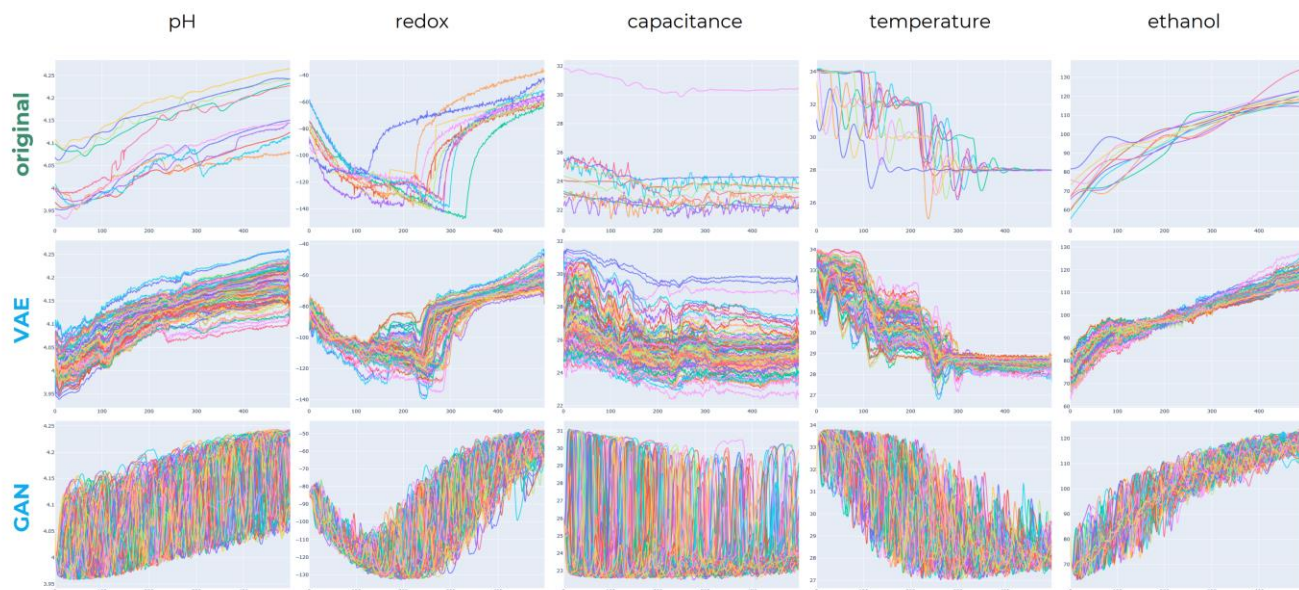


Figure 7: Time series graphs.

In the time series graphs for capacitance (Fig. 7), there is one deviating curve that we have previously mentioned. Although a logical and tempting approach would be to exclude this experiment, sensors in practice are frequently noisy. The motivation of the project is thus to create a soft sensor robust against sensors feeding bad data, which therefore requires the inclusion of the experiment in our analysis. Surprisingly, the VAE handles the capacitance variable by creating time series that fill in the spaces between the cluster of 10 experiments and the 1 outlier experiment. This appears to be consistent with the t-SNE graphs with synthetic data superimposed on original data (Fig. 8).

The t-distributed stochastic neighbor embedding (t-SNE) is a technique proposed by van der Maaten and Hinton [15] to visualize high-dimensional data in a smaller space easily comprehended by the human mind, typically two dimensions. Although it is an abstraction with subjective interpretation where neither the points nor distances between them necessarily correspond to real-world meanings, it is nonetheless a useful indicator of data similarity.

In the t-SNE graph for VAE (Fig. 8), the original data (denoted in green triangles) form an approximate boundary zone within which the synthetic data (denoted in blue hexagons) populate the spaces. Although we would hope that the synthetic data cluster near the same positions as the original, the VAE graph is more optimistic than the GAN graph (Fig. 9), where the synthetic points seemingly do not respect the areas demarcated by original points at all.

