Soft Checksum Based Out-of-Distribution Detection for Scientific Machine Learning

Elijah Lewis* mentored by Casey Lauer†
University of Illinois Urbana-Champaign, Department of Aerospace Engineering, Urbana IL, 61801

Surrogate machine learning models provide computationally efficient approximations of physical simulations, but their predictive accuracy often degrades when applied to inputs that differ from their training distribution. Out-of-distribution (OOD) detection is a critical challenge in scientific machine learning, particularly for regression models used in these physical simulations. We introduce a method for OOD detection based on a soft checksum, whose calculated error reflects the discrepancy between the checksum output and a deterministic function of the model's other outputs. We implement this approach by including a checknode in the output layer of a Neural Network and designing a composite loss function such that the neural network learns to produce accurate checksum predictions on in-distribution (ID) data and inaccurate checksum predictions for OOD samples. As the checksum function only depends on the output nodes, the checksum error can be predicted with a single forward pass with minimal added computational load. Evaluation on a 0D auto-ignition combustion dataset shows that this checksum error can effectively distinguish between ID and OOD predictions, even in the absence of ground truth labels.

I. Nomenclature

$\mathcal{D}$	=	dataset
N	=	number of data-points in dataset
$f(x)$	=	true function
$\boldsymbol{x} \in \mathbb{R}^d$	=	function inputs
$\boldsymbol{y} \in \mathbb{R}^k$	=	function outputs
$\hat{\boldsymbol{y}}$	=	model predictions
$\boldsymbol{\theta}$	=	model parameters
$\mathcal{L}$	=	loss function
$\mathbb{C}(\boldsymbol{y})$	=	checksum function of $\boldsymbol{y}$
$\hat{\mathbb{C}}_{\boldsymbol{y}}$	=	model checknode output
ϵ	=	model error
$\epsilon_{\mathbb{C}}$	=	checksum error
ω	=	frequency
λ	=	checksum loss weight

*Undergraduate sophomore, Department of Aerospace Engineering
†Doctoral candidate, Department of Aerospace Engineering

II. Introduction

Physics simulation software is vital to the scientific and engineering communities, as it provides the ability to analyze complex scenarios without costly lab equipment or challenges extracting quantities of interest. However, high-fidelity simulations are often computationally expensive, limiting their deployment in many scenarios. To address this, surrogate machine learning (ML) models have emerged as an efficient alternative, offering approximate results in less computational time through the use of neural networks [1, 2, 3]. This type of model, a *regression model*, generalizes to new scenarios by learning the underlying function and relationships between inputs and outputs (which are continuous and real-valued) from large precomputed simulation datasets. Once trained, these surrogate models can make predictions orders of magnitude faster than the costly simulations they were based on [4]. This decrease in computation time enables faster decision-making in scientific and engineering applications.

Despite their advantages, surrogate ML models have some significant drawbacks. A critical concern for their usage is accuracy, specifically when they encounter data that is outside of the distribution of their training data. The model will still make a prediction, but the output may be inaccurate. To demonstrate the potential failure of a surrogate model on OOD inputs, we present a simple example. Figure 1 presents a one-dimensional regression scenario demonstrating the concept of OOD data. The x values represent the inputs and the y values represent the outputs. The model is trained only on the in-distribution dataset, recognizing the trend: $y = x$. The plot demonstrates how a model trained only on in-distribution (ID) data ($0 < x < 3$) may fail to generalize correctly to OOD data ($3 \leq x < 5$) when the ID data is not chosen well, and does not capture some behavior of the true function.

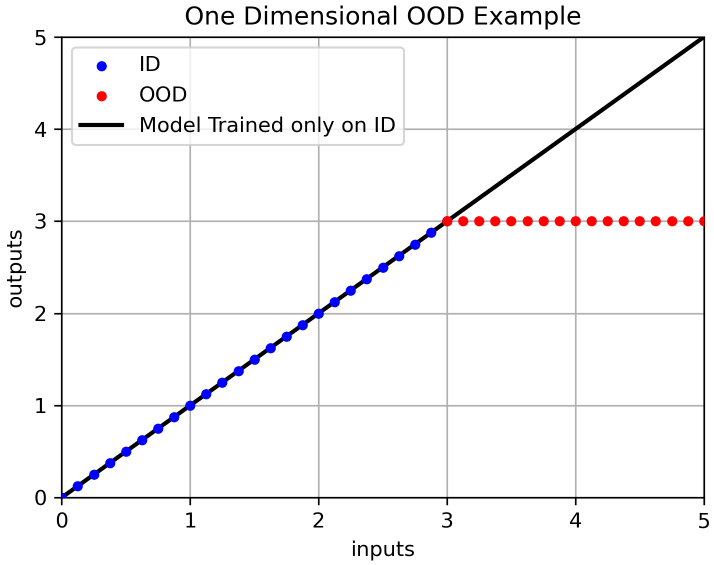

Fig. 1 Graph of OOD data impact in a one-dimensional regression scenario based on a piecewise function (Equation 1). The black line represents the predictions of a model trained only on ID data, which fails to correctly predict the OOD data.

$$y = \begin{cases} x & \text{if} \quad x < 3 \\ 3 & \text{if} \quad x \geq 3 \end{cases} \tag{1}$$

This figure highlights the importance of OOD detection, which aims to identify when a model is making predictions on data outside the domain that it was trained on. Without access to ground truth data, it is difficult to quantify the accuracy of regression models or assess the reliability of their predictions in real-world scenarios.

