A Surrogate Tiny Machine Learning Model of Variational Autoencoder For Real-Time Baseline Correction of Magnetometer Data

Talha Siddique
Department of Electrical and Computer Engineering
University of New Hampshire
Durham, NH, USA
talha.siddique@unh.edu

MD Shaad Mahmud
Department of Electrical and Computer Engineering
University of New Hampshire
Durham, NH, USA
mdshaad.mahmud@unh.edu

***Abstract*—Magnetometers play a vital role in geophysics and space weather prediction applications by collecting terrestrial magnetic field data. They monitor solar-induced geomagnetic disturbances, providing essential insights into predicting space weather effects on technologies such as satellites, power grids, and communication networks. However, this data often contains inherent background noise, necessitating accurate baseline correction methods. Traditional correction techniques are robust but computationally demanding and unsuitable for real-time applications. Recent progress has investigated the utilization of Tiny Machine Learning (TinyML) to process magnetometer data in real time, especially when resources are limited. However, these edge-based ML solutions often lack the robustness of more computationally intensive probabilistic models, such as Variational Autoencoders (VAEs). This paper introduces a TinyML-VAE surrogate model designed for real-time magnetometer baseline correction. The surrogate model approximates an implemented VAE's performance while operating within the constrained resources of an edge device. The new model retains the VAE's uncertainty quantification capabilities by leveraging surrogate modeling techniques, ensuring robustness. Experimental outcomes have been displayed, illustrating a comparison between the performance of the TinyML-VAE and the benchmark established by the standard VAE.**

***Index Terms*—Edge Machine Learning, TinyML, Variational Autoencoder, Data Denoising, Geomagnetic Data**

I. Introduction

Magnetometers are pivotal in many domains, ranging from geophysics to space weather forecasting [1]. The terrestrial magnetic field data collected by such instruments suffers from background noise. Therefore, accurate and timely baseline correction is a prerequisite for the validity of scientific analyses [2] [3]. Traditional baseline correction methods, while robust, often necessitate human intervention and are computationally intensive [2]. Consequently, they fail to meet the requirements of real-time applications [3].

Recently, initiatives have been undertaken to develop a machine learning (ML)-enabled magnetometer system for real-time baseline correction and prediction of ground magnetic perturbations [3]. The work leverages a subset of edge ML called Tiny Machine Learning (TinyML). The TinyML paradigm deals with developing and deploying ML models for resource-constrained edge devices like microcontroller units (MCU) [3] [4]. Preliminary results on the ML-enabled magnetometer have shown promise for real-time, low-power, and efficient data processing. However, unlike offline or cloud-based ML deployments, the TinyML framework cannot accommodate computationally expensive models due to the target hardware's resource limitations [4]. For example, in the past literature, cloud-based methods like Variational Autoencoders (VAEs) have become preferred for robust data denoising [5]. VAEs, a probabilistic counterpart of Autoencoders (AE), consist of two sub-modules, one representing an encoder and the other a decoder [6]. VAE utilized the encoder to map the raw input data into a posterior probability distribution over a lower-dimension space. The decoder takes samples from the encoder-generated distribution to produce a prediction probability distribution. The predicted distribution denotes the denoised data constructed from its raw counterparts. Given the VAE approach's probabilistic nature, the dispersion of the distribution denotes the prediction uncertainty, thereby ensuring robustness [7] . Nevertheless, the computational resources required for implementing such probabilistic models exceed what is available in most MCUs, and therefore, the TinyML framework tends to lack robustness [8].

This paper aims to bridge the aforementioned gaps through the creation of a TinyML implementation, functioning as a stand-in for the conventional denoising Variational Autoencoder (TinyML-VAE surrogate). This method effectively executes real-time baseline correction for magnetometer data, utilizing the surrogate model concept within machine learning. A surrogate model, also known as an emulator, is a particular case of supervised ML applied in the field of engineering design [9]. Their use cases have primarily been focused on accelerating computationally expensive numerical simulation models. In surrogate modeling, a statistical model is constructed to approximate the simulation output [9]. By applying the surrogate modeling pipeline, the TinyML-VAE surrogate approximates the original VAE, capturing its uncertainty quan-

This work was supported by NSF EPSCoR Award OIA-1920965.

tification capabilities while significantly reducing computational requirements. This work motivates future research in integrating large-scale ML models at the edge device for improvements in robustness.

