A Stochastic Polynomial Expansion for Uncertainty Propagation through Networks

Anonymous authors
Paper under double-blind review

Abstract

Network-based machine learning constructs are becoming more prevalent in sensing and decision-making systems. As these systems are implemented in safety-critical environments such as pedestrian detection and power management, it is crucial to evaluate confidence in their decisions. At the heart of this problem is a need to understand and characterize how errors at the input of networks become progressively expanded or contracted as signals move through layers, especially in light of the non-trivial nonlinearities manifest throughout modern machine learning architectures. When sampling methods become expensive due to network size or complexity, approximation is needed and popular methods include Jacobian (first order Taylor) linearization and stochastic linearization. However, despite computational tractability, the accuracy of these methods can break down in situations with moderate to high input uncertainty. Here, we present a generalized method of propagating variational multivariate Gaussian distributions through neural networks. We propose a modified Taylor expansion function for nonlinear transformation of Gaussian distributions, with an additional approximation in which the polynomial terms act on independent Gaussian random variables (which are identically distributed). With these approximated higher order terms (HOTs), we obtain significantly more accurate estimation of layer-wise distributions. Despite the introduction of the HOTs, this method can propagate a full covariance matrix with a complexity of $\boldsymbol{O}(n^2)$ (and $\boldsymbol{O}(n)$ if only propagating marginal variance), comparable to Jacobian linearization. Thus, our method finds a balance between efficiency and accuracy. We derived the closed form solutions for this approximate Stochastic Taylor expansion for seven commonly used nonlinearities and verified the effectiveness of our method in deep residual neural networks, Bayesian neural networks, and variational autoencoders. This general method can be integrated into use-cases such as Kalman filtering, adversarial training, and variational learning.

1 Introduction

A fundamental problem in uncertainty estimation and verification is to characterize how a given input distribution becomes transformed by the operant function (succinctly, $Y = f(X)$). When f takes the form of a modern machine learning (ML) architecture, this problem quickly becomes analytically intractable, necessitating either sampling methods or approximation. The predominant approximation technique remains Jacobian linearization (JL), i.e., deterministic first order Taylor expansion around the mean of input distribution. To give a few examples in ML contexts, Gandhi et al. (2018); Dera et al. (2021); Petersen et al. (2024) used Jacobian linearization for uncertainty propagation through networks, and Beiu et al. (1994); Abdelaziz et al. (2015) used a piece-wise linear approximation of **sigmoid** functions prior to JL. Perhaps unsurprisingly, these methods work best in low-uncertainty regimes, because taking derivate at input mean ignores the uncertainty of the derivative itself.

In another line of research, Bayesian neural networks try to do uncertainty awareness training by fitting Gaussians to their weights, so they will naturally output distributions rather than a deterministic results, avoiding overconfidence. In exact Bayesian inference, one needs to solve the posterior distribution of the

network parameters θ given data D, i.e. $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta')p(\theta')d\theta'}$. The marginal distribution is an integral of the likelihood $p(D|\theta')$ over all possible combinations of network parameters. Since the likelihood is intractable due to the nonlinearity, this integral has to be approximated.

Early approaches attempted to precompute posterior mean and variance lookup tables to avoid Monte Carlo sampling at runtime (Hinton and van Camp, 1993), but this necessitated discretizing continuous variables, introducing imprecision. Monte Carlo-based methods remained popular for some time—Graves proposed Monte Carlo Variational Inference (MCVI) to approximate the evidence lower bound, though gradient estimation via sampling remained a limitation.

An alternative approach, the method of moments, seeks to approximate the output distribution by matching the first and second moments of propagated distributions to a Gaussian. The exact moments of Gaussian distributions transformed by ReLU were first derived in Frey and Hinton (1999) and later reused in several works, such as Gast and Roth (2018). Spiegelhalter and Lauritzen and MacKay provided approximations for **sigmoid** and **tanh** functions using different reformulations, while Wang et al. presented another similar approach. Shekhovtsov and Flach developed an analytical method for propagating uncertainty through **argmax** and softmax, assuming logistic and Gumbel priors.

To integrate these ideas into BNN training, Jylänki et al. (2014) introduced expectation propagation. A more scalable alternative, probabilistic backpropagation (PBP), was later proposed by Hernández-Lobato and Adams (2015), which propagates moments forward through the network to estimate the marginal likelihood, then backpropagates gradients with respect to approximated posterior parameters. However, PBP only propagates the diagonal of covariance. To address this limitation, Wu et al. (2019) introduced a formulation that enables full covariance propagation through ReLU and Heaviside activations.

Contributions To summarize, current uncertainty propagation of variational Gaussian distributions through nonlinear layers relies on one of the following (1) Monte Carlo sampling (2) local linearization of the nonlinearities or (3) direct derivation (or approximation) of the first two moments of output distributions under the assumption of zero correlation. The first is plausible thanks to powerful tensor calculation with GPUs, but is prone to undersampling, especially in high dimensional settings. The second allows propagating full covariance matrices, but introduces errors from ignoring higher order moments' contribution to the covariance calculation. The last one introduces errors from ignoring the correlations' contributions to the variance calculation.

To address these shortcomings, we here postulate a generalized framework of using a stochastic polynomial expansion as a surrogate nonlinearity, and derive the closed-form solutions of the mean, covariance, and cross-covariance of propagated multi-variate distributions, for seven nonlinearities that are ubiquitous in modern ML network constructs. This is achieved with a computational complexity of $\mathcal{O}(n^2)$, comparable to that of first-order Taylor expansion. Our methodology is inspired by stochastic linearization (SL), which uses expected value of the first derivative as gain and mean of output as bias, or $\hat{Y} = \mathbb{E}[\nabla_x f(X)] \circ (X - \mathbb{E}[X]) + \mathbb{E}[f(X)]$. Stochastic linearization minimizes mean square of the residual (Booton, 1953; Kazakov, 1954), and has been used in the context of feedback control systems (see, e.g., (Ching et al., 2010; Elishakoff and Crandall, 2017)).

2 Theory

To ensure consistency and clarity, we explicitly define the notation and convention used throughout this paper. All vectors are vertical. All instances of the $\circ$ operation denote element-wise (Hadamard) operations, including both element-wise multiplication and exponentiation. For example, given a vector $\mathbf{X}$, we define $\mathbf{X}^{\circ 2} = \mathbf{X} \circ \mathbf{X}$, whereas $\mathbf{X}^2 = \mathbf{X}\mathbf{X}^\top$, which represents the outer product. This convention extends to differntial operators as well. Specifically, the element-wise second derivative of a function f is given by:

$$\nabla^{\circ 2} f = \left(\frac{\partial^2 f}{\partial x_1^2} \ , \ \cdots \ , \ \frac{\partial^2 f}{\partial x_n^2} \right)^\top = \operatorname{diag}\left(\nabla^2 f\right)$$

Let $\mathbf{X} \coloneqq (\mathrm{X}_1, \mathrm{X}_2, \cdots, \mathrm{X}_\mathrm{n})^\top$ be a Gaussian random vector following $\mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$, and $\boldsymbol{\Xi} \coloneqq \mathbf{X} - \mathbb{E}[\mathbf{X}]$. Let $\bar{f}(\cdot)$ be a smooth, univariate function. We define the vector function $f(\mathbf{X}) = (\bar{f}(\mathrm{X}_1), \bar{f}(\mathrm{X}_2), \cdots, \bar{f}(\mathrm{X}_\mathrm{n}))^\top$. Here,

we have in mind that $f(\cdot)$ describes the activation function at the output of a feedforward layer. Now, let us define a set of i.i.d. surrogate distributions $\boldsymbol{\Xi}_{(s)} \sim \mathcal{N}(\mathbf{0}, \tilde{\boldsymbol{\Sigma}})$, $s = 1, 2, ..., S$.

We now propose the central construct in this paper, the **pseudo-Taylor polynomial expansion (PTPE)** of $f(X)$ as:

$$g(\mathbf{X}) = \mathbb{E}\left[f(\mathbf{X})\right] + \sum_{s=1}^{S} \frac{\mathbb{E}\left[\nabla_{\mathbf{x}}^{\circ s} f(\mathbf{X})\right]}{s!} \circ \left(\boldsymbol{\Xi}_{(s)}^{\circ s} - \mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\right]\right) \tag{1}$$

We postulate and later show that this expansion provides a tractable and accurate approximation of $f(X)$ for the purposes of propagating uncertainty through feedforward network architectures.