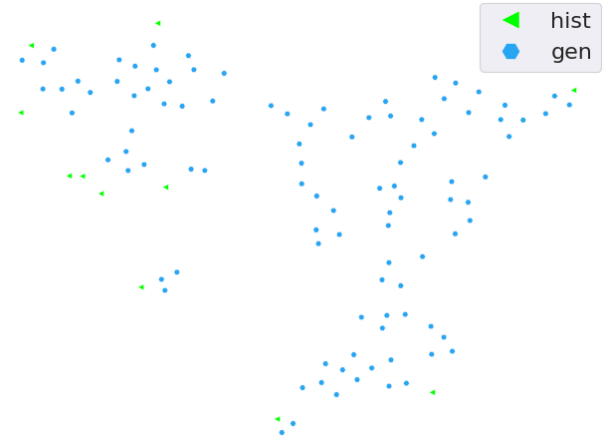


Figure 8: t-SNE graph for VAE and original data.

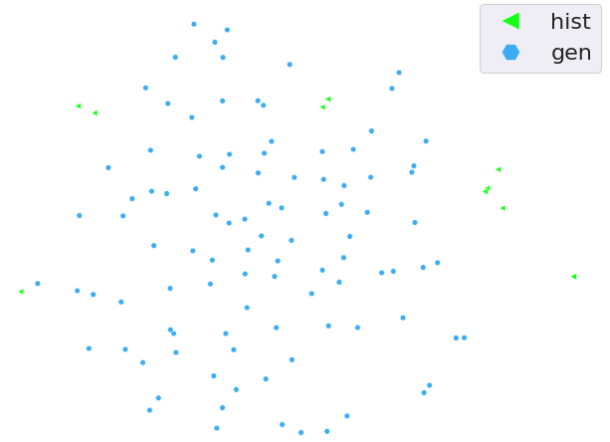


Figure 9: t-SNE graph for GAN and original data.

We now proceed to the discussion of the regression model. Listed below are the statistics for the distribution of R^2 scores (Table 1) and for RMSE scores (Table 2). The $\% \Delta$ columns display the change in the specified statistic using the original model as a baseline. For the Δ mean columns, we calculate the p -value using Welch's unequal variance t-test of means. For the Δ standard deviation columns, we calculate the p -value using the Levene test of equal variance. In addition, Figure 10 shows the histograms for R^2 scores, and Figure 11 show the histograms for RMSE scores.

Distribution of R^2 scores

| MODEL | MEAN | % Δ MEAN (P -VALUE) | STD. DEV. | % Δ STD. DEV. (P -VALUE) |
|-----------|--------|-------------------------------|-----------|------------------------------------|
| ORIGINAL | 0.6367 | – | 0.7367 | – |
| SYNTHETIC | 0.8059 | +26.59% ($p = 0.026$)* | 0.1513 | –79.46% ($p = 0.040$)* |
| BLEND | 0.7872 | +23.64% ($p = 0.047$)* | 0.1363 | –81.50% ($p = 0.027$)* |

Table 1: Distribution of R^2 scores. Higher R^2 scores are better, max 1. Lower standard deviation scores are better, min 0.

Distribution of RMSE scores

| MODEL | MEAN | % Δ MEAN (P -VALUE) | STD. DEV. | % Δ STD. DEV. (P -VALUE) |
|-----------|---------|-------------------------------|-----------|------------------------------------|
| ORIGINAL | 0.09219 | – | 0.06449 | – |
| SYNTHETIC | 0.07891 | –14.40% ($p = 0.064$) | 0.02985 | –53.70% ($p = 0.129$) |
| BLEND | 0.08612 | –6.58% ($p = 0.388$) | 0.02747 | –57.40% ($p = 0.067$) |

Table 2: Distribution of RMSE scores. Lower RMSE scores are better, min 0. Lower standard deviation scores are better, min 0.