For this study, a standard VAE was implemented and trained on historical magnetometer data, and the TinyML-VAE surrogate is an approximation of the former model. The performance of the surrogate was compared against the benchmark set by the standard implementation. The remainder of the paper is as follows. In Section-II the adopted methodology is explained, which elaborates on the data acquisition and processing steps, along with the development of the models, and their hardware deployment. Section-III presents and discusses the empirical results on the performance of the two models. Finally, Section-IV concludes the paper, with a note on future work.

II. Methodology

A. Data Overview, Acquisition And Processing

The Earth's magnetic field is a vector quantity that can be defined in the Cartesian framework using three orthogonal components: B_N, B_E, and B_Z. The vector components, when presented in a local magnetic coordinate system, are positive in the direction of north (N), east (E), and vertically down (Z). Most geomagnetic observatories deploy variometers (e.g., fluxgate magnetometers) that collect the component changes relative to an undetermined field. This undetermined field is called the baseline, and it depicts the average or background magnetic field present along that component without any external fields. The data for this paper were obtained from SuperMAG [2]. SuperMAG is a global consortium of institutions and governmental bodies that manages nearly 600 ground-based magnetometers worldwide. The platform offers access to verified perturbations in the Earth's magnetic field, all presented in a unified local magnetic coordinate system with consistent time intervals and a common baseline removal approach. The SuperMAG system isolates variations from electric currents in Earth's upper atmosphere by filtering out the slowly changing, predominant Earth main field [2]. SuperMAG's data processing mechanism is not performed in real time. Therefore, this paper employs historical raw and baseline-adjusted local terrestrial magnetic field data from the Ottawa (OTT) station for 2001-2018. The magnetometer in Ottawa is situated at a magnetic latitude of 54.98° N. It operates on a Universal Time (UT) offset of -5 hours, indicating that local midnight corresponds to 05:00 UT. The data recorded from OTT has less than 1% of missing values. Such incomplete data points were addressed through linear interpolation. The final dataset utilized for this study, $\vec{D} = \{\vec{D_R}, \vec{D_B}\}$, consists of the raw magnetic components data, $\vec{D_R} = \{B_N, B_E, B_Z\}$, and their baseline corrected counterparts, $\vec{D_B} = \{B'_N, B'_E, B'_Z\}$.

B. Model Overview And Implementation

1) Variational Autoencoder (VAE): Autoencoders (AE) are a type of neural network architecture used for unsupervised learning tasks [6]. A standard AE comprises two key subnetworks: an encoder and a decoder. The encoder maps the input into a lower-dimensional space called the latent-space. The idea is to capture the essential features or characteristics of the input data in a relatively smaller number of dimensions. The decoder then leverages the latent-space representation to output a reconstructed form of the input data. AE aims to estimate the optimal set of network parameters that minimize the discrepancy between the input and corresponding output.

AEs can be adapted to remove noise from data. Suppose the original dataset X consists of inherent noise, and we have a denoised counterpart X' for validation [6]. If X has N number of samples and M number of features ($N \times M$), then the encoder uses the activation function $f : \mathbb{R}^M \to \mathbb{R}^m$ for mapping. As shown in Eq-1, f maps each sample x_i in X to a point z_i in the latent-space $\mathbb{R}^m$, where $m < M$, and w_f represents the weight parameters of the encoder network. The decoder network uses a second activation function $g : \mathbb{R}^m \to \mathbb{R}^M$, which predicts $\hat{x}'_i$ from z_i, as exhibited in Eq-2, where $\hat{x}'_i$ is a denoised instance of x_i, and w_g denotes the weights of the decoder module. As mentioned, the AE utilizes a loss function $\mathcal{L}$ to minimize the difference between the denoised validation data X' and its corresponding decoder-predicted value, $\hat{X}$. The choice of the activation functions and the optimization algorithm used in minimizing $\mathcal{L}$ is contingent on the problem in focus. A common example of a loss function is mean-squared error (MSE), which is depicted in Eq-3.

$$z_i = f(x_i, w_f) \tag{1}$$

$$\hat{x}_i = g(z_i, w_g) \tag{2}$$

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x'_i - g(f(x_i, w_f), w_g))^2 = \frac{1}{N}\sum_{i=1}^{N}(x'_i - \hat{x}'_i)^2 \tag{3}$$