In the above, we use the form of a Taylor polynomial expansion to describe the behavior (in expectation) of the function $f(\mathbf{X})$ subject to the stochastic input $\mathbf{X}$. The choice of the i.i.d. surrogate polynomials, $\{\mathbf{1}, \boldsymbol{\Xi}_{(1)}, \boldsymbol{\Xi}_{(2)}^{\circ 2}, \boldsymbol{\Xi}_{(3)}^{\circ 3}, \cdots\}$, is made to simplify the ensuing derivations. Note that if taking only the first two terms, this expansion is equivalent to stochastic linearization, because $\boldsymbol{\Xi}_{(1)} - \mathbb{E}[\boldsymbol{\Xi}_{(1)}]$ is equivalent to $\mathbf{X} - \mathbb{E}[\mathbf{X}]$. It is straightforward to observe that $g(\mathbf{X})$ has the same first moment as $f(\mathbf{X})$ because all terms after the first one are designed to have zero mean. In the following, we will provide empirical evidence that the second moment is well-captured for many common activation functions.

First, we derive the solution for covariance and cross-covariance using the proposed stochastic polynomial expansion.

Lemma 1. *Define*

$$\boldsymbol{A}_0 = \mathbb{E}\left[f(\mathbf{X})\right] \qquad \boldsymbol{A}_1 = \frac{\mathbb{E}\left[\nabla_{\mathbf{x}} f(\mathbf{X})\right]}{1!} \qquad \boldsymbol{A}_2 = \frac{\mathbb{E}\left[\nabla_{\mathbf{x}}^{\circ 2} f(\mathbf{X})\right]}{2!} \qquad \cdots$$

Then, the covariance matrix of $g(\mathbf{X})$ is

$$\Sigma_{g(\mathbf{X})} = \sum_{s=1}^{S} \boldsymbol{A}_s \circ \left(\mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s} \; \boldsymbol{\Xi}_{(s)}^{\circ s \top}\right] - \mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\right] \mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\right]^{\top}\right) \circ \boldsymbol{A}_s^{\top} \tag{2}$$

for an S-th order expansion. For $S = 3$,

$$\begin{aligned} \Sigma_{g(\mathbf{X})} = & \boldsymbol{A}_1 \circ \tilde{\boldsymbol{\Sigma}} \circ \boldsymbol{A}_1^{\top} + \\ & \boldsymbol{A}_2 \circ \left(2\tilde{\boldsymbol{\Sigma}}^{\circ 2}\right) \circ \boldsymbol{A}_2^{\top} + \\ & \boldsymbol{A}_3 \circ \left[6\tilde{\boldsymbol{\Sigma}}^{\circ 3} + 9 \; \mathrm{diag}(\tilde{\boldsymbol{\Sigma}}) \circ \tilde{\boldsymbol{\Sigma}} \circ \mathrm{diag}(\tilde{\boldsymbol{\Sigma}})^{\top}\right] \circ \boldsymbol{A}_3^{\top} \end{aligned}$$

Proof. The expected values can be solved using central moments of Gaussian distributions and Isserlis' theorem. All power operations are element-wise. Note that since $\boldsymbol{A}_s$ are n dimensional vector, and all the power and product operations are element-wise, the complexity of calculating covariance is $\mathcal{O}(n^2)$. For detailed derivation, see Appendix A.1.2. □

It is useful to note that an addition (residual or recurrent) layer sums the activation of two (or more) layers, e.g. $\mathbf{X}$ and $g(\mathbf{Y})$. In thise case, the covariance of $\mathbf{X} + g(\mathbf{Y})$ is the sum of their covariances and cross-covariances, i.e.

$$\boldsymbol{\Sigma}_{\mathbf{X}+g(\mathbf{Y})} = \boldsymbol{\Sigma}_{\mathbf{X}} + \boldsymbol{\Sigma}_{g(\mathbf{Y})} + \boldsymbol{\Sigma}_{\mathbf{X}g(\mathbf{Y})} + \boldsymbol{\Sigma}_{g(\mathbf{Y})\mathbf{X}}$$

It it thus helpful to postulate an additional lemma for the purpose of calculating covariance after addition.

Lemma 2. *Let* $\mathbf{Y} \coloneqq (\mathrm{Y}_1, \mathrm{Y}_2, \cdots, \mathrm{Y}_n)^\top$ *be another Gaussian random vector that is cross-correlated to* $\mathbf{X}$ *with* $\boldsymbol{\Sigma}_{\mathbf{YX}}$*, and* $\boldsymbol{\Omega} \coloneqq \mathbf{Y} - \mathbb{E}[\mathbf{Y}]$*. Then, the cross-covariance matrix between* $\mathbf{Y}$ *and* $\mathbf{Z} \coloneqq g(\mathbf{X})$ *is*

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{YZ}} &= \sum_{t=1,t\ \mathit{is}\ \mathit{odd}}^{S} \boldsymbol{A}_t^\top \circ \mathbb{E}\left[\boldsymbol{\Omega}\ \boldsymbol{\Xi}_{(t)}^{\circ t\top}\right] \\ \boldsymbol{\Sigma}_{\mathbf{ZY}} &= \sum_{s=1,s\ \mathit{is}\ \mathit{odd}}^{S} \boldsymbol{A}_s \circ \mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\ \boldsymbol{\Omega}^\top\right] \end{aligned} \tag{3}$$

for an S*-th order expansion. For* $S = 3$*,*

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{YZ}} &= \boldsymbol{A}_1^\top \circ \boldsymbol{\Sigma}_{\mathbf{YX}} + 3\boldsymbol{A}_3^\top \circ \boldsymbol{\Sigma}_{\mathbf{YX}} \circ \mathrm{diag}(\boldsymbol{\Sigma}_{\mathbf{X}})^\top \\ \boldsymbol{\Sigma}_{\mathbf{ZY}} &= \boldsymbol{A}_1 \circ \boldsymbol{\Sigma}_{\mathbf{XY}} + 3\boldsymbol{A}_3 \circ \boldsymbol{\Sigma}_{\mathbf{XY}} \circ \mathrm{diag}(\boldsymbol{\Sigma}_{\mathbf{X}}) \end{aligned}$$

Proof. The expected value can be calculated using Isserlis' theorem. Note that this term is nonzero only if t and s are odd. For details of derivation, see Appendix A.1.3. □

With these results, to find the covariance of the output of a nonlinear layer, assuming the input follows a multi-variate normal distribution, one needs to derive the coefficients of the PTPE, i.e., $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, etc., for the nonlinearity of interest. Note that these coefficients only depend on mean $\tilde{\boldsymbol{\mu}}$ and variance $\tilde{\boldsymbol{\sigma}}^{\mathbf{2}} = \mathrm{diag}(\tilde{\boldsymbol{\Sigma}})$, not correlations, rendering the computational complexity $\mathcal{O}(n)$. We briefly discuss some of the techniques we adopted to solve for these polynomial coefficients and list the final results in Table Table 1 and Table 4. For detailed derivation for all nonlinearities, see Appendix A.2 - A.7.

Tanh, Sigmoid, and Softplus. Because the integral $\int \nabla\mathbf{tanh}(x)p(x)dx$ is not tractable analytically, so we make a further approximation by substituting **tanh** with the error function which is very similar but more tractable. Specifically, we propose

$$\mathbf{tanh}(x) \approx \frac{1}{p}\sum_{j=1}^{p} \mathbf{erf}[\gamma_j x]$$

where $\{\gamma_1, \cdots, \gamma_p\}$ is a set of scaling factors obtained by numerical optimization (see Eq.7 in Appendix), and the relationship between approximation accuracy and the number of scaling factors is discussed in **??**. The error function is defined as

$$\mathbf{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$$

Then, the integral $\int \nabla\mathbf{tanh}(x)p(x)dx$ can be approximated as $\frac{1}{p}\sum_{j=1}^{p}\int \nabla\mathbf{erf}(\gamma_j x)p(x)dx$, which is tractable analytically. The higher order derivatives of the error function are simply derivatives of Gaussian functions $\boldsymbol{\varphi}(x)$, which are related to Hermite polynomials $\mathbf{H}_s(x)$ through

$$\frac{d^s}{dx^s}\left[\frac{1}{\sigma}\boldsymbol{\varphi}\left(\frac{x}{\sigma}\right)\right] = \left(\frac{-1}{\sqrt{2\sigma^2}}\right)^s \mathbf{H}_s\left(\frac{x}{\sqrt{2\sigma^2}}\right)\frac{1}{\sigma}\boldsymbol{\varphi}\left(\frac{x}{\sigma}\right) \tag{4}$$

where

$$\mathbf{H}_0(x) = 1 \qquad \mathbf{H}_1(x) = 2x \qquad \mathbf{H}_2(x) = 4x^2 - 2 \qquad \cdots$$

We show the pseudo-Taylor coefficients are convolutions of Gaussian derivatives and Gaussian pdf, which are analytically tractable (see Appendix A.2). We used a similar treatment for **sigmoid** function (see Appendix A.3). Using error functions as a approximation was first suggested by Spiegelhalter and Lauritzen, but we use a linear combination, which is easily parallelizable and enhanced approximation accuracy. The derivation for **softplus** can reuse the results of **sigmoid**, because the derivative of **softplus** is just **softplus** with a scaling factor β (A.4).