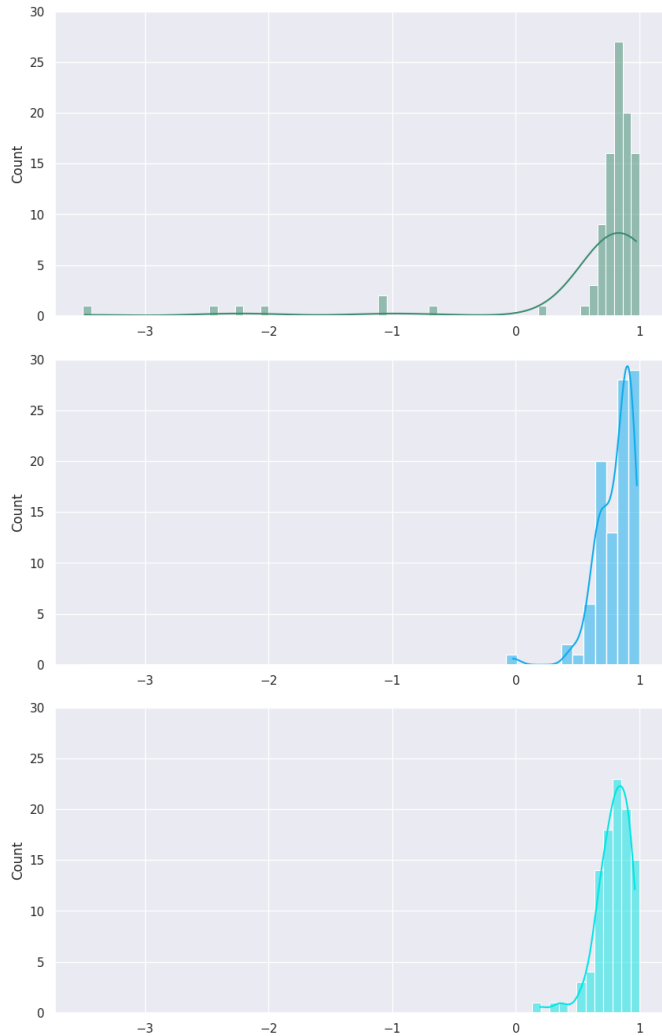


Figure 10: Histograms for R^2 .
For both figures, original (top), synthetic (center), blend (bottom).