During the training of AE, both forward and backward propagation occurs [6]. The input data X is fed into the model, and the encoder maps it into the latent-space $\mathbb{R}^m$. The decoder uses the latent-space representation Z to predict the reconstructed data $\hat{X}'$. The $\hat{X}'$ is compared against the validation set X' via the loss function $\mathcal{L}$, and through backward propagation, the weight parameters w_f and w_g are updated. The above process is repeated until $\mathcal{L}$ reaches the required threshold value. In the case of testing, the AE model predicts the results through forward propagation only.

AEs learn a deterministic mapping from the input space to a fixed latent space. Its probabilistic counterpart, Variational AE (VAE), integrates Bayesian inference to exhibit both the latent-space representation and the predicted output as a posterior probability distribution [6]. Bayesian inference implements Bayes' Theorem, which posits that the posterior probability distribution is directly proportional to the product of the prior distribution and the likelihood function [7]. Given a posterior distribution, its variance quantifies uncertainty, making VAE a robust option relative to a standard AE. In VAE, the encoder produces a posterior distribution of the latent-space representation $f(Z|X, w_f)$ as demonstrated in Eq-4 and the sample generation of Z for the decoder is referred to in Eq-5. The decoder takes the latent-space samples $(\mu_{z,x_i'}, \sigma^2_{z,x_i}$ and predicts a posterior distribution of the reconstructed data $g(\hat{X}'|Z, w_g)$, as shown in Eq-6. The loss function for VAE $\mathcal{L}_{VAE}$ primarily consists of two components (See Eq-7) [6]. The first term is the expected negative log-likelihood. This factor motivates the decoder to learn data reconstruction. Should the decoder fail to reconstruct the original data adequately, it will cause $\mathcal{L}_{VAE}$ to incur a large cost. The second term is the Kullback–Leibler (KL) divergence, used as a regularization factor. The KL divergence measures the amount of information lost for using $f(Z|X, w_f)$ as an approximation for the actual latent-space representation distribution $p(Z)$. The KL divergence is an inference problem, and its mathematical formulation is presented in Eq-8. Variational inference (VI) solves the KL divergence term by approximating the posterior with a simpler distribution and minimizing the divergence between this approximate posterior and the prior. VI allows the optimization problem to learn the weight parameters of the encoder and decoder through backpropagation. For simplicity, it is specified that the prior $p(Z) \sim \mathcal{N}(0, 1)$. The above assumption allows the KL divergence problem to be decomposed to the terms expressed in Eq-9, where μ_i and σ_i are the mean and standard deviation of the i^{th} component of Z.

$$f(Z|X, w_f) \sim \mathcal{N}(\mu_{z,x_i'}, \sigma^2_{z,x_i'}) \tag{4}$$

$$Z = \mu(X) + \sigma(X)\epsilon, \epsilon \sim \mathcal{N}(0, 1) \tag{5}$$

$$g(\hat{X}'|Z, w_g) \sim \mathcal{N}(\hat{\mu_{x'}}, \hat{\sigma_{x'}}^2) \tag{6}$$

$$\mathcal{L}_{VAE} = -\mathbf{E}[\log g(\hat{X}'|Z, w_g)] + KL[f(Z|X, w_f)||p(Z)] \tag{7}$$

$$KL[f(Z|X, w_f)||p(Z)] = \int f(Z|X, w_f) \log \frac{f(Z|X, w_f)}{p(Z)} dz \tag{8}$$

$$KL[f(Z|X, w_f)||p(Z)] = \frac{1}{2}\sum_{i=1}^{N}(\sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2) \tag{9}$$

For this paper, a VAE was implemented using Python's TensorFlow and TensorFlow-probability libraries. The model takes raw magnetometer component values as input and predicts the posterior distribution of the baseline corrected values as output. The multivariate encoder network comprises a single variable input layer, three hidden layers, and an output layer with two neurons. The hidden layers are the TensorFlow-probability library's "Variational Dense Layers" that transform the noisy input data X into two output vectors: mean $(\mu_{z,x})$ and variance