ReLU and LeakyReLU. It is obvious that we cannot apply our method directly on **ReLU**, because it is not continuously differentiable. Hence, we modified the results for **softplus** at the limit of $\beta \to \infty$, considering the relationship (A.5)

$$\lim_{\beta\to\infty} \frac{1}{\beta}\mathbf{log}\left(1+e^{\beta x}\right) = \max\{0, x\}$$

Similarly, leaky ReLU and any piece-wise linear activation function can be described as a combination of ReLU functions with different scaling, shifting, and/or mirroring.

GELU and SiLU. The derivatives of GELU can be expressed using derivatives of Gaussian cdf $\mathbf{\Phi}(x)$

$$\frac{\partial^s}{\partial x^s}\mathbf{GELU}(x) = s\frac{\partial^{s-1}}{\partial x^{s-1}}\mathbf{\Phi}(x) + x\frac{\partial^s}{\partial x^s}\mathbf{\Phi}(x)$$

The expected value of the GELU derivative involves integrating the product of Hermite polynomials and Gaussian functions, which is analytically tractable (A.6). Using normal cdf to approximate a **sigmoid** function, the derivations for a SiLU function becomes similar to that of the GELU (A.7).

Before presenting the final expression, we introduce a few element-wise operators to simplify the formulation. Let $\tilde{\boldsymbol{\sigma}}^2 = \mathrm{diag}(\tilde{\boldsymbol{\Sigma}})$ denote the input variance and $\acute{\sigma}_j^2$ represent a constant dependent on the nonlinearity (see Table 1 column 3). Defining $\hat{\boldsymbol{\sigma}}_j^2 = \tilde{\boldsymbol{\sigma}}^2 + \acute{\sigma}_j^2$, we then have

$$\mathcal{L}(\tilde{\mu}\,;\hat{\sigma}_j) \coloneqq \frac{1}{\hat{\sigma}_j}\,\boldsymbol{\varphi}\left(\frac{x}{\hat{\sigma}_j}\right)\Big|_{x=\tilde{\mu}} \qquad \text{Likelihood of observing } x=\tilde{\mu} \text{ if } X\sim\mathcal{N}(0,\hat{\sigma}_j^2)$$

$$\mathcal{F}(\tilde{\mu}\,;\hat{\sigma}_j) \coloneqq \mathbf{\Phi}\left(\frac{x}{\hat{\sigma}_j}\right)\Big|_{x=\tilde{\mu}} \qquad \text{Cumulative likelihood of observing } x\le\tilde{\mu}$$

$$\mathcal{I}(\tilde{\mu}\,;\hat{\sigma}_j) \coloneqq \int_{-\infty}^{\tilde{\mu}} \mathbf{\Phi}\left(\frac{x}{\hat{\sigma}_j}\right)\mathrm{d}x = \tilde{\mu}\,\mathcal{F}(\tilde{\mu}\,;\hat{\sigma}_j) + \hat{\sigma}_j^2\,\mathcal{L}(\tilde{\mu}\,;\hat{\sigma}_j) \qquad \text{Expected value of excess } (\tilde{\mu}-X)$$

(only when $\tilde{\mu} > x$, and $X\sim\mathcal{N}(0,\hat{\sigma}_j^2)$)

We also define an element-wise derivative operator $\boldsymbol{\mathcal{D}}^s$ such that

$$\boldsymbol{\mathcal{D}}^s\boldsymbol{\mathcal{L}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) \coloneqq \nabla_{\boldsymbol{x}}^{\circ s}\boldsymbol{\mathcal{L}}(\boldsymbol{x}\,;\hat{\boldsymbol{\sigma}}_j)|_{\boldsymbol{x}=\tilde{\boldsymbol{\mu}}} = \left(\frac{-1}{\sqrt{2\hat{\boldsymbol{\sigma}}_j^2}}\right)^s \mathbf{H}_s\left(\frac{\tilde{\boldsymbol{\mu}}}{\sqrt{2\hat{\boldsymbol{\sigma}}_j^2}}\right)\boldsymbol{\mathcal{L}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) \qquad \dagger$$

Notice that

$$\boldsymbol{\mathcal{D}}\,\boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) = \boldsymbol{\mathcal{F}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j)$$
$$\boldsymbol{\mathcal{D}}^2\,\boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) = \boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{F}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) = \boldsymbol{\mathcal{L}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j)$$

Then, it becomes evident that all of these pseudo-Taylor coefficients $\boldsymbol{A}_s$ can be written as derivatives of $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\mathcal{L}}$ (Table. 1). This is a result of the Gaussian assumption and the reformulation of**tanh**and sigmoid.

3 Results

3.1 PTPE significantly improves estimation accuracy when exposed to higher input variance

As an initial empirical test and demonstration of concept, we applied PTPE to a single, univariate nonlinearity subject to a parameterized normally distributed input. We varied the input mean and variance and examined how the output mean and variance compared to those predicted by PTPE. For this comparison, the true output statistics were obtained through 10^7 Monte Carlo sampling across all input parameters. As expected, PTPE far outstrips Jacobian linearization, and this effect is prominent especially when input variance is high. With up-to third order PTPE, the estimated variance by our method is already very close to the ground truth (Fig. 1 col 4).

†All operations in the right hand side of the equation are element-wise.

Table 1: General solutions for the pseudo-Taylor coefficients. For notational simplicity, all the product, division, and power operations are element-wise. For expanded solutions of the first five coefficients, see Table 4 in the Appendix.

Nonlinearity	General Solution	$\hat{\boldsymbol{\sigma}}_j^2 = \tilde{\boldsymbol{\sigma}}^2 + \acute{\sigma}_j^2$	Definition of γ_j
Tanh	$\boldsymbol{A}_s = \frac{1}{s!}\frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\mathcal{D}}^s \left[2\boldsymbol{\mathcal{F}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) - 1\right]$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2}$	$\tanh(x) \approx \frac{1}{p}\sum_{j=1}^{p} \mathrm{erf}(\gamma_j x)$
Sigmoid	$\boldsymbol{A}_s = \frac{1}{s!}\frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\mathcal{D}}^s \boldsymbol{\mathcal{F}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j)$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2}$	$\mathrm{sigmoid}(x) \approx \frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\Phi}\left(\sqrt{2\gamma_j^2}\, x\right)$
Softplus	$\boldsymbol{A}_s = \frac{1}{s!}\frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\mathcal{D}}^s \boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j)$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2\beta^2}$	$\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{softplus}(x;\beta) \approx \frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\Phi}\left(\sqrt{2\gamma_j^2\beta^2}\, x\right)$
ReLU	$\boldsymbol{A}_s = \frac{1}{s!}\boldsymbol{\mathcal{D}}^s \boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\tilde{\boldsymbol{\sigma}})$	$\acute{\sigma}^2 = 0$	
LeakyReLU(θ)	$\boldsymbol{A}_s = \frac{1}{s!}\boldsymbol{\mathcal{D}}^s \left[\theta \boldsymbol{x} + \boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\tilde{\boldsymbol{\sigma}})\right]$	$\acute{\sigma}^2 = 0$	
GELU	$\boldsymbol{A}_s = \frac{1}{s!}\boldsymbol{\mathcal{D}}^s \left[\boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}) - \boldsymbol{\mathcal{L}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}})\right]$	$\acute{\sigma}^2 = 1$	
SiLU	$\boldsymbol{A}_s = \frac{1}{s!}\frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\mathcal{D}}^s \left[\boldsymbol{\mathcal{I}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j) - \acute{\sigma}_j^2 \boldsymbol{\mathcal{L}}(\tilde{\boldsymbol{\mu}}\,;\hat{\boldsymbol{\sigma}}_j)\right]$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2}$	same as Sigmoid

3.2 PTPE accurately quantifies uncertainty in canonical network architectures

To benchmark PTPE for uncertainty quantification in neural networks, we trained 9 residual neural networks (He et al. (2016)) with three depths (13, 33, and 65 layers) and 3 three typical nonlinearities (Tanh, ReLU, GELU) on CIFAR10 (Krizhevsky (2009)). We corrupted each input image with additive Gaussian noise to simulate noise in low light conditions (first type of corruption in Hendrycks and Dietterich (2019)), then compared the PTPE-predicted and reference (via 10^6 Monte Carlo sampling) output distributions. Four levels of corruption, with noise variance values of [1, 10, 100, 1000], were applied to RBG values ([0, 255]) of the input image. If z-scored, the corresponding noise variance scales are [1e-5, 1e-4. 1e-3, 1e-2]. The visualization of the corrupted images are shown in (Fig. 2). The layerwise application of PTPE is outlined in Algorithm 1 with accompanying pseudo code.