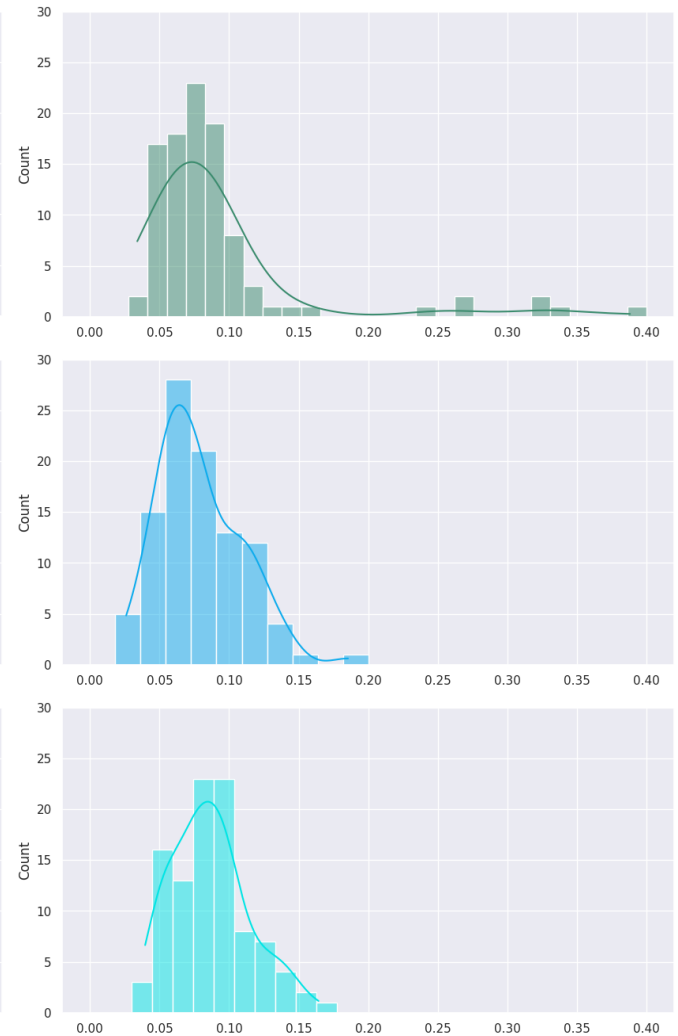


Figure 11: Histograms for RMSE.

Because the interpretation of R^2 scores is straightforward, we can immediately observe that the mean R^2 score of 0.64 for the original model is poor, although this value is skewed downward by the severe outliers visible in the histogram. Ultimately, this results in a high spread of the distribution, rendering the concern of inconsistent model performance. The introduction of synthetic data increases the R^2 score, with the full synthetic model offering the greatest increase in predictive power, a 27% increase to 0.81. The blend model trails slightly at 0.79. We note, however, that R^2 values in the 0.8 range are still lackluster despite offering a substantial improvement. We observe the most dramatic improvements in standard deviation, particularly with the blended model. It decreased nearly 82%, from 0.74 to 0.14. The synthetic model trails slightly at 0.15. We can also confirm this visually from the histograms of the synthetic and blend models, with R^2 values clustering closer to 1 without the presence of extreme outliers.

While all the changes in R^2 values were statistically significant at the 0.05 level, the changes in RMSE were not as striking, nor did

they meet the threshold for significance. Nonetheless, we observed slight improvements, with the synthetic model leading in the reduction of mean RMSE by 14%, and the blend model leading in reduction of standard deviation by 57%. It is of note that for both metrics, the synthetic model yielded stronger improvements in the mean values, while the blend model yielded stronger improvements in consistency.

Finally, to illustrate the practical usage of the soft sensor, we include several time series graphs for individual experiments, plotting the ethanol concentration v. time curves from the model prediction superimposed on the real data. Experiment 8 (Fig. 12) is the original experiment with noisy capacitance data. Experiment 3 (Fig. 13) and Experiment 0 (Fig. 14) are arbitrary selections from the original set. In Figures 12, 13, and 14, the top left represents the model predictions from original model, the top right represents the synthetic model trained on 20 randomly selected experiments, the bottom left represents the synthetic model trained on the entirety of the synthetic data, and the bottom right represents the blended model.