In contrast to regression models, classification models inherently produce probability distributions over predefined categories, allowing them to indicate prediction confidence. Classification models work to classify input data between

set categories, with an ideal classification model assigning a probability of 1 to the correct category and a 0 to all others. However, uncertain or inaccurate predictions often have multiple categories with similar probabilities, signaling reduced confidence. But this kind of uncertainty is not always a reliable reflection of how accurate the prediction actually is. Regression-based models output numerical predictions without any probability distributions or confidence scores, making uncertainty quantification significantly harder than classification. As a result, without proper mechanisms in place, there will be no indication that a model is making predictions in an unfamiliar or unreliable region. This makes OOD detection a crucial part of surrogate models, as it provides a way to detect potentially untrustworthy predictions and prevent the application of incorrect results in scientific or engineering workflows.

III. Background and Related Work

A. Existing OOD Detection Methods

There are several different methods that have been proposed for OOD detection and uncertainty estimation in regression problems. Bayesian neural networks, which aim to estimate uncertainty by learning a distribution over the model's weights, are a common approach, but they usually rely on complex sampling or approximation techniques that are slow and hard to use for large neural networks [5]. Monte Carlo dropout is a practical approximation of Bayesian methods, but has many of the same limitations with slower training time [6]. While dropout-based models provide an estimate of uncertainty, they can suffer from instability and rely on careful calibration of distance metrics. A widely adopted alternative to Bayesian and Monte Carlo methods is a deep ensemble. In a deep ensemble, the user trains multiple sub-models and uses the variance in their predictions as a measure of uncertainty. Deep ensembles are easy to implement, tend to produce better-calibrated uncertainties than approximate Bayesian methods, and are robust under distributional shift [5]. However they incur higher computational costs due to the need to train and evaluate multiple models.

Other approaches explicitly incorporate OOD data during training to improve anomaly detection. Outlier Exposure, introduced for classification tasks, trains models on unrelated inputs and penalizes confident predictions on the OOD samples [7]. Energy-based methods offer an alternative by using a scalar energy function derived from model outputs, training the network to assign low energy to ID data and high energy to outliers [8]. While originally developed for classification, both of these inspire our approach explored in this paper.

B. Checksums

Checksums have existed for decades to detect errors in data transmission, with Fletcher's Checksum first published in 1982 [9]. Their primary purpose is to ensure complete data integrity by generating a value from a larger dataset. This is done by appending check bytes to the end of a transmission in such a way that the calculated checksum is zero. If the recipient does not calculate a value of zero, a transmission error is indicated and the message is resent.

It is not always necessary to enforce checksums to maintain zero errors. In some instances it is acceptable to have small amounts of error, often with large transmissions such as images and videos. In other scenarios the initial transmission may be more important than the cost of resending or correcting the transmission. Multiple methods currently exist to estimate the amount of error by using error estimating codes or a soft checksum, and allow a set threshold for maximum error, where transmissions below this value do not need to be resent [10].

In this paper we implement the concept of checksums in machine learning to quantify prediction error for OOD data. As regression models always have nonzero error, a binary checksum like Fletcher's Checksum would unhelpfully flag all predictions. Instead, we borrow the term soft checksum and error estimation ideas to describe our method for differentiating between ID and OOD data.

IV. Soft Checksum based Out-of-Distribution Detection

Let us define a dataset $\mathcal{D} = \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$ with inputs $\mathbf{x} \in \mathbb{R}^d$ directly corresponding to outputs $\mathbf{y} \in \mathbb{R}^k$. A neural network is defined as $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$, with predicted outputs $\hat{\boldsymbol{y}}$, inputs $\boldsymbol{x}$. The parameters $\boldsymbol{\theta}$ are the weights and biases optimized based on the loss function $\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$, explained in more detail in Section IV.B. Model hyperparameters are user selected parameters such as width (amount of neurons per layer) and depth (amount of layers) of the network, the learning rate (how quick the model learns) and the batch size.

We now augment the model to include a checknode:

$$\left(\hat{\boldsymbol{y}},\ \hat{\mathbb{C}}_{\boldsymbol{y}}\right) = \hat{f}\ (\boldsymbol{x};\ \theta) \tag{2}$$

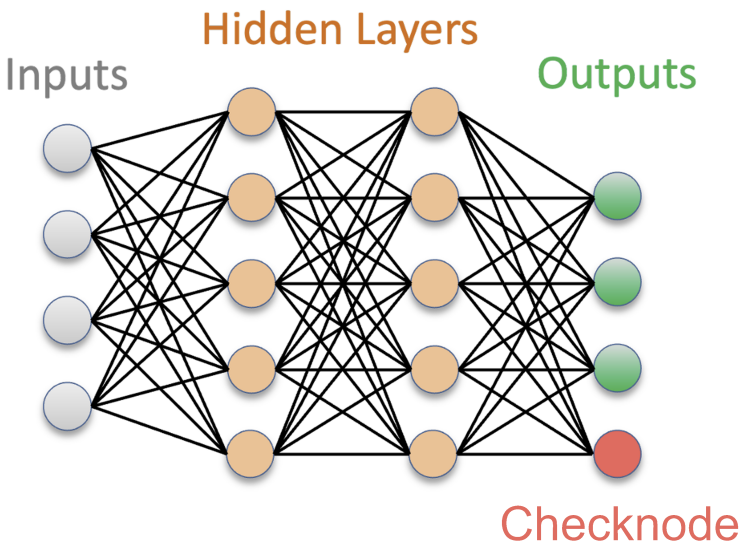

Fig. 2 Figure of an example neural network layout with checknode in the output layer.