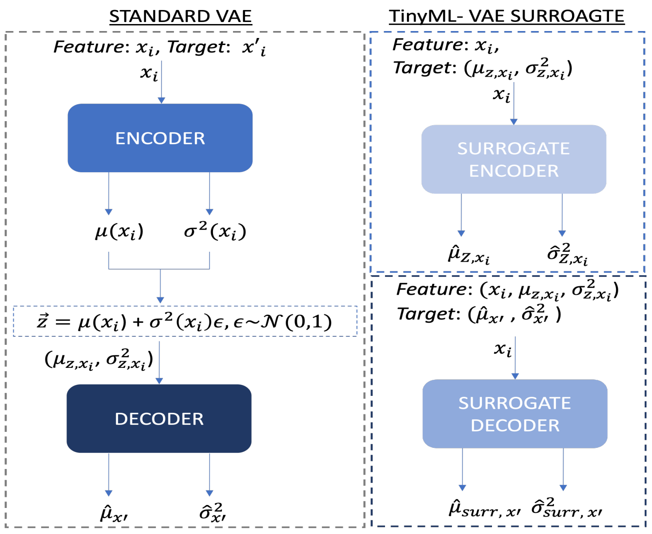
STANDARD VAE
Feature: x_i, Target: x'_i
x_i
ENCODER
$\mu(x_i)$
$\sigma^2(x_i)$
$\vec{z} = \mu(x_i) + \sigma^2(x_i)\epsilon, \epsilon \sim \mathcal{N}(0,1)$
$(\mu_{z,x_i}, \sigma^2_{z,x_i})$
DECODER
$\hat{\mu}_{x'}$
$\hat{\sigma}^2_{x'}$
TinyML- VAE SURROAGTE
Feature: x_i,
Target: $(\mu_{z,x_i}, \sigma^2_{z,x_i})$
x_i
SURROGATE ENCODER
$\hat{\mu}_{Z,x_i}$
$\hat{\sigma}^2_{Z,x_i}$
Feature: $(x_i, \mu_{z,x_i}, \sigma^2_{z,x_i})$
Target: $(\hat{\mu}_{x'}, \hat{\sigma}^2_{x'})$
x_i
SURROGATE DECODER
$\hat{\mu}_{surr,x'}$
$\hat{\sigma}^2_{surr,x'}$

Fig. 1. Overview of the TinyML-VAE Surrogate methodology.

$(\sigma^2_{z,x})$. These two vectors define the Gaussian distribution parameters representing the latent space. The encoder produces a tensor with "$2 \times dimensionalityofthelatentspace$" because the mean and variance are being modeled. The sampling function shown in Eq-5 takes $\mu_{z,x}$ and $\sigma^2_{z,x}$ and samples from the corresponding Gaussian distribution to produce the latent variable Z. The decoder network is a mirror image of the encoder, and it takes $(\mu_{z,x}, \sigma^2_{z,x})$ as its input and outputs the $\hat{\mu_{x'}}$ and $\hat{\sigma_{x'}}^2$, which are the mean and variance of the predicted denoised data $\hat{X}'$. Both networks use the Rectified Linear Unit (ReLU) as their activation function.

As mentioned above, the objective of this study is to perform baseline correction of the three raw magnetic component data (B_N, B_E, orB_Z). Distinct instances of the implemented VAE model were trained and tested for the three components separately. For example, an instance of the model was first trained and tested using B_N as input and B_N' as the target variable, and the same process was independently repeated for B_E, and B_Z. During the training and testing of the model for a particular component, the raw data of the component from $\vec{D_R}$ were provided as the input value for X. Its baseline corrected counterparts from $\vec{D_R'}$ were used as the ground truth during validation and testing. From the dataset, $\vec{D}$ 80% were taken for training, and the remaining 20% were used for testing.