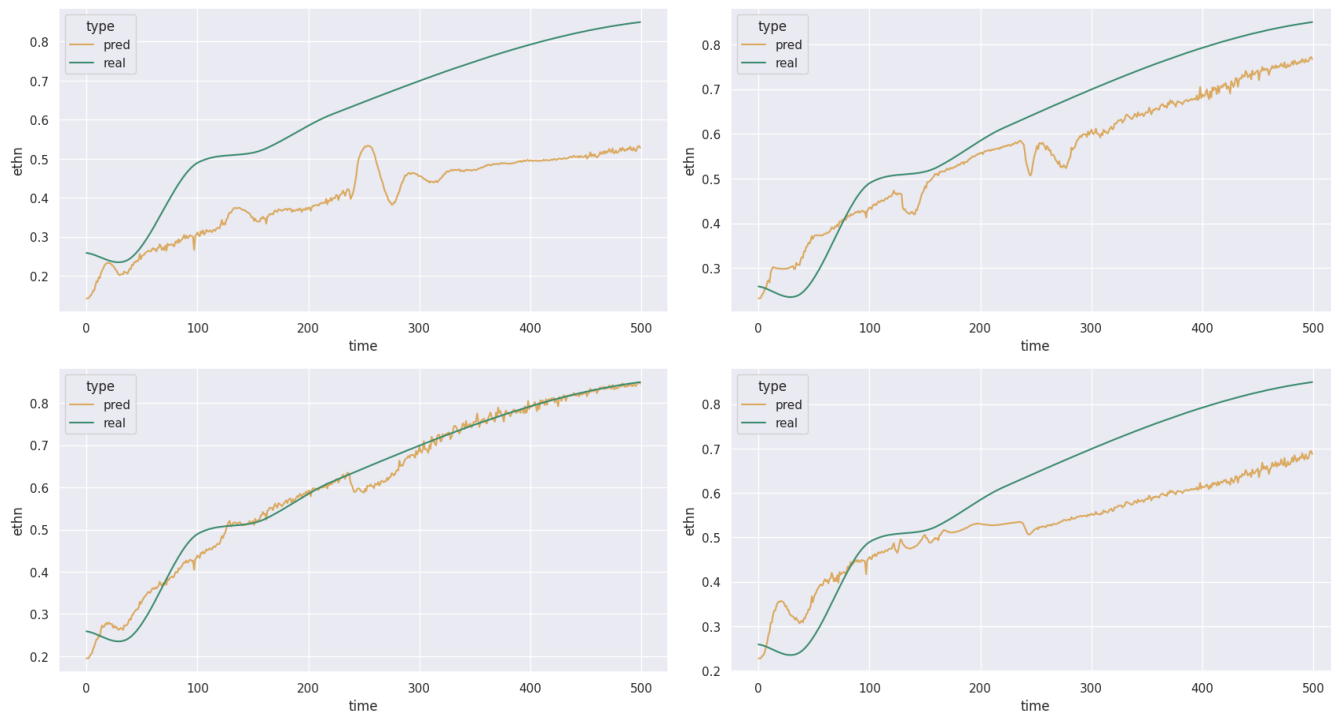


Figure 12: Experiment 8

(top left): original model. $R^2 = -0.56$.

(bottom left): synthetic model (all sets). $R^2 = 0.98$.

(top right): synthetic model (20 sets). $R^2 = 0.81$.

(bottom right): blend model. $R^2 = 0.54$.