Because the checksum function is only dependent upon the outputs, it can be applied to both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, meaning the checksum function can be easily applied to any regression based dataset. This allows us to calculate the accuracy of the checksum value in situations where there are no ground truth values to compare with the predictions. By using the same checksum function used during training, the checksum error $\epsilon_{\mathbb{C}}$ is calculated in Equation 3.

If the checksum error directly relates to the model error, $\epsilon_{\mathbb{C}}$ can be used to quantify the model's accuracy. In practice, the correlation is more abstract and requires specialized loss functions to encourage desired results.

In Equation 2, $\hat{\boldsymbol{y}}$ denotes the predicted outputs, and $\hat{\mathbb{C}}_{\boldsymbol{y}}$ represents the predicted output of a checksum function $\mathbb{C}(\boldsymbol{y})$. The checksum error $\epsilon_{\mathbb{C}}$, defined as the absolute difference between the predicted checksum and the checksum computed from the model predictions is given by:

$$\epsilon_{\mathbb{C}} = \left|\mathbb{C}(\hat{\boldsymbol{y}}) - \hat{\mathbb{C}}_{\boldsymbol{y}}\right| \tag{3}$$

This checksum error provides a quantitative measure of the discrepancy between predicted and computed checksums, enabling the identification of predictions that deviate from the learned relationships in the training data.

The dataset used during training is randomly divided into $\mathcal{D}_{\text{training}}$ and $\mathcal{D}_{\text{validation}}$, where the training dataset is used to train the model, and the validation dataset is used to validate that the model is not over-fitting the training data. All other data that the model may be exposed to is classified based on its input domain: if it is within the domain of $\mathcal{D}_{\text{training}}$ then it is ID and the model should ideally perform well, all other data outside this domain is $\mathcal{D}_{\text{OOD}}$ and presumed to

be predicted inaccurately. Distinguishing between these two cases is critical, as models are typically not equipped to recognize when they are making unreliable predictions on unfamiliar inputs.

A. Calculating Checksums

The checksum function is designed as a deterministic function applied to the predictions of a model. Because it is derived solely from the model's predictions, the checksum can help asses whether an output behaves in a way that is structurally consistent with the training distribution, serving as a tool to flag unusual or potentially unreliable predictions. Importantly, the function needs to be simple enough that the model still trains in a timely manner, yet complex enough that the model cannot easily generalize to OOD data. Equations 4 and 5 are checksum functions based on the sum or sine transformation of $\boldsymbol{y}$ or $\hat{\boldsymbol{y}}$.

$$\mathbb{C}(\boldsymbol{y}) = \sum_i y_i \tag{4}$$

$$\mathbb{C}(\boldsymbol{y}) = \sin\left(\omega \left| \sum_{i=0}^{k} y_i \right|\right) \tag{5}$$

B. Loss Functions

In all neural networks, a loss function is used to quantify the difference between the predictions and the true values. This difference acts as a guide for the learning process, allowing the network to change its internal parameters ($\boldsymbol{\theta}$) through backpropagation in order to minimize the error. Various loss functions exist to relate the predictions to the true values, with mean squared error (MSE) (Equation 6) among the most common, and what we use to train the predicted outputs ($\hat{\boldsymbol{y}}$) of our models.

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=1}^{k} (y_i - \hat{y}_i)^2 \tag{6}$$

Checksum Loss (α)

A beneficial aspect of neural networks is that the loss functions can be added together such that a single total loss value could be made up of multiple terms that each capture different objectives or constraints. The first of these terms is the checksum loss term, based on MSE between the checknode and the checksum function applied to the true outputs $\boldsymbol{y}$:

$$\mathcal{L}_{\text{Checksum Loss}}(\boldsymbol{y}, \hat{\mathbb{C}}_{\boldsymbol{y}}) = \frac{\alpha}{k} \left(\mathbb{C}(\boldsymbol{y}) - \hat{\mathbb{C}}_{\boldsymbol{y}}\right)^2 \tag{7}$$

In order to maintain the same order of magnitude as in Equation 6, the checksum loss term is also divided by the number of outputs k before it is added to the total loss value. An additional variable is used to quantify the weight of the checksum loss term, α.