We measure the estimation accuracy of moments in three ways: the Euclidean distance from the reference mean to the predicted mean, $||\mu_{\text{est}} - \mu_{\text{ref}}||_2$, the Frobenius norm of the covariance residuals $||\boldsymbol{\Sigma}_{\text{est}} - \boldsymbol{\Sigma}_{\text{ref}}||_{\text{fro}}$, and the 2-Wasserstein distance (or Kantorovich-Rubinstein metric) between the reference and estimated distributions, assuming both distributions were Gaussian. This 2-Wasserstein distance is defined as

$$W_2 = \sqrt{||\mu_{\text{est}} - \mu_{\text{ref}}||_2^2 + \mathrm{trace}\left(\boldsymbol{\Sigma}_{\text{est}} + \boldsymbol{\Sigma}_{\text{ref}} - 2\left(\boldsymbol{\Sigma}_{\text{est}}^{1/2}\boldsymbol{\Sigma}_{\text{ref}}\boldsymbol{\Sigma}_{\text{est}}^{1/2}\right)^{1/2}\right)}$$

We summarize the results in Fig. 3, 8, and 9. Overall, the experimental results align with expectations: (1) Jacobian linearization degrades dramatically in moderate to high variance regime. (2) Direct derivation is not suitable for this task due to the assumption of independence, since the overlapping convolution kernels and residual layers introduce substantial correlation. (3) Introducing up-to the third order PTPE typically outperformes stochastic and Jacobian linearization by a large margin. We also compared using 4 scaling factors v.s. using 8, and the results showed no significant difference (Fig. 11), justifying our choice of four scaling factors rather than a larger number in this case.

3.3 PTPE addresses the limitations of DVI by incorporating non-piecewise-linear activations.

One major contribution of this work is to address the lack of accurate deterministic moment estimation for general nonlinearities in the field of variational inference. The method of deterministic variational inference

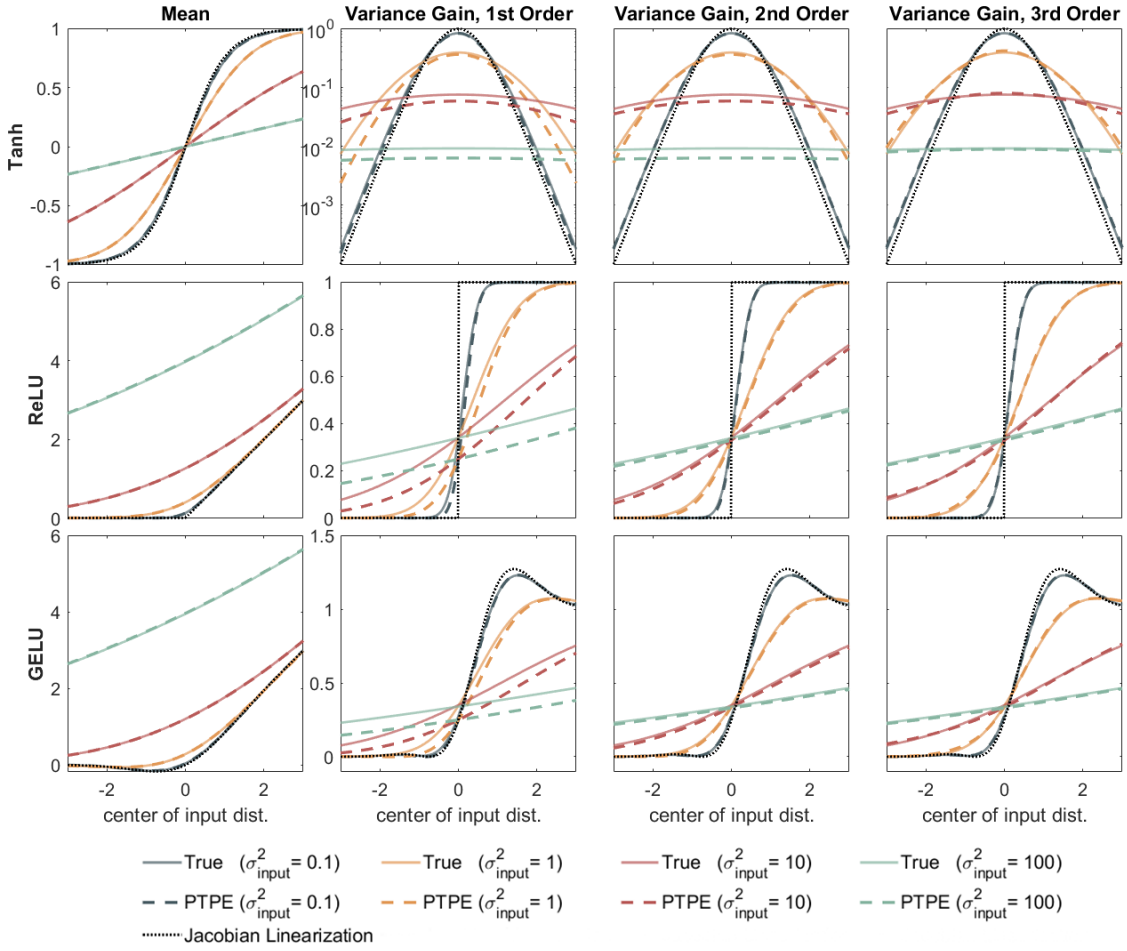

Figure 1: **Solid lines**: mean (column 1) and variance gain (output variance divided by input variance) (column 2-5) obtained by sampling 1e7 datapoints from Gaussian distributions with centers ranging from -3 to 3. **Dashed lines**: approximated mean and variance gain predicted with 1st, 2nd, and 3rd order pseudo Taylor polynomial expansion (column 2 - 4). Colors correspond to different input variances (blue: 0.1, yellow: 1, red: 10, green: 100). **Dotted lines**: approximated mean and variance gain using Jacobian linearization (first order deterministic Taylor expansion around input mean, e.g. Petersen et al. (2024)). For other nonlinearities, see Fig. 7.

(DVI) proposed by Wu et al. shares the same goal, which is to find a deterministic method to approximate moments in neural networks, thus eliminating gradient variance. However, closed form solutions of posterior mean and covariance were solved only for piecewise linear activations such as ReLU and Heaviside. We know $\text{Var}(f(X)) = \mathbb{E}[f(X)^2] - \mathbb{E}[f(X)]^2$, and the first term, $\mathbb{E}[f(X)^2] = \int f(x)^2 p_X(x) dx$, becomes arduous to solve for more complex nonlinearities $f(\cdot)$ †. Our approach circumvents this issue by taking derivatives inside the integrals, which provides tractability for general nonlinearities.

We provide additional context and quantification by replacing the forward-passing functions of Gaussian moments in DVI with PTPE and conducting regression experiments on eight UCI datasets. Following the methodology suggested by Hernández-Lobato and Adams, we search over MLPs with up to four layers

† In numerical analysis, Gauss-Hermite quadrature is a well-established method used to approximate integrals of the same type, serving as the foundation for cubature Kalman filtering (Arasaratnam and Haykin, 2009; Särkkä, 2013). While it is fundamentally a sampling-based method, it selects sampling (quadrature) points using Hermite polynomials to improve efficiency. In Section A.13, we discuss the key differences between PTPE and Gauss-Hermite quadrature and demonstrate through experiments that PTPE, which does not rely on sampling, converges significantly faster.

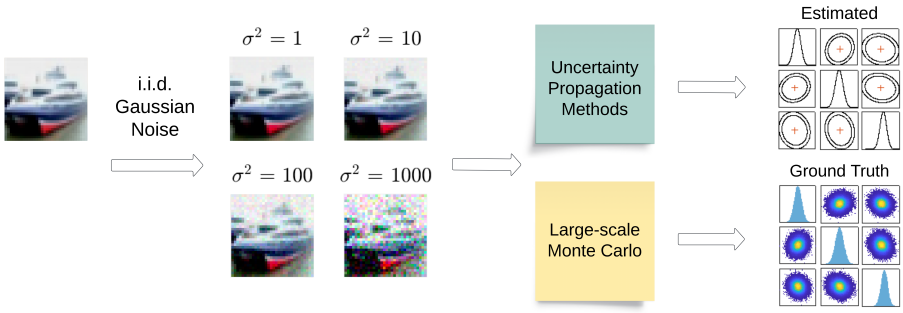

Figure 2: Schematic of experiment setup. We inject i.i.d. Gaussian noise to input image to simulate sensor noises, then compare the estimated output distribution to the ground truth obtained by large-scale simulation (sampling 10^6 noisy images).