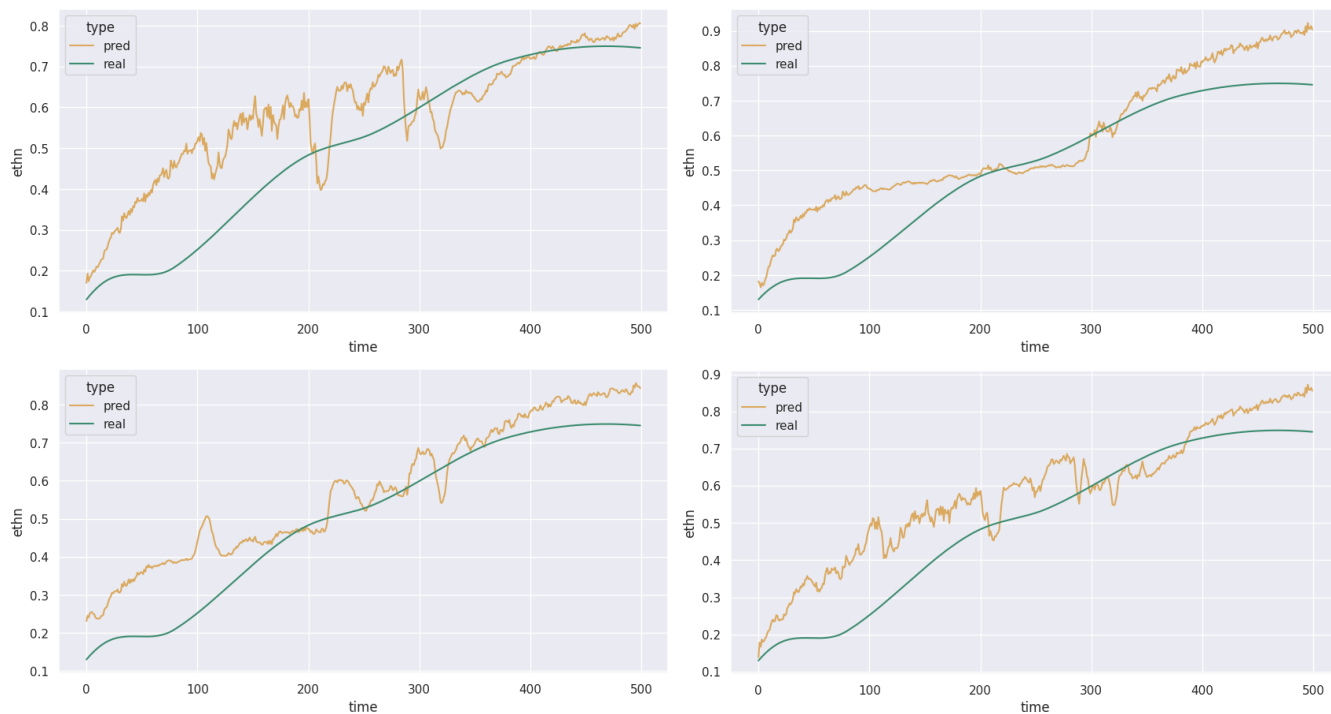


Figure 13: Experiment 3

(top left): original model. $R^2 = 0.61$.

(bottom left): synthetic model (all sets). $R^2 = 0.80$.

(top right): synthetic model (20 sets). $R^2 = 0.70$.

(bottom right): blend model. $R^2 = 0.74$.