Checksum Penalty (β)

The checksum penalty loss term, (Equation 8) encourages the model to produce outputs consistent with the checksum function of the model outputs: $\hat{\mathbb{C}}_{\boldsymbol{y}} \rightarrow \mathbb{C}(\hat{\boldsymbol{y}})$. Similar to Equation 7, a weight (β) is applied.

$$\mathcal{L}_{\text{Checksum Penalty}}(\hat{\boldsymbol{y}}, \hat{\mathbb{C}}_{\boldsymbol{y}}) = \frac{\beta}{k} \left(\mathbb{C}(\hat{\boldsymbol{y}}) - \hat{\mathbb{C}}_{\boldsymbol{y}}\right)^2 \tag{8}$$

Checksum Reward (γ)

To encourage the predicted checksum value $\hat{\mathbb{C}}_{\boldsymbol{y}}$ to be far from $\mathbb{C}(\hat{\boldsymbol{y}})$ for OOD data, another loss term must be used. This term is called the checksum reward, and is calculated on OOD data generated during training using the method outlined in Section IV.C. The reward term returns the minimum of two values: the reciprocal of the mean squared error and a new term, κ, which is the maximum enforceable value that the reward term will work on, negating this term when the checknode value ($\hat{\mathbb{C}}_{\boldsymbol{y}}$) is sufficiently far from the calculated checksum value ($\mathbb{C}(\hat{\boldsymbol{y}})$). Similar to the checksum penalty term, γ acts as a weight. Additionally, μ functions as a small value to prevents a division by zero error.

$$\mathcal{L}_{\text{Checksum Reward}}(\hat{\boldsymbol{y}}_{\text{OOD}}, \hat{\mathbb{C}}_{\boldsymbol{y}}) = \gamma \min\left(\kappa, \left[\left(\mathbb{C}(\hat{\boldsymbol{y}}_{\text{OOD}}) - \hat{\mathbb{C}}_{\boldsymbol{y}}\right)^2 + \mu\right]^{-1}\right) \tag{9}$$

These various loss functions can be combined into a single loss value that the model minimizes during training. The total loss value is calculated as in Equation 10.

$$\mathcal{L}_{\text{Total}}(\boldsymbol{y}, \hat{\boldsymbol{y}}, \hat{\mathbb{C}}_{\boldsymbol{y}}) = \mathcal{L}_{\text{MSE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathcal{L}_{\text{Checksum Loss}}(\boldsymbol{y}, \hat{\mathbb{C}}_{\boldsymbol{y}}) + \mathcal{L}_{\text{Checksum Penalty}}(\hat{\boldsymbol{y}}, \hat{\mathbb{C}}_{\boldsymbol{y}}) + \mathcal{L}_{\text{Checksum Reward}}(\hat{\boldsymbol{y}}_{\text{OOD}}, \hat{\mathbb{C}}_{\boldsymbol{y}}) \tag{10}$$

C. Generating Out-Of-Distribution Inputs

For the checknode error to effectively identify when a prediction is OOD, it is beneficial to create a gap between the ID and OOD data. This can be achieved by ensuring the checksum error remains low for ID data and increases significantly for OOD data. Specifically, when the input is OOD, the predicted checknode value ($\hat{\mathbb{C}}_{\boldsymbol{y}}$) should noticeably deviate from the value computed by apply the checksum function to the model's predicted outputs ($\mathbb{C}(\hat{\boldsymbol{y}})$). To encourage this behavior, the model is exposed to synthetic OOD data $\boldsymbol{x}_{\text{OOD,Training}}$ during training as detailed in Section IV.B. Because access to real OOD datasets during training is often impractical or unavailable, these data points are generated randomly.

Out-of-distribution data includes inputs that either fall outside the bounds of a dataset or lie in regions the model has not seen, even if that is within the overall bounds. To create a function that can be generalized to all regression datasets, the function must only be dependent on the ID training data.

We generate OOD input samples by first determining the input range using the minimum and maximum of each input dimension. Then, for each sample, a random number of dimensions are chosen to be OOD, and the remaining dimensions are kept ID, with an upper bound of dimensions to be ID Max, which determines the shape of the synthetic OOD data as demonstrated in Figure 3, an example of synthetic OOD data generated from a chemical combustion kinetics dataset. A more complete description of the dataset used in this figure is provided in Section IV.D.

The OOD dimensions are filled with values randomly drawn from a user-chosen range outside the original data bounds, and the ID dimensions are values uniformly sampled within the input data range.

D. Dataset Generation

To assess the effectiveness in detecting OOD inputs of the proposed soft checksum-based OOD detection method, we train and evaluate a neural network on a 0D auto-ignition combustion dataset. The dataset is populated with data from multiple *Cantera* simulations, a trusted numerical solver [11]. The chemical mechanism includes seven species:

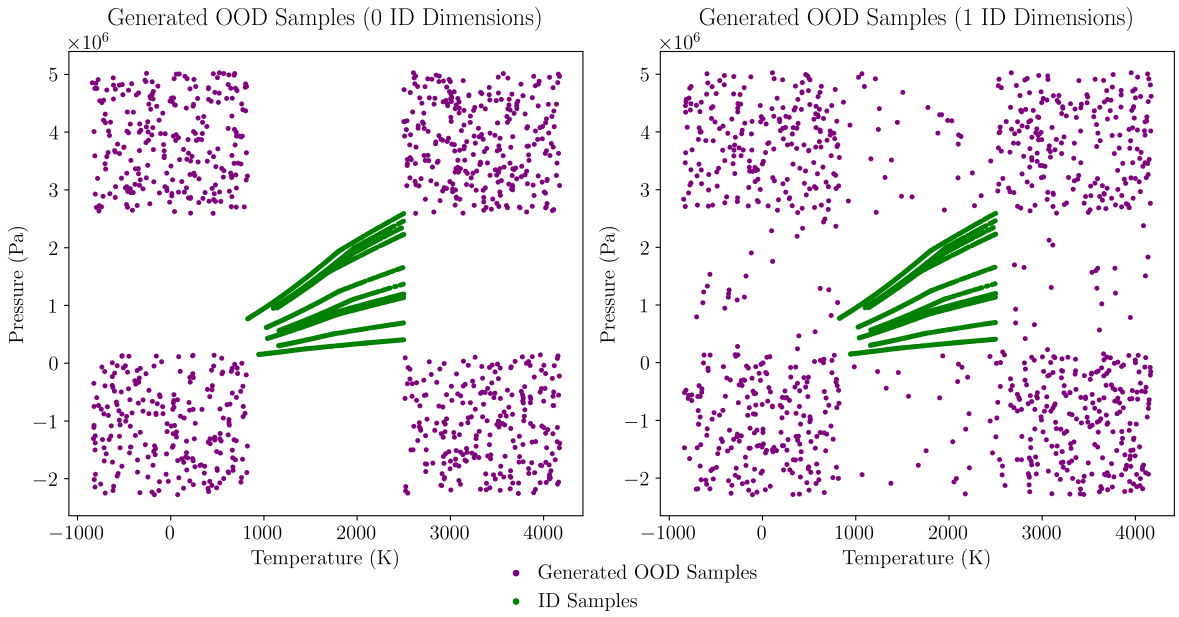

Fig. 3 Synthetic OOD inputs for various ID Max values of a chemical combustion kinetics dataset with 9 input dimensions. An ID Max value of 0 (left) guarantees all dimensions to be OOD, resulting in points only existing in the corners surrounding the ID data. An ID Max value of 1 (right) includes data points that directly border the ID distribution.

$C_2H_4, O_2, H_2, CO, CO_2, H_2O$ and inert species N_2. There are three reactions, Equations 11, 12, 13.

$$C_2H_4 + O_2 \rightarrow 2H_2 + 2CO \tag{11}$$

$$CO + \frac{1}{2}O_2 \rightleftharpoons CO_2 \tag{12}$$

$$H_2 + \frac{1}{2}O_2 \rightleftharpoons H_2O \tag{13}$$

Cantera uses known reaction parameters to numerically solve for the species net production rate, ω, at a given state, and integrate forward in time according to Equation 14, where $\mathbf{Y}$ is the species mole fractions. Reference [11] for more information on the governing equations and numerical methods.

$$\frac{d\mathbf{Y}}{dt} = \omega \tag{14}$$

Calculating ω can be a costly step in the simulation process, and therefore we aim to train a surrogate model to replace this calculation. The data is split into inputs: temperature, pressure, and $\mathbf{Y}$, and outputs: ω.

We define OOD data based on an arbitrary boundary within the simulation space. Data points with peak temperatures below 2500 K are treated as ID, while those above this threshold are designated as OOD, as illustrated in Figure 4. The ID data is randomly partitioned into training and validation subsets, while all OOD samples are held out, mimicking the scenario where they were never included in the first place.

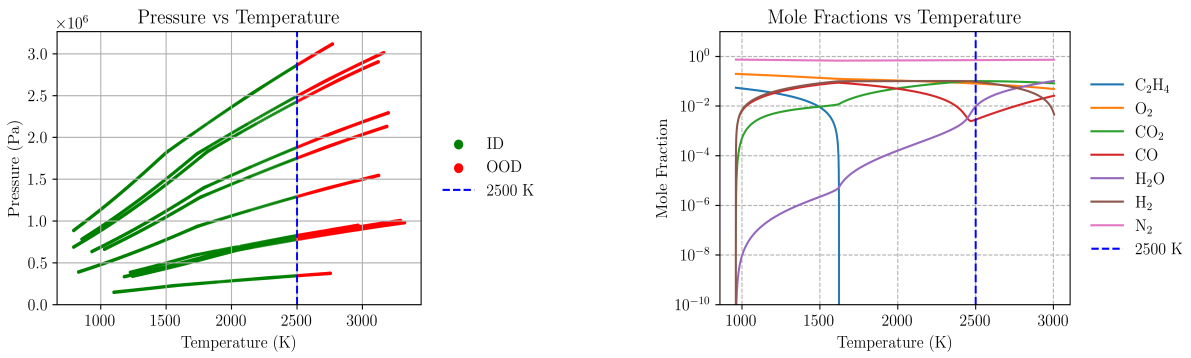

Fig. 4 Simulation results from the 0D auto-ignition combustion simulation, with ID and OOD regions separated at 2500 K.

V. Experimental Setup

The regression model is a neural network augmented with a checknode in the output layer, as described in Section IV. The network is trained with the composite loss function (section IV.B) consisting of checksum loss α, penalty β, and reward weights γ of different values. Each loss component weight (α, β, γ) is set to either 0 or 1 to isolate its effect on performance. The summation and sine checksum function (Equations 4 and 5) were applied across every combination of these weights, including the baseline case $(\alpha, \beta, \gamma = 0)$. The baseline configuration has all checksum loss components disabled, effectively training a standard neural network with an untrained checknode. This serves as a control case, allowing direct comparison to models trained with one or more checksum loss terms activated. By evaluating performance metrics under this setting, we establish a reference point for improvement and comparison.

Evaluation was performed on the chemical kinetics dataset (section IV.D), with all hyperparameters held constant to ensure fair comparison. We used 3 hidden layers each with 256 nodes per layer, a learning rate of 0.0001, batch size of 256, and trained each model for 1500 epochs. For OOD sample generation (section IV.C) in the checksum reward term, we set the in-distribution maximum to allow 1 ID value per sample, and sampled the out-of-distribution points uniformly in the range 10^{-6} to 1 times the original data bounds.