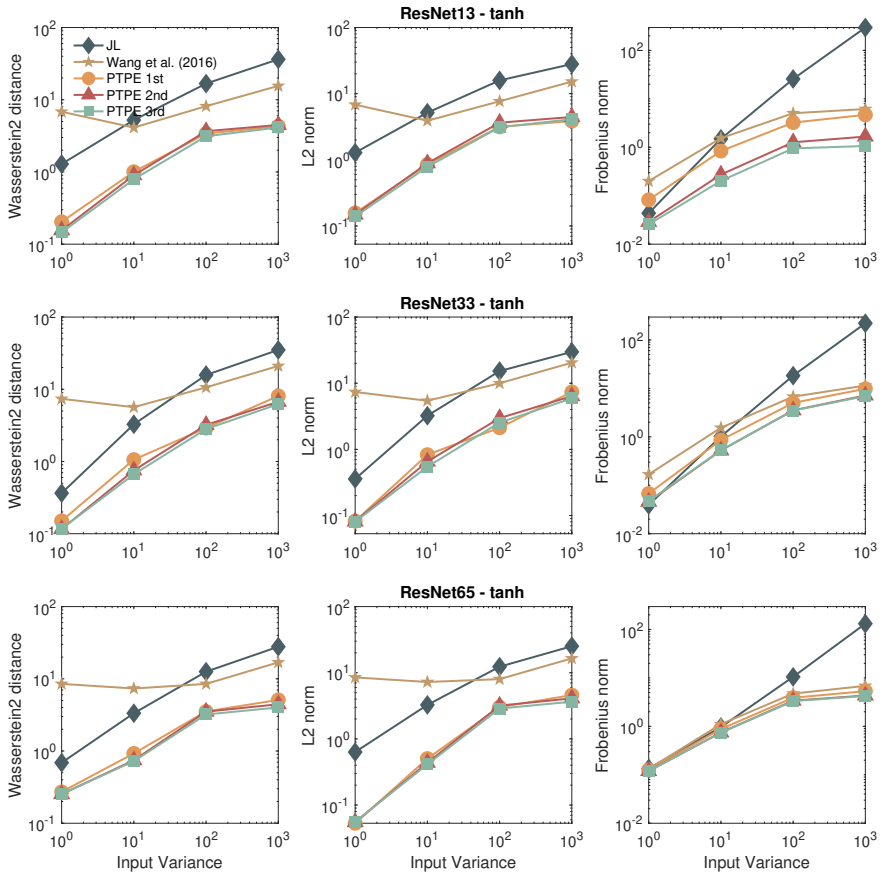

Figure 3: Estimation accuracies of expectation propagation methods evaluated on residual networks of three different depth, and four different input variance. The smaller the better. We chose Petersen et al. (2024) as an example of JL and Wang et al. (2016) as an alternative **tanh** approximation. For non-linearities ReLU and GELU, see Fig. 8 and 9 in the Appendix.

Table 2: Averaged test performance in RMSE. The smaller the better. The best value is highlighted in bold

Dataset	Concrete Strength	Energy Efficiency	Kin8nm	Naval Propulsion	Power Plant	Protein Structure	Wine Quality (Red)	Yacht Hydro-dynamics
Data points	1030	768	8192	11934	9568	45730	1599	308
Dimensions	8	8	8	16	4	9	11	6
MCVI	7.128 ± 0.123	2.646 ± 0.081	0.099 ± 0.001	0.005 ± 0.001	4.327 ± 0.035	4.842 ± 0.031	0.646 ± 0.008	6.887 ± 0.675
PBP	5.667 ± 0.093	1.804 ± 0.048	0.098 ± 0.001	0.006 ± 0.000	4.124 ± 0.035	4.732 ± 0.013	0.635 ± 0.008	1.015 ± 0.054
Dropout	5.23 ± 0.12	1.66 ± 0.04	0.10 ± 0.00	0.01 ± 0.00	4.02 ± 0.04	4.36 ± 0.01	**0.62 ± 0.01**	1.11 ± 0.09
PTPE ReLU	5.196 ± 0.206	0.615 ± 0.024	0.072 ± 0.001	**0.003 ± 0.000**	3.925 ± 0.025	4.445 ± 0.042	0.633 ± 0.010	0.640 ± 0.057
PTPE GELU	**5.068 ± 0.153**	**0.570 ± 0.021**	**0.071 ± 0.000**	0.004 ± 0.000	**3.915 ± 0.024**	4.415 ± 0.043	0.634 ± 0.010	**0.623 ± 0.049**
PTPE Tanh	5.574 ± 0.148	0.580 ± 0.022	0.076 ± 0.001	0.005 ± 0.000	4.073 ± 0.028	**4.364 ± 0.036**	0.628 ± 0.010	1.678 ± 0.193

Table 3: Averaged test performance in average log-likelihood. The larger the better. The best value is highlighted in bold, and the second best is underlined.

Dataset	Concrete Strength	Energy Efficiency	Kin8nm	Naval Propulsion	Power Plant	Protein Structure	Wine Quality (Red)	Yacht Hydro-dynamics
MCVI	-3.391 ± 0.017	-2.391 ± 0.029	0.897 ± 0.010	3.734 ± 0.116	-2.890 ± 0.010	-2.992 ± 0.006	-0.980 ± 0.013	-3.439 ± 0.163
PBP	-3.161 ± 0.019	-2.042 ± 0.019	0.896 ± 0.006	3.731 ± 0.006	-2.837 ± 0.009	-2.973 ± 0.003	-0.968 ± 0.014	-1.634 ± 0.016
Dropout	-3.04 ± 0.02	-1.99 ± 0.02	0.95 ± 0.01	3.80 ± 0.01	-2.80 ± 0.01	-2.89 ± 0.01	-0.93 ± 0.01	-1.55 ± 0.03
DVI	-3.06 ± 0.01	-1.01 ± 0.06	1.13 ± 0.00	**6.29 ± 0.04**	-2.80 ± 0.00	-2.85 ± 0.01	**-0.90 ± 0.01**	-0.47 ± 0.03
PTPE ReLU	**-3.010 ± 0.037**	-1.045 ± 0.044	1.251 ± 0.009	5.751 ± 0.086	-2.789 ± 0.007	-2.821 ± 0.024	-0.966 ± 0.029	-0.910 ± 0.044
PTPE GELU	-3.092 ± 0.056	-0.789 ± 0.039	1.278 ± 0.007	5.858 ± 0.135	**-2.780 ± 0.006**	-2.801 ± 0.019	-0.982 ± 0.029	**-0.236 ± 0.052**
PTPE Tanh	-3.159 ± 0.039	-0.827 ± 0.043	1.234 ± 0.008	6.050 ± 0.028	-2.825 ± 0.006	-2.802 ± 0.013	-0.939 ± 0.016	-0.699 ± 0.067
LL Tanh	-3.07 ± 0.07	**-0.65 ± 0.05**	**1.29 ± 0.01**	**6.29 ± 0.19**	-2.79 ± 0.01	**-2.79 ± 0.00**	-0.98 ± 0.01	-0.92 ± 0.03

containing 50 hidden units (100 for the larger Protein Structure dataset) and report the best test performance. We randomly set aside 10% of the data as test samples, and the error bars reflect the results from 20 random splits. Details on the implementation, hyperparameters, and training results are available at `https://anonymous.4open.science/r/Stochastic_Polynomial_Expansion-0BEB/`.

We evaluate RMSE and log-likelihood on the held-out data and summarize the results in Tables 2 and 3. For comparison, we include reported statistics (where available) from Monte Carlo Variational Inference (MCVI) (Graves, 2011a), Probabilistic Backpropagation (PBP) (Hernández-Lobato and Adams, 2015), Dropout (Gal and Ghahramani, 2016), DVI (Wu et al., 2019), and Linearized Laplace (MacKay, 1992a; Foong et al., 2019). The results indicate that PTPE-DVI achieves competent accuracy, demonstrating the effectiveness of PTPE.

3.4 PTPE enabled Variational Autoencoder demonstrates improved performance with $\mathcal{O}(n)$ complexity.

Since PTPE is fundamentally designed to accurately propagate Gaussian moments through nonlinearities, it can be incorporated into various applications, one of which is the decoder of a Variational Autoencoder (VAE) (Kingma and Welling, 2014). In the decoding stage, a VAE propagates the Gaussian means and variances encoded by the encoder through layers of nonlinearity in the decoder. The original model, which we refer to as the "vanilla VAE," accomplishes this by propagating Monte Carlo samples. Instead, we replace the decoder with PTPE while keeping the trainable parameters unchanged and train the model to reconstruct MNIST handwritten digits (LeCun et al., 1998). As shown in Fig. 4, the PTPE-enabled VAE achieves a higher Evidence Lower Bound (ELBO) and improved reconstruction accuracy compared to the vanilla VAE.