2) Surrogate TinyML Model For VAE (TinyML-VAE Surrogate): Surrogate models, sometimes referred to as emulators or meta-models, are used to mimic the performance of complex simulations or algorithms but at a fraction of the computational cost [9]. The idea behind surrogate modeling is to use a simpler, less computationally intensive model to approximate the output of a more complex model. In essence, the surrogate model serves as a statistical approximation of the original model. The surrogate model is generally trained on a dataset that is generated by running the original, computationally-intensive model. This approach is highly relevant for TinyML

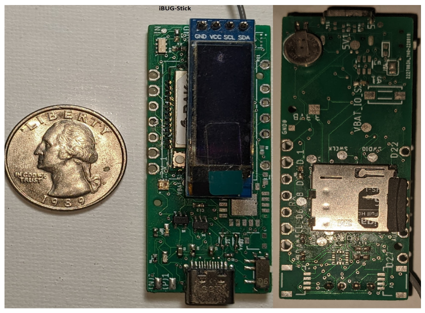

Fig. 2. The iBUG-Stick development board (Front and Back Views) is presented adjacent to a US cent coin, for size comparison.

implementations, which often have limitations on memory, processing power, and energy consumption.

To implement the TinyML-VAE surrogate model, initially, the standard VAE is treated as three distinct submodules: the encoder $f()$, the latent space representation Z, and the decoder model $g()$ (See Eq-4-6). In the standard VAE, at each time step, $f()$ takes a raw magnetic component data $x_i (x_i \epsilon X)$ and outputs $(\mu(x_i), \sigma(x_i)^2$. Then Eq-5 is used to derive a set of samples for the latent space representation $\vec{z_i}$. Given the set of values for $\vec{z_i}$ for a particular input x_i, the mean and variance of $\vec{z_i}$ is obtained $(\mu_{z,x_i}, \sigma^2_{z,x_i})$. The encoder of the surrogate model f_{surr} is trained with X as input and $(\mu_{z,x_i}, \sigma^2_{z,x_i})$ as target variable. Here, f_{surr} is a multivariate feed-forward neural network that consists of an input layer for a single variable and an output layer with two targets. The f_{surr} networks comprise three hidden layers, with ReLU as their activation function and MSE as the loss function.

The standard VAE decoder $g()$, at each time step, takes $(\mu_{z,x_i}, \sigma^2_{z,x_i})$ as input and predicts the mean and variance of the baseline corrected magnetic component $(\hat{\mu_{x'}}, \hat{\sigma_{x'}}^2)$ as output. The surrogate decoder model g_{surr} is a multivariate, multivariable feed-forward network where the input layer has three nodes and the output layer has two. In between, it has three hidden layers, all of which use ReLU as their activation function. The network g_{surr} is trained with $(x_i, \mu_{z,x_i}, \sigma^2_{z,x_i})$ as features $(\hat{\mu_{x'}}, \hat{\sigma_{x'}}^2)$ as target values. The network g_{surr} predicts the mean and variance of the baseline corrected magnetic component values $(\mu_{su\hat{r}r,x'}, \sigma_{su\hat{r}r,x'}{}^2)$. The trained surrogate encoder and the decoder are separately converted into TinyML models using the Tensorflow-Lite Micro library for deployment and testing using an MCU board. A summary of the TinyML-VAE surrogate's methodology is presented in Fig-1

C. Hardware And Deployment

The surrogate encoder, sampling function decoder TinyML model, trained for a particular terrestrial magnetic component data, were deployed into an iBUG-stick development board for testing (See Fig-2). The iBUG board is an ML-enabled IoT sensing platform, which has been used in the past literature for real-time environmental monitoring [10]. It has an RAK11300 Long Range Wide Area Network (LoRaWAN) module and a 133MHz dual-core Raspberry Pi RP2040 MCU. The appropriate input from the test dataset was provided as streaming value via the board's USB port. At each time step, for each test input, the TinyML encoder's output $(\hat{\mu}_{enc}, \hat{\sigma}^2_{enc})$. The decoder takes the $(\hat{\mu}_{enc}, \hat{\sigma}^2_{enc})$ as input, and predicts $(\hat{\mu}_{dec}, \hat{\sigma}^2_{dec})$. Both the encoder's and decoder's forecasts were recorded for further validation.