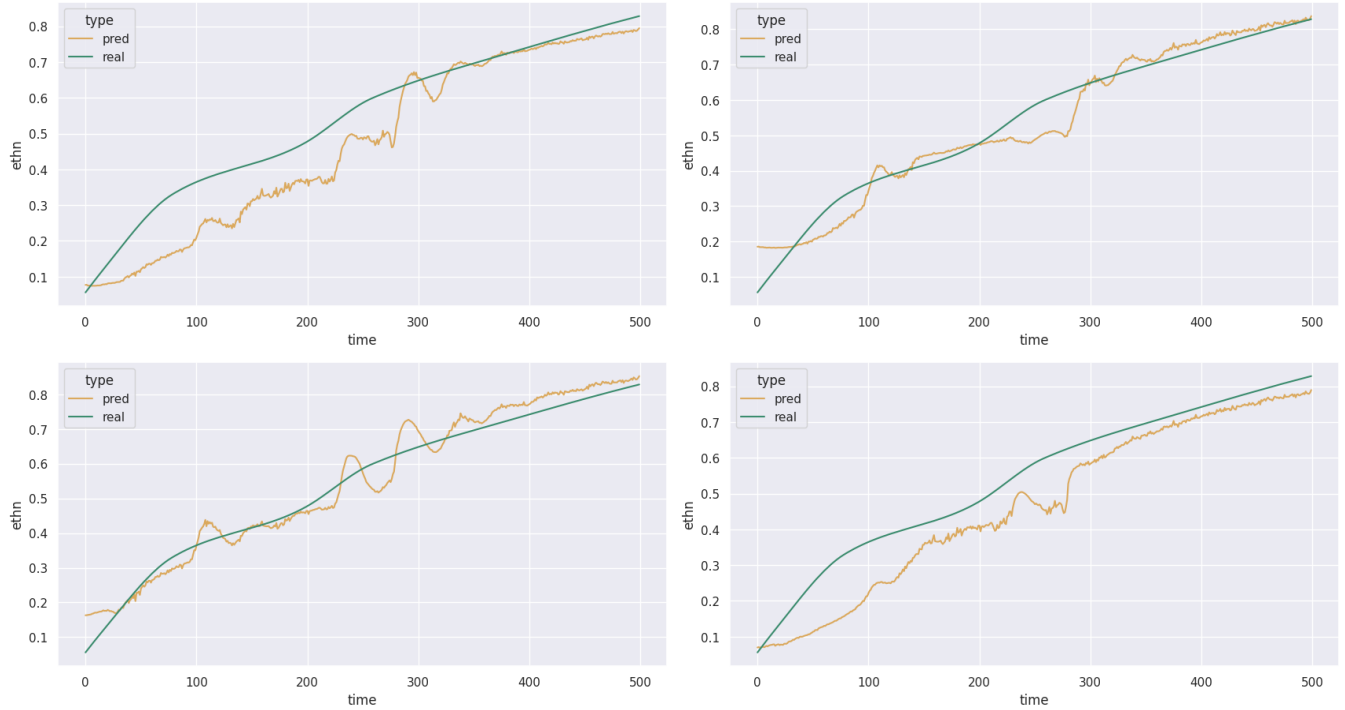


Figure 14: Experiment 0

(top left): original model. $R^2 = 0.80$.

(bottom left): synthetic model (all sets). $R^2 = 0.96$.

(top right): synthetic model (20 sets). $R^2 = 0.95$.

(bottom right): blend model. $R^2 = 0.81$.

These time series graphs slightly dampen the success observed from merely comparing the distributions of repeated sampling. In all cases, the addition of synthetic data in some manner improved the R^2 scores, although to widely varying degrees. In Figure 12, both the synthetic 20-set model and the blend model substantially improved the predictive power but remained mediocre, requiring a push to the synthetic 100-set model to achieve a usable model. This is likely because the deviating capacitance data were incorporated into the VAE training, which propagated into the VAE synthetic data, but did not appear in a sufficiently critical mass when randomly sampling from the synthetic data. Thus, without being trained on a much larger set of data, the model is unable to correctly account for deviating inputs.

Figure 13 seems to exhibit the opposite problem. This experiment does not have a deviating sensor, but the best model still

severely underperforms at 0.80. This suggests that the influence of the deviating sensor in the original data and propagated into the synthetic data has corrupted the model training, such that it expects to see deviated values rather than true values. Thus, it does not properly predict a normal experiment. Figure 14 is the most optimistic of the time series. Both the 20-set and 100-set synthetic models perform very well, although for some reason the blend model is biased downward.

Limitations

We advise caution in the interpretation of the results in Tables 1 and 2, as they may represent an overly optimistic perspective of the problem. While in general there are vast improvements in the predictive power and robustness of the regression model from synthetic training data, this does not guarantee good results for any arbitrary case.

We raise an important question: would we observe similar improvement in the regression model if the VAE was only permitted to be trained on a partial set of the original data? The results from the individual time curves suggest that the generation of synthetic data amplifies rather than resolves the problem of deviating sensors. We thus propose future work of extending the random splitting of data to the data generation phase such that it can be observed how well the regression model generalizes to data that neither it nor the model that generated its synthetic training data has ever seen.

In addition, the optimization of both generative and regression models was beyond the scope of this work. Future work may involve investigating the reasons for the poor GAN performance. Can it be improved by modifying hyperparameters? Can the use of more modern architectures such as WGAN resolve these problems? In addition, the failure of convergence for the VAE remains a mystery. Perhaps the tuning of VAE hyperparameters may also be of benefit. Finally, the use of a FNN may not be the best choice for temporal data. Can more sophisticated architectures, such as recurrent neural networks (RNNs) that are more applicable for time series be developed to replace the FNN?