A key bottleneck in applying many well-established Bayesian methods to VAEs is scalability. Many Bayesian approaches require sampling, which suffers from curse of dimensionality. PTPE offers an alternative solution: the PTPE-VAE shown in Fig. 4 propagates only the diagonal of the covariance, resulting in a computational complexity of $\mathcal{O}(n)$. Moreover, the training procedure remains identical to that of the vanilla VAE. Details on the implementation, hyperparameters, and training results are available at `https://anonymous.4open.science/r/Stochastic_Polynomial_Expansion-0BEB/`.

4 Discussion

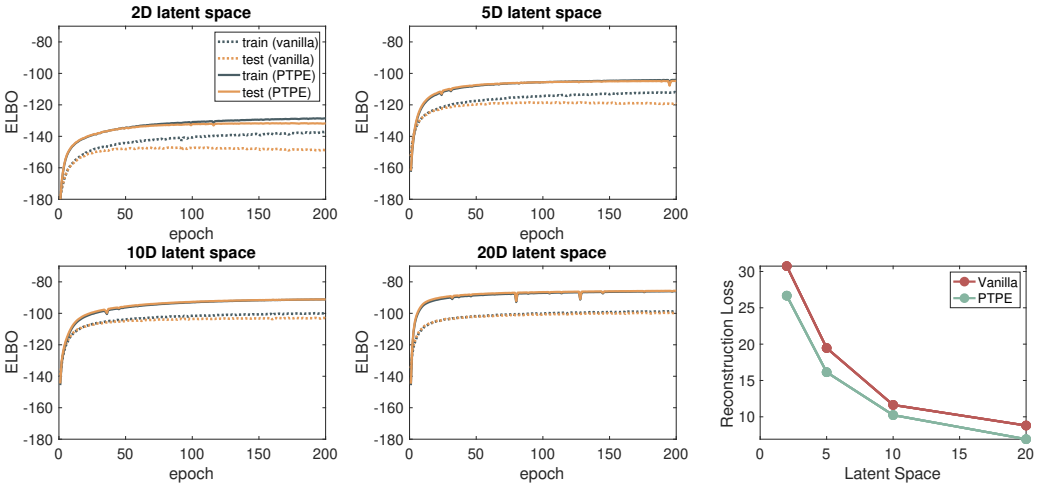

Figure 4: (Left four) Training and testing objective, measured as ELBO, for both vanilla and PTPE implemented VAE. (Right) Reconstruction loss (MSE) of the two types of models.

One immediate potential limitation of PTPE is its reliance on the assumption that inputs are Gaussian. It has been well-established that at the limit of infinite width, a deep neural network with Gaussian input is equivalently a Gaussian process (Neal (1994); Williams (1997); de G. Matthews et al. (2018); Lee et al. (2018); Gao et al. (2023)), and a similar phenomenon is also reported in Bayesian neural networks with Gaussian weights (Goulet et al. (2021); Nguyen and Goulet (2022)). Based on this observation, we assume a "wide enough" neural network will have approximate Gaussianity in each layer, so that the error of using variational Gaussian distributions to approximate layer-wise distributions becomes negligible. We verify this assumption through simulation (see e.g., Fig. 5).

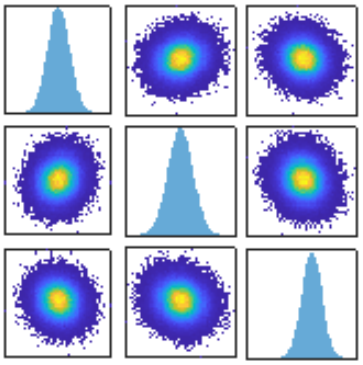

Figure 5: Empirical distributions (of the first three units) before the final softmax layer of a residual neural net trained on CIFAR10. This network has 13 ReLU layers. The distributions are obtained by Monte Carlo sampling 10^5 images with additive Gaussian noises. For distributions of all 10 units, see Fig. 6 in Appendix.

In this paper, we focus on the propagation of Gaussian distributions. This choice is due to their prevalence in machine learning and their convenient property of being Lévy alpha-stable, meaning a linear combination of Gaussian random variables remains Gaussian. This makes Gaussian distributions pertinent to our objectives. Consequently, our method could potentially be extended to other types within the Lévy alpha-stable family. For instance, Petersen et al. demonstrated the propagation of Cauchy distributions through neural networks. A more comprehensive survey is provided in Wang et al. (2016), where the authors examined the propagation of exponential family distributions (including Beta, Rayleigh, Gamma, Poisson, and Gaussian), though this requires more intricate derivations.

A strength of PTPE is its generality. As mentioned in the introduction, several immediate motivating use-cases are in the training of robust networks including probabilistic network models. Furthermore, our proposed method may also find application in safety-critical engineering systems that require estimates on uncertainty. Recently, researchers combined an LSTM and Kalman filtering to monitor the states of plasma inside a nuclear fusion device Pavone et al. (2023). The Kalman filter, by construction, requires statistics on the output of the LSTM in order to generate control signals. Such statistics were generated by using a probabalistic architcture within the LSTM, i.e., where parameters are specified by a learned distribution. PTPE provides a potential alternative path for such problems (we discuss in A.13), but enabling uncertainty propagation through deterministic learned architectures.

5 Conclusion

In this article, we developed a stochastic polynomial expansion approach, PTPE, to perform uncertainty propagation in neural networks. Our method offers significant advantages in accurately propagating the full covariance matrix of an input distribution compared to state-of-the-art methods, without substantially sacrificing computational efficiency. We derived analytical solutions for the first two moments of the output distributions for seven commonly used nonlinearities, demonstrating remarkable accuracy in predicting univariate mean and variance, particularly under high uncertainty. Additionally, we assessed its multivariate accuracy in deep residual neural networks trained on image categorization tasks. By incorporating up to third-order polynomial expansion, our method generally outperformed others, except in scenarios with minimal uncertainty in which the performance of competing methods is comparable. Overall, our proposed method provides a tractable framework for solving uncertainty propagation problems. It can potentially be effectively applied in various domains, including adversarial training, Bayesian inference, generative models, and safety-critical applications, offering a versatile tool for enhancing the reliability and robustness of neural networks.

References

A. H. Abdelaziz, S. Watanabe, J. R. Hershey, E. Vincent, and D. Kolossa. Uncertainty propagation through deep neural networks. In *Interspeech 2015*, Dresden, Germany, September 2015.

I. Arasaratnam and S. Haykin. Cubature kalman filters. *IEEE Transactions on Automatic Control*, 54(6): 1254–1269, 2009. doi: 10.1109/TAC.2009.2019800.

V. Beiu, J. Peperstraete, J. Vandewalle, and R. Lauwereins. Vlsi complexity reduction by piece-wise approximation of the sigmoid function. 01 1994.

R. C. Booton. The analysis of nonlinear control systems with random inputs. In *Proceedings, symposium on nonlinear circuit analysis*, volume 2, pages 369–391, 1953. see also IEEE Transactions on Circuit Theory, CT-1 (1954), pp 9–18.

P. Bromiley. Products and convolutions of gaussian probability density functions. Technical report, Tina-Vision Memo, 2003. URL `http://in.ruc.edu.cn/wp-content/uploads/2016/09/The-product-and-convolution-of-guassian-distributions.pdf`.

S. Ching, Y. Eun, C. Gokcek, P. Kabamba, and S. Meerkov. Quasilinear control: Performance analysis and design of feedback systems with nonlinear sensors and actuators. *Quasilinear Control: Performance Analysis and Design of Feedback Systems with Nonlinear Sensors and Actuators*, 01 2010. doi: 10.1017/CBO9780511976476.

W. J. Cody. Rational chebyshev approximations for the error function. *Mathematics of Computation*, 23 (107):631–637, 1969. ISSN 00255718, 10886842.

A. G. de G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. *ArXiv*, abs/1804.11271, 2018.

D. Dera, N. Bouaynaya, G. Rasool, R. Shterenberg, and H. Fathallah-Shaykh. Premium-cnn: Propagating uncertainty towards robust convolutional neural networks. *IEEE Transactions on Signal Processing*, PP: 1–1, 07 2021. doi: 10.1109/TSP.2021.3096804.

I. Elishakoff and S. H. Crandall. Sixty years of stochastic linearization technique. *Meccanica*, 52(1):299–305, Jan 2017. ISSN 1572-9648. doi: 10.1007/s11012-016-0399-x.

A. Y. K. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner. 'in-between' uncertainty in bayesian neural networks. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019. arXiv:1906.11537.

B. J. Frey and G. E. Hinton. Variational learning in nonlinear gaussian belief networks. *Neural Computation*, 11(1):193–213, Jan 1999.