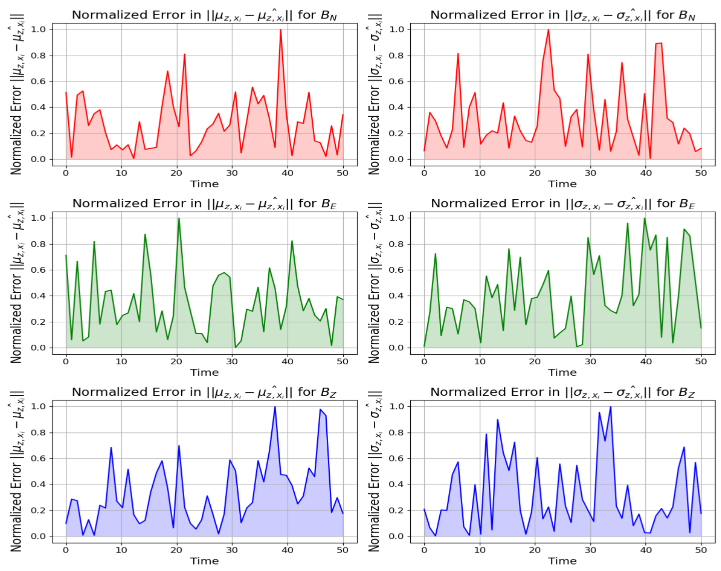

Fig. 3. Normalized error plot versus time (min), comparing the Standard VAE Encoder's output $(\mu_{z,x_i}, \sigma_{z,x_i})$, against the TinyML-VAE Surrogate Encoder's output, $(\mu_{\hat{z},x_i}, \sigma_{\hat{z},x_i})$.

III. Results

In this study, Normalized Error (NE) and Normalized Root Mean-Squared Error (NRMSE) have been adopted as the metrics of choice to compare the performances of the two models. Zero is considered a perfect fit in both metrics, whereas 1 signifies a maximum error. In Fig-3, the NE between the benchmark and surrogate predicted mean and variance are presented. The surrogate encoder trained for the component B_E had the poorest performance for mean and variance. It can also be observed that, generally, the mean prediction of the surrogate was more accurate than its corresponding variance. This is because the benchmark variances are samples from a normal distribution, and the surrogate encoder variance is a deterministic proxy. The observations are also consistent when NRMSE is considered. Table-I lists the NMRSE values for the standard VAE, TinyML-VAE encoder, and TinyML-VAE decoder. The standard VAE NMRSE is the model's performance compared to the ground truth or test dataset. Therefore, the standard model has no values under the variance rows, as the variance is just a quantification of the model's prediction uncertainty. In the case of the surrogate encoder and decoder, they compare against the benchmark result set by the standard VAE's encoder and decoder.

The variances of the two surrogates are considered, as they are a deterministic approximation of the standard uncertainty

TABLE I
NORMALIZED RMSE COMPARISON FOR MAGNETIC COMPONENTS

Variable	Norm. RMSE Standard VAE	Norm. RMSE TinyML-VAE Encoder	Norm. RMSE TinyML-VAE Decoder
B_N	0.41	0.54	0.58
B_E	0.43	0.61	0.66
B_Z	0.41	0.53	0.55
σ_N^2	–	0.55	0.58
σ_E^2	–	0.60	0.66
σ_Z^2	–	0.50	0.55

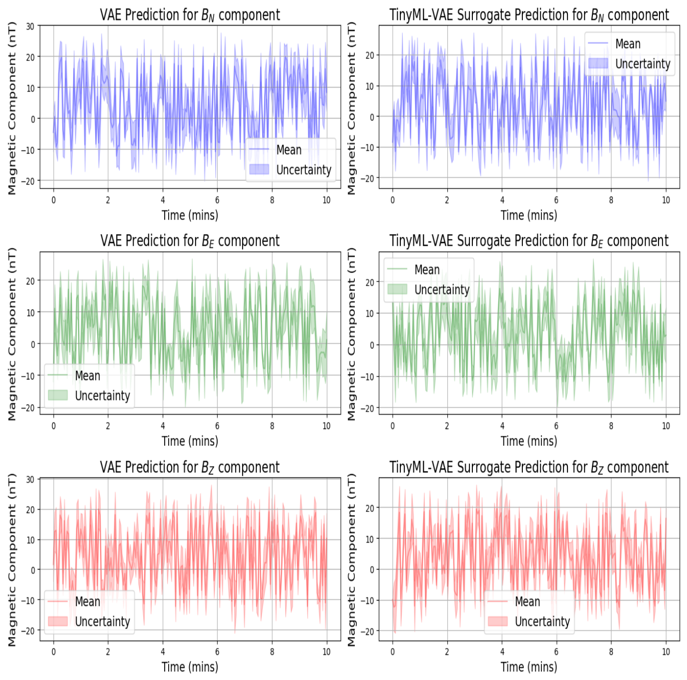

Fig. 4. Standard VAE and TinyML-VAE Surrogate Decoder's Predicted Mean and Variance For Each Magnetic Component (B_N, B_E, B_Z) Versus Time.