VI. Results

Checksum Function	α	β	γ	FNR 95% ↓	FNR 99% ↓	AUROC ↑	ID Error ↓
Summation Checksum	0	0	0	37.09%	56.98%	87.86%	3.07×10^{-4}
	0	0	1	26.20%	32.21%	84.41%	2.97×10^{-5}
	0	1	0	29.8%	48.38%	90.52%	3.99×10^{-5}
	0	1	1	9.81%	18.04%	95.88%	4.12×10^{-5}
	1	0	0	10.66%	15.70%	95.78%	$\mathbf{2.35 \times 10^{-5}}$
	1	0	1	**3.22**%	**5.51**%	**98.74**%	7.16×10^{-5}
	1	1	0	18.38%	28.77%	93.77%	5.63×10^{-5}
	1	1	1	8.25%	19.25%	92.99%	6.35×10^{-5}
Sine Checksum	0	0	0	79.82%	90.17%	60.16%	$\mathbf{2.86 \times 10^{-5}}$
	0	0	1	15.91%	25.00%	95.18%	6.74×10^{-5}
	0	1	0	5.19%	16.27%	97.91%	6.43×10^{-3}
	0	1	1	**0.69**%	3.90%	98.77%	7.00×10^{-3}
	1	0	0	1.06%	**1.88**%	99.65%	1.11×10^{-4}
	1	0	1	0.71%	1.89%	**99.66**%	1.02×10^{-4}
	1	1	0	4.90%	7.23%	97.93%	5.25×10^{-2}
	1	1	1	5.13%	7.21%	98.32%	5.27×10^{-2}

Table 1 Comparison of Summation and Sine Checksum functions under different checksum loss (α), penalty (β), and reward weights (γ). All values are averaged over five runs, with best results in bold, ↓ indicates lower is better, ↑ indicates higher is better.

To benchmark our model's OOD detection performance, we used several common metrics, as shown in Table 1. In high-stakes scientific modeling, minimizing false negatives while maintaining predictive confidence is critical, making FNR, AUROC and ID prediction error particularly relevant to evaluating reliability. False Negative Rate (FNR) represents the percentage of OOD data with a checksum error below a specific percentile of ID data checksum error. AUROC (Area under the receiver operating characteristic curve) is a measure of the probability that a randomly chosen OOD sample has a higher error than a randomly chosen ID sample, ranging from 0.5 (randomly chosen) to 1.0 (perfect detection). We also report the average ID prediction error to assess the impact of our method on model accuracy.

Reported metrics reflect the average across five independent training runs for each configuration. This is because the neural network's internal parameters are typically initialized randomly, which can lead to fluctuations in performance and error metrics. Averaging results across multiple runs helps mitigate the impact of outliers, offering a more reliable estimate of general model behavior.

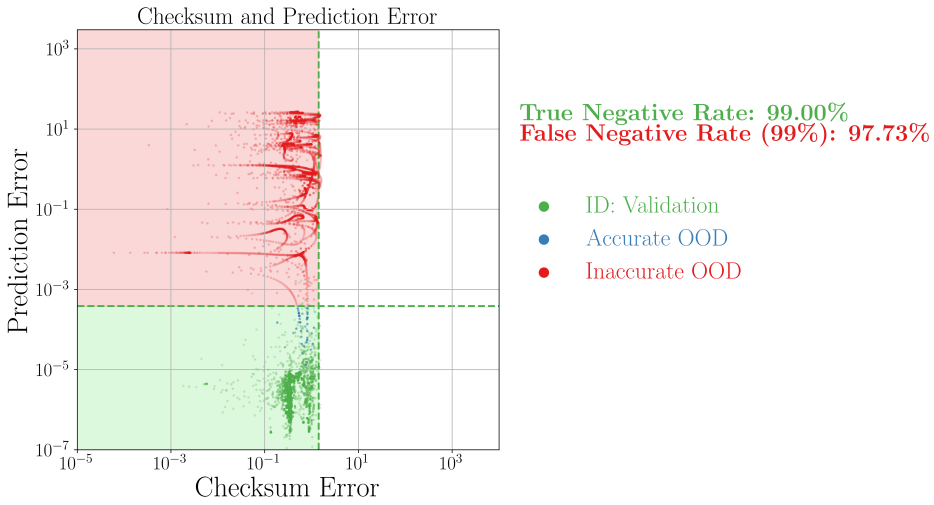

Fig. 5 Checksum and prediction error for baseline checksum neural network of width 256 and depth 3 using the Sine Checksum equation with $\omega = \pi$, $\text{OOD}_{\text{Min}} = 0.0001$, $\text{OOD}_{\text{Max}} = 1$, $\text{ID}_{\text{Max}} = 1$. Network trained without the checksum loss, penalty, or reward weights $(\alpha, \beta, \gamma = 0)$.