G. Fubini. Sugli integrali multipli. *Atti della Reale Accademia dei Lincei*, 16:608–614, 1907.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

M. S. Gandhi, K. Lee, Y. Pan, and E. A. Theodorou. Propagating uncertainty through the tanh function with application to reservoir computing. *ArXiv*, abs/1806.09431, 2018.

T. Gao, X. Huo, H. Liu, and H. Gao. Wide neural networks as gaussian processes: Lessons from deep equilibrium models. *arXiv preprint arXiv:2305.00001*, 2023.

J. Gast and S. Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

J.-A. Goulet, L.-H. Nguyen, and S. Amiri. Tractable approximate gaussian inference for bayesian neural networks. *Journal of Machine Learning Research*, 2021.

I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, 8th edition edition, 2015.

A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356, 2011a.

A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2348–2356, 2011b.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

J. M. Hernández-Lobato and R. P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1861–1869, 2015.

G. E. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory (COLT)*, pages 5–13. ACM, 1993.

L. Isserlis. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, 12(1/2):134–139, 1918. ISSN 00063444, 14643510.

S. Janson. *Gaussian Hilbert Spaces / Svante Janson.* Cambridge Tracts in Mathematics ; no. 129. Cambridge University Press, Cambridge, 1997. ISBN 9780511526169.

P. Jylänki, A. Nummenmaa, and A. Vehtari. Expectation propagation for neural networks with sparsity-promoting priors. *Journal of Machine Learning Research*, 15:1849–1901, 2014.

I. E. Kazakov. Approximate method of statistical investigations of nonlinear systems. In *Proceedings, Voenno-Vozdushnaya Akademia imeni N.I. Zhukuvskogo*, volume 394, pages 1–52, 1954.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992a.

D. J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5): 720–736, 1992b. doi: 10.1162/neco.1992.4.5.720.

J. V. Michalowicz, J. M. Nichols, F. Bucholtz, and C. C. Olson. A general isserlis theorem for mixed-gaussian random variables. *Statistics & Probability Letters*, 81:1233–1240, 2011.

R. M. Neal. Priors for infinite networks. Technical Report CRG-TR-94-1, University of Toronto, 1994.

E. W. Ng and M. Geller. A table of integrals of the error functions. *Journal of Research of the National Bureau of Standards - B. Mathematical Sciences*, 73B(1):1–20, January-March 1969. doi: 10.6028/jres.073B.001. Paper 73Bl-281.

L.-H. Nguyen and J.-A. Goulet. Analytically tractable hidden-states inference in bayesian neural networks. *Journal of Machine Learning Research*, 2022.

A. Pavone, A. Merlo, S. Kwak, and J. Svensson. Machine learning and bayesian inference in nuclear fusion research: an overview. *Plasma Physics and Controlled Fusion*, 65(5), 2023. doi: 10.1088/1361-6587/acc60f.

F. Petersen, A. A. Mishra, H. Kuehne, C. Borgelt, O. Deussen, and M. Yurochkin. Uncertainty quantification via stable distribution propagation. In *The Twelfth International Conference on Learning Representations*, 2024.

A. Shekhovtsov and B. Flach. Feed-forward propagation in probabilistic neural networks with categorical and max layers. In *International Conference on Learning Representations (ICLR)*, 2019.

D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, 1990. doi: https://doi.org/10.1002/net.3230200507.

S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, Cambridge, UK, 2013. ISBN 978-1-107-03065-7.

H. Wang, X. Shi, and D. Y. Yeung. Natural-parameter networks: A class of probabilistic neural networks. In *Neural Information Processing Systems*, pages 118–126, 2016.

C. K. Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems*, pages 295–301, 1997.

A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt. Deterministic variational inference for robust bayesian neural networks. In *International Conference on Learning Representations*, 2019.

A Appendix

A.1 Derivation of mean, covariance, and cross-covariance propagated through a univariate nonlinear function

Revisit the definitions of notations.

$$
\begin{aligned}
\mathbf{U} &\quad \text{multivariate Gaussian input} \quad \sim \mathcal{N}^n(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
\boldsymbol{W} &\quad \text{weight matrix (constant)} \\
\boldsymbol{b} &\quad \text{bias vector (constant)} \\
\mathbf{X} &\quad = \boldsymbol{W}^\top \mathbf{U} + \boldsymbol{b} \quad \sim \mathcal{N}^n(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \\
\tilde{\boldsymbol{\mu}} &\quad = \boldsymbol{W}^\top \boldsymbol{\mu} + \boldsymbol{b} \\
\tilde{\boldsymbol{\Sigma}} &\quad = \boldsymbol{W}^\top \boldsymbol{\Sigma} \boldsymbol{W} \\
\boldsymbol{\Xi} &\quad = \mathbf{X} - \tilde{\boldsymbol{\mu}} \quad \sim \mathcal{N}^n(\mathbf{0}, \tilde{\boldsymbol{\Sigma}}) \\
\tilde{\boldsymbol{\sigma}}^2 &\quad = \operatorname{diag}\left(\tilde{\boldsymbol{\Sigma}}\right) \\
\{\gamma_1 \ , \ \dots \ , \ \gamma_p\} &\quad \text{positive scaling factor set obtained through numerical optimizations}
\end{aligned}
$$

All $\circ$ oprations are element-wise, including the Hadamard product and Hadamard exponentiation. However, for notationaly simplicity, all product, division, and power operations are element-wise starting from A.2.

In the context of machine learning, all non-linearities are applied element-wise – they are univarite. Thus, the off-diagonal entries of their Hessian matrices (second order partial derivatives) are zero, and it has similar effect on higher order partial derivatives. This makes PTPE on multivariate input easier to write down.

Given a smooth nonlinear function $\bar{f}(\cdot)$ of univariate random variables, define vector operation $f(\mathbf{X}) \coloneqq (\bar{f}(\mathrm{X}_1), \bar{f}(\mathrm{X}_2), \cdots, \bar{f}(\mathrm{X}_\mathrm{n}))^\top$. We define an approximation $g(\cdot)$, which stochastically expands $f(\cdot)$ under the Taylor scheme, such that $f(\cdot)$ and $g(\cdot)$ have approximately the same first and second moment. This expansion uses i.i.d. surrogate polynomials, $\{\mathbf{1}, \boldsymbol{\Xi}_{(1)}, \boldsymbol{\Xi}^{\circ 2}_{(2)}, \boldsymbol{\Xi}^{\circ 3}_{(3)}, \cdots\}$, and such choice reduces computational complexity of covariance, which is shown later.

$$
g\left(\mathbf{X}\right) = \mathbb{E}\left[f\left(\mathbf{X}\right)\right] + \frac{\mathbb{E}\left[\nabla_{\mathbf{x}} f\left(\mathbf{X}\right)\right]}{1!} \circ \left(\boldsymbol{\Xi}_{(1)} - \mathbb{E}\left[\boldsymbol{\Xi}_{(1)}\right]\right) + \frac{\mathbb{E}\left[\nabla^{\circ 2}_{\mathbf{x}} f\left(\mathbf{X}\right)\right]}{2!} \circ \left(\boldsymbol{\Xi}^{\circ 2}_{(2)} - \mathbb{E}\left[\boldsymbol{\Xi}^{\circ 2}_{(2)}\right]\right) + \cdots
$$

and denote

$$
\begin{aligned}
\boldsymbol{A}_0 &= \mathbb{E}\left[f\left(\mathbf{X}\right)\right] \\
\boldsymbol{A}_1 &= \frac{\mathbb{E}\left[\nabla_{\mathbf{x}} f\left(\mathbf{X}\right)\right]}{1!} \\
\boldsymbol{A}_2 &= \frac{\mathbb{E}\left[\nabla^{\circ 2}_{\mathbf{x}} f\left(\mathbf{X}\right)\right]}{2!} \\
\boldsymbol{A}_3 &= \frac{\mathbb{E}\left[\nabla^{\circ 3}_{\mathbf{x}} f\left(\mathbf{X}\right)\right]}{3!} \\
&\cdots
\end{aligned}
$$

such that

$$
g\left(\mathbf{X}\right) = \mathbb{E}\left[f\left(\mathbf{X}\right)\right] + \underbrace{\sum_{s=1}^{\infty} \boldsymbol{A}_s \circ \left(\boldsymbol{\Xi}^{\circ s}_{(s)} - \mathbb{E}\left[\boldsymbol{\Xi}^{\circ s}_{(s)}\right]\right)}_{\text{zero mean}}
$$

A.1.1 Mean

$\boldsymbol{A}_0$ is simply the mean of the output. In the later sections we show this value can either be analytically solved or approximated using similarly behaving nonlinear functions.