quantification. A similar trend can be observed with the model trained for B_E component performed the poorest. However, since NRMSE considers the entire prediction sample set, the variance's NRMSE seems comparable to its mean counterpart. This is because, unlike the NE, the NRMSE metric is an average over each model's entire prediction set. Consequently, the averaging operation is likely to reduce the overall error.

Finally, in Fig-4, the predicted mean and variance from the decoders of both standard VAE and surrogate VAE is presented in a time series plot, where at each time step, the predicted mean is enveloped by the upper and lower bound (its corresponding variance), which is a quantification of uncertainty. The authors of this paper highlight that the performance of the decoder and the overall uncertainty quantification approximation depends on the effective mapping of the standard model's latent space by the encoder. This mapping can be improved through a distributed or ensemble architecture, where multiple encoders are executed on distinct MCUs, and the results from each model can then be approximated into a distribution. A similar approach can be adopted for the decoders to ensure a more effective uncertainty quantification approximation.

IV. Conclusion And Future Work

In this study, surrogate ML techniques were employed to construct a TinyML model that emulates the functionality of a computationally expensive VAE. Consequently, the surrogate methodology allowed the TinyML model to approximate the uncertainty quantification capabilities of a standard VAE, thus ensuring robustness. The models were trained and tested for real-time baseline adjustments on historical magnetometer data. The performance of the TinyML model was evaluated against the benchmark set by the standard VAE. The proposed system is being integrated with a magneto-inductive sensor for real-time data calibration as an ongoing work on developing an ML-enabled autonomous magnetometer system. In addition, an active learning framework is being implemented in conjunction with a cloud-based platform. The goal is for the TinyML-VAE model to be retrained in the cloud and updated at the edge as the real-time data distribution shifts with time.

Acknowledgment

We thank members of the the "Machine-learning Algorithms for Geomagnetically Induced Currents In Alaska and New Hampshire" (MAGICIAN) team. This work was supported by NSF EPSCoR Award OIA-1920965.

References

[1] J. J. Love and C. A. Finn, "Real-time geomagnetic monitoring for space weather-related applications: Opportunities and challenges," *Space Weather*, vol. 15, no. 7, pp. 820–827, 2017. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017SW001665
[2] J. W. Gjerloev, "The SuperMAG data processing technique: TECHNIQUE," *Journal of Geophysical Research: Space Physics*, vol. 117, no. A9, pp. n/a–n/a, Sep. 2012. [Online]. Available: http://doi.wiley.com/10.1029/2012JA017683
[3] T. Siddique and M. S. Mahmud, "Real-time machine learning enabled low-cost magnetometer system," in *2022 IEEE Sensors*, 2022, pp. 1–4.
[4] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, "A comprehensive survey on tinyml," *IEEE Access*, pp. 1–1, 2023.
[5] Y. Li, X. Lu, Y. Wang, and D. Dou, "Generative time series forecasting with diffusion, denoise, and disentanglement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 009–23 022, 2022.
[6] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey," *arXiv preprint arXiv:2101.00734*, 2021.
[7] M. Liu, D. Grana, and L. P. de Figueiredo, "Uncertainty quantification in stochastic inversion with dimensionality reduction using variational autoencoder," *Geophysics*, vol. 87, no. 2, pp. M43–M58, 2022.
[8] Q. Lu and B. Murmann, "Enhancing the Energy Efficiency and Robustness of tinyML Computer Vision Using Coarsely-Quantized Log-Gradient Input Images," *ACM Transactions on Embedded Computing Systems*, p. 3591466, Apr. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3591466
[9] R. Alizadeh, J. K. Allen, and F. Mistree, "Managing computational complexity using surrogate models: a critical review," *Research in Engineering Design*, vol. 31, pp. 275–298, 2020.
[10] M. F. Yousuf, T. Siddique, and M. S. Mahmud, "iBUG: AI Enabled IoT Sensing Platform for Real-time Environmental Monitoring." IEEE, 2022.