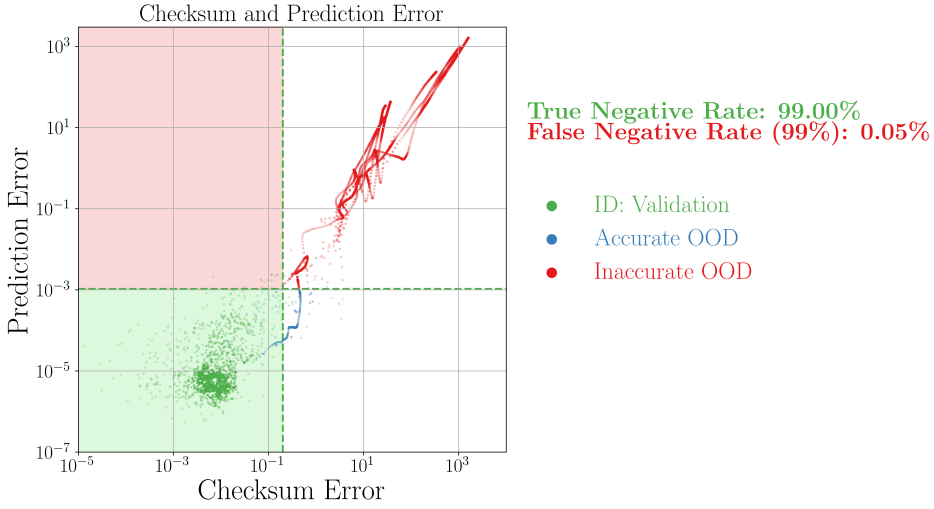

Fig. 6 Checksum and prediction error for tuned checksum neural network of width 256 and depth 3 using the Sine Checksum equation with $\omega = \pi$, $\text{OOD}_{\text{Min}} = 0.0001$, $\text{OOD}_{\text{Max}} = 1$, $\text{ID}_{\text{Max}} = 1$. Network trained using the checksum loss and reward weights $(\alpha, \gamma = 1)$.

Figure 5 and Figure 6 plot checksum error versus prediction error for the baseline and tuned sine checksum networks. Points are colored by dataset: ID validation (green), accurate OOD (blue), and inaccurate OOD (red). The vertical line labeled true negative rate is the checksum error threshold at which 99% of ID points fall below. This threshold is used to flag any data with a higher checksum error as OOD. The horizontal line is the prediction error at which 99% of ID data is below. The green section of the graph is the *true negative* region, indicating data with low prediction error and no OOD label. Most ID data should fall in this region, with minimal OOD presence. The red section is the *false negative* region, indicating data with high prediction error that was not identified to be OOD. An effective OOD detector produces a strong correlation between checksum and prediction errors, with few or no OOD points in the red false negative region. These findings are explored in more detail in the following discussion, which analyzes how checksum structure and loss weight settings influence detection performance.

VII. Discussion

Figures 5 and 6 demonstrate the effectiveness of the sine checksum function in distinguishing OOD data. Figure 5 shows a baseline model trained without checksum loss terms, while Figure 6 presents a tuned model using both checksum loss and reward terms. The contrast highlights the importance of checksum loss function configuration in shaping the model's OOD sensitivity. In the baseline model (figure 5, $\alpha, \beta, \gamma = 0$), checksum and prediction errors are uncorrelated: both ID and OOD points fall below the checksum error threshold, with OOD points demonstrating high prediction error, indicating the network cannot distinguish OOD inputs. After applying the checksum loss and reward weights ($\alpha, \gamma = 1$) (figure 6), there is a clear diagonal trend, and nearly all OOD points are above the checksum error threshold—only 0.05% of OOD remain misclassified at the 99% threshold. This demonstrates that a properly tuned checknode produces a strong linear correlation between checksum and prediction error, effectively isolating OOD samples.

We systematically evaluated both the summation and sine checksum functions by toggling the checksum loss weight (α), penalty weight (β), and reward weight (γ) between 0 and 1. The most effective OOD setting for summation checksum is when both the loss and reward weights are enabled ($\alpha = 1, \beta = 0, \gamma = 1$), yielding an FNR(99%) = 5.51%, an AUROC of 98.74%, and an ID Prediction error of 7.16×10^{-5}; all other summation configurations produced FNR(99%) above 15% and AUROC below 96%. ID Prediction Error for the summation checksum remained stable across all weight settings, with the lowest error (2.35×10^{-5}) for only the loss term ($\alpha = 1, \beta = 0, \gamma = 0$). Notably, AUROC in the summation checksum remains above 84.41% and exceeds 90% in most configurations, demonstrating robust overall discrimination.

The sine checksum offers higher sensitivity to the loss terms: activating the loss term alone ($\alpha = 1, \beta = 0, \gamma = 0$) yielded the lowest FNR(99%) of 1.88%, with nearly identical performance when adding the reward term ($\alpha = 1, \beta = 0, \gamma = 1$) at 1.89%. This increased sensitivity may be due to the nonlinearity of the sine checksum function, which amplifies small deviations in output behavior. The best overall AUROC performance for both checksum functions is the checksum loss and reward terms ($\alpha = 1, \beta = 0, \gamma = 1$), with the sine function achieving an average AUROC score of 99.66% and the summation checksum achieving 98.74%. The best performing model with this setup is depicted in Figure 6. However, sine's ID Prediction Error is much less stable, with the lowest error for the baseline ($\alpha = 0, \beta = 0, \gamma = 0$) of 2.86×10^{-5} ranging up to 5.27×10^{-2} for all weights active ($\alpha = 1, \beta = 1, \gamma = 1$).