Conclusion

In this work we have demonstrated the general utility of synthetic VAE data for the improvement of a regression model, with increases in predictive power as high as 27% and decreases in variability as high as 82%, as measured by the mean and standard deviation of the R^2 scores. However, it is important to recognize that these figures represent an average case from a distribution of repeated sampling, which does not

guarantee a well-functioning soft sensor in all situations. Further research is necessary to optimize the generative and regressive models before an industry usable soft sensor can be developed.

Acknowledgements

I would like to thank Dr. Hyun Kwon, Dr. Roy Villafane, the JN Andrews Honors Program, and the Office of Research and Creative Scholarship for their support of this project.

Code Samples

The Colab notebooks used in this project will be made open-source and available at: <https://github.com/joeyio/ethanol-sensor-public>.

References

- [1] US Energy Information Administration. "Biofuels explained." <https://www.eia.gov/energyexplained/biofuels/ethanol-supply.php> (accessed Apr. 15, 2024).
- [2] US Department of Energy. "Ethanol Benefits and Considerations." <https://afdc.energy.gov/fuels/ethanol-benefits> (accessed Apr. 15, 2024).
- [3] H. J. Kwon, E.C. Rivera, C.K. Yamakawa, C.E.V. Rossell, and J. Nolasco. "Prediction of intensified ethanol fermentation of sugarcane using a deep learning soft sensor and process analytical technology," submitted for publication, 2023.
- [4] S. Demir, K. Mincev, K. Kok, N.G. Paterakis. "Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting." *Applied Energy*, vol. 304, no. 117695, Dec. 2021, doi: [10.1016/j.apenergy.2021.117695](https://doi.org/10.1016/j.apenergy.2021.117695).

- [5] D. Papadopoulos and V.D. Karalis. "Variational Autoencoders for Data Augmentation in Clinical Studies." *Applied Sciences*, vol. 13, no. 15, July 2023, doi: [10.3390/app13158793](https://doi.org/10.3390/app13158793).
- [6] S. Kim. "Variable Autoencoders for Biosensor Data Augmentation." *Andrews University Honors Theses*, no. 252, Apr. 2021, doi: [10.32597/honors/252](https://doi.org/10.32597/honors/252).
- [7] L. Weng. "Flow-based Deep Generative Models." Lil'Log. <https://lilianweng.github.io/posts/2018-10-13-flow-models/> (accessed Apr. 15, 2024).
- [8] A. Nikitin, L. Iannucci, and S. Kaski. "TSGM: A Flexible Framework for Generative Modeling of Synthetic Time Series." 2023, [arXiv:2305.11567](https://arxiv.org/abs/2305.11567).
- [9] J. Rocca. "Understanding Variational Autoencoders (VAEs)." Toward Data Science. <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73> (accessed Apr. 15, 2024).
- [10] S.G. Odaibo. "Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function." 2019, [arXiv:1907.08956v1](https://arxiv.org/abs/1907.08956v1).
- [11] Google Developers. "Overview of GAN Structure." <https://developers.google.com/machine-learning/gan/gan-structure> (accessed Apr. 15, 2024).
- [12] Scikit-learn Developers. "Metrics and scoring: quantifying the quality of predictions." Scikit-learn Documentation, https://scikit-learn.org/stable/modules/model_evaluation.html (accessed Apr. 15, 2024).
- [13] J. Yoon, D. Jarrett, and M. van der Schaar. "Time-series Generative Adversarial Networks." NeurIPS 2019, https://papers.nips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.
- [14] X. Wang and H. Liu. "Data supplement for a soft sensor using a new generative model based on a variational autoencoder and Wasserstein GAN." *Journal of Process Control*, vol. 85, pp. 91-99, Jan. 2020, doi: [10.1016/j.jprocont.2019.11.004](https://doi.org/10.1016/j.jprocont.2019.11.004).
- [15] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE." *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008, <https://jmlr.org/papers/v9/vandermaaten08a.html>.