Overall, AUROC proves to be the most comprehensive OOD metric, capturing the trade-off between true positive and false positive rates across all thresholds. Both checksum functions exceed 98% AUROC with checksum loss and reward weights ($\alpha = 1, \beta = 0, \gamma = 1$), but the sine checksum function consistently outperforms the summation checksum when any loss term is used. These observations confirm the reliability and flexibility of the soft checksum method across a range of configurations.

VIII. Conclusion

Incorporating the soft-checksum method into regression models significantly improves OOD detection without noticeably degrading in-distribution accuracy, provided loss function weights are properly selected. There is no significant increase in model complexity or memory usage. With the checksum loss and reward weights activated ($\alpha = 1, \beta = 0, \gamma = 1$), the sine checksum's average AUROC score is 99.66%, at the cost of larger ID Prediction Error compared to the summation checksum. The summation checksum achieves a favorable trade-off between OOD detection and ID accuracy, making it a reliable, low-complexity choice. The combination of checksum loss and reward weights ($\alpha, \gamma = 1$) consistently delivers the best AUROC values for both checksum functions, validating the soft-checksum method as an effective strategy for improving surrogate model robustness.

IX. Future Work

Building on our promising results, we plan to pursue several directions to further validate and extend the soft-checksum method.

Application to Additional Dataset

To further evaluate the soft-checksum OOD detection method, we will investigate its performance on more complex regression datasets.

Benchmarking Against Existing OOD Methods

We plan to compare the performance of the method against other prominent OOD detection techniques (deep ensembles [5, 7, 12] and Bayesian dropout neural networks [13]) using metrics such as FNR, AUROC and ID Prediction Error.

Checksum Function Exploration

The relationship between OOD detection and checksum function performance is still largely unknown, and further research is needed to determine which function is best suited for each use case. Additionally, the exploration of alternative checksum functions such as a polynomial or Fourier series function is planned.

Multiple Checknodes

Using multiple checknodes in the output layer could enhance detection and enable the use of several checksum functions at the same time.

References

[1] Vurtur Badarinath, P., Chierichetti, M., and Davoudi Kakhki, F., "A Machine Learning Approach as a Surrogate for a Finite Element Analysis: Status of Research and Application to One Dimensional Systems," *Sensors*, Vol. 21, No. 5, 2021. https://doi.org/10.3390/s21051654, URL https://www.mdpi.com/1424-8220/21/5/1654.

[2] Fritzen, F., Fernández, M., and Larsson, F., "On-the-Fly Adaptivity for Nonlinear Twoscale Simulations Using Artificial Neural Networks and Reduced Order Modeling," *Frontiers in Materials*, Vol. 6, 2019. https://doi.org/10.3389/fmats.2019.00075, URL http://dx.doi.org/10.3389/fmats.2019.00075.

[3] Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., and Perdikaris, P., "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *Journal of Computational Physics*, Vol. 394, 2019, p. 56–81. https://doi.org/10.1016/j.jcp.2019.05.024, URL http://dx.doi.org/10.1016/j.jcp.2019.05.024.

[4] Edelen, A., Neveu, N., Frey, M., Huber, Y., Mayes, C., and Adelmann, A., "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems," *Phys. Rev. Accel. Beams*, Vol. 23, 2020, p. 044601. https://doi.org/10.1103/PhysRevAccelBeams.23.044601, URL https://link.aps.org/doi/10.1103/PhysRevAccelBeams.23.044601.

[5] Lakshminarayanan, B., Pritzel, A., and Blundell, C., "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," , 2017. URL https://arxiv.org/abs/1612.01474.

[6] Wang, S. I., and Manning, C. D., "Fast dropout training," *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, JMLR.org, 2013, p. II–118–II–126.

[7] Hendrycks, D., Mazeika, M., and Dietterich, T., "Deep Anomaly Detection with Outlier Exposure," , 2019. URL https://arxiv.org/abs/1812.04606.

[8] Liu, W., Wang, X., Owens, J. D., and Li, Y., "Energy-based Out-of-distribution Detection," , 2021. URL https://arxiv.org/abs/2010.03759.

[9] Fletcher, J., "An Arithmetic Checksum for Serial Transmissions," *IEEE Transactions on Communications*, Vol. 30, No. 1, 1982, pp. 247–252. https://doi.org/10.1109/TCOM.1982.1095369.

[10] Lee, M.-S., and Bahk, S., "Soft Checksum Method for Error-tolerant Multi-hop Transmission in Wireless Sensor Networks," *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, 2021, pp. 1895–1897.

[11] Goodwin, D. G., Moffat, H. K., Schoegl, I., Speth, R. L., and Weber, B. W., "Cantera: An Object-oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes," https://www.cantera.org, 2024. https://doi.org/10.5281/zenodo.14455267, version 3.1.0.

[12] Mathelin, A., Deheeger, F., Mougeot, M., and Vayatis, N., "Deep Anti-Regularized Ensembles provide reliable out-of-distribution uncertainty quantification," , 2023. URL https://arxiv.org/abs/2304.04042.

[13] Nguyen, A. T., Lu, F., Munoz, G. L., Raff, E., Nicholas, C., and Holt, J., "Out of Distribution Data Detection Using Dropout Bayesian Neural Networks," , 2022. URL https://arxiv.org/abs/2202.08985.