A.1.2 Covariance

For clarity of reading, we omit the subscript of ξ, but will revisit the independence of polynomial basis. The covariance function of $f(\mathbf{X})$ with S-th order expansion is

$$\begin{aligned}\operatorname{cov}\left(g(\mathbf{X})\right) &= \mathbb{E}\left[\left(\sum_{s=1}^{S}\boldsymbol{A}_s\circ\left(\boldsymbol{\Xi}_{(s)}^{\circ s}-\mathbb{E}[\boldsymbol{\Xi}_{(s)}^{\circ s}]\right)\right)\left(\sum_{t=1}^{S}\boldsymbol{A}_t\circ\left(\boldsymbol{\Xi}_{(t)}^{\circ t}-\mathbb{E}[\boldsymbol{\Xi}_{(t)}^{\circ t}]\right)\right)^{\top}\right]\\ &=\sum_{s=1}^{S}\sum_{t=1}^{S}\left(\boldsymbol{A}_s\boldsymbol{A}_t^{\top}\right)\circ\mathbb{E}\left[\left(\boldsymbol{\Xi}_{(s)}^{\circ s}-\mathbb{E}[\boldsymbol{\Xi}_{(s)}^{\circ s}]\right)\left(\boldsymbol{\Xi}_{(t)}^{\circ t}-\mathbb{E}[\boldsymbol{\Xi}_{(t)}^{\circ t}]\right)^{\top}\right]\\ &=\sum_{s=1}^{S}\sum_{t=1}^{S}\left(\boldsymbol{A}_s\boldsymbol{A}_t^{\top}\right)\circ\left(\mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}(\boldsymbol{\Xi}_{(t)}^{\circ t})^{\top}\right]-\mathbb{E}[\boldsymbol{\Xi}_{(s)}^{\circ s}]\mathbb{E}[\boldsymbol{\Xi}_{(t)}^{\circ t}]^{\top}\right)\end{aligned}$$

which is an $n\times n$ matrix, and the $\{i,j\}$-th entry is

$$\sum_{s=1}^{S}\sum_{t=1}^{S}A_{s,i}A_{t,j}\left(\mathbb{E}\left[\Xi_{(s),i}^{s}\Xi_{(t),j}^{t}\right]-\mathbb{E}\left[\Xi_{(s),i}^{s}\right]\mathbb{E}\left[\Xi_{(t),j}^{t}\right]\right)$$

Since $\boldsymbol{\Xi}_{(1)},\boldsymbol{\Xi}_{(2)}^{\circ 2},\boldsymbol{\Xi}_{(3)}^{\circ 3},\cdots$ are independent, the off-diagonal entries ($s\neq t$) are zero, then the $\{i,j\}$-th entry becomes

$$\sum_{s=1}^{S}A_{s,i}A_{s,j}\left(\mathbb{E}\left[\Xi_{(s),i}^{s}\Xi_{(s),j}^{s}\right]-\mathbb{E}\left[\Xi_{(s),i}^{s}\right]\mathbb{E}\left[\Xi_{(s),j}^{s}\right]\right)$$

Rewrite in matrix form

$$\operatorname{cov}\left(g(\mathbf{X})\right)=\sum_{s=1}^{S}\boldsymbol{A}_s\circ\left(\mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\ \boldsymbol{\Xi}_{(s)}^{\circ s^{\top}}\right]-\mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\right]\mathbb{E}\left[\boldsymbol{\Xi}_{(s)}^{\circ s}\right]^{\top}\right)\circ\boldsymbol{A}_s^{\top}\tag{5}$$

where $\boldsymbol{A}_s$ and $\boldsymbol{\Xi}_{(s)}$ are both n dimensional vertical vectors. Using central moments of normal distributions,

$$\mathbb{E}\left[\Xi_{(s),i}^{s}\right]=\begin{cases}0 & \text{if } s \text{ is odd}\\ \tilde{\sigma}_i^s(s-1)!! & \text{if } s \text{ is even}\end{cases}$$

With application of Isserlis' theorem (Isserlis, 1918),

$$\mathbb{E}\left[\Xi_{(s),i}^{s}\Xi_{(s),j}^{s}\right]=\sum_{p\in P_{\mathcal{B}}^2}\prod_{\{c,d\}\in p}\tilde{\rho}_{cd}\tilde{\sigma}_c\tilde{\sigma}_d$$

where $c,d\in\{i,j\}$, and $\tilde{\rho}_{cd}$ is correlation. The sum is over all the pairings of the set $\mathcal{B}=\{\underbrace{i,i,\cdots,i}_{\text{s}},\underbrace{j,j,\cdots,j}_{\text{s}}\}$, i.e. all distinct (suppose each i or j is different from other i's or j's) ways of partitioning $\mathcal{B}$ into pairs $\{c,d\}$, and the product is over the pairs contained in p (Janson, 1997; Michalowicz et al., 2011), so there exists $(2s-1)!!$ pairs in the partition, or $(2s-1)!!$ terms in the sum. For example, the first four terms of Eqn. 5 are

$$\begin{aligned}&\boldsymbol{A}_1\circ\tilde{\boldsymbol{\Sigma}}\circ\boldsymbol{A}_1^{\top}\\ &\boldsymbol{A}_2\circ\left(2\tilde{\boldsymbol{\Sigma}}^{\circ 2}\right)\circ\boldsymbol{A}_2^{\top}\\ &\boldsymbol{A}_3\circ\left[6\tilde{\boldsymbol{\Sigma}}^{\circ 3}+9\ \operatorname{diag}(\tilde{\boldsymbol{\Sigma}})\circ\tilde{\boldsymbol{\Sigma}}\circ\operatorname{diag}(\tilde{\boldsymbol{\Sigma}})^{\top}\right]\circ\boldsymbol{A}_3^{\top}\\ &\boldsymbol{A}_4\circ\left[24\tilde{\boldsymbol{\Sigma}}^{\circ 4}+72\ \operatorname{diag}(\tilde{\boldsymbol{\Sigma}})\circ\tilde{\boldsymbol{\Sigma}}^{\circ 2}\circ\operatorname{diag}(\tilde{\boldsymbol{\Sigma}})^{\top}\right]\circ\boldsymbol{A}_4^{\top}\end{aligned}$$

$\boldsymbol{A}_s$ and $\operatorname{diag}(\tilde{\boldsymbol{\Sigma}})$ are both n dimensional vertical vector. With this result, to find the covariance of the output of a nonlinear layer, assuming the input follows a multi-variate normal distribution, one just needs to derive the factors of Taylor polynomial, $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, etc., for the nonlinearity.

A.1.3 Cross-covariance

Let $\mathbf{Y} \coloneqq (\mathrm{Y}_1, \mathrm{Y}_2, \cdots, \mathrm{Y}_\mathrm{n})^\top$ be a Gaussian random vector that is cross-correlated to $\mathbf{X}$, and $\mathbf{\Omega} = \mathbf{Y} - \mathbb{E}[\mathbf{Y}]$. If $\mathbf{X}$ undergoes a non-linear transformation via function $f(\cdot)$,

$$\mathbf{Z} = f(\mathbf{X})$$

The cross-covariance between $\mathbf{Y}$ and $\mathbf{Z}$ can be written as

$$\mathbf{\Sigma_{YZ}} = \mathbb{E}\left[\mathbf{\Omega}\left(\sum_{t=1}^{S} \boldsymbol{A}_t \circ \left(\mathbf{\Xi}_{(t)}^{\circ t} - \mathbb{E}\left[\mathbf{\Xi}_{(t)}^{\circ t}\right]\right)\right)^\top\right]$$

which is an $n \times n$ matrix, and the $\{i, j\}$-th entry is

$$\sum_{t=1}^{S} A_{t,j}\left(\mathbb{E}\left[\Omega_i \Xi_{(t),j}^{t}\right] - \mathbb{E}\left[\Omega_i\right]\mathbb{E}\left[\Xi_{(t),j}^{t}\right]\right)$$

$\mathbb{E}\left[\Omega_i\right]$ is zero by definition. The terms with product of odd number of Gaussian random variables are zero by Isserlis' theorem. It can be simplified as

$$\mathbf{\Sigma_{YZ}}(i,j) = \sum_{t=1, t \text{ is odd}}^{S} A_{t,j}\mathbb{E}\left[\Omega_i \Xi_{(t),j}^{t}\right]$$
$$\mathbf{\Sigma_{ZY}}(i,j) = \sum_{s=1, s \text{ is odd}}^{S} A_{s,i}\mathbb{E}\left[\Xi_{(s),i}^{s} \Omega_j\right]$$

Rewrite in matrix form

$$\begin{aligned}\mathbf{\Sigma_{YZ}} &= \sum_{t=1, t \text{ is odd}}^{S} \boldsymbol{A}_t^\top \circ \mathbb{E}\left[\mathbf{\Omega}\ \mathbf{\Xi}_{(t)}^{\circ t^\top}\right] \\ \mathbf{\Sigma_{ZY}} &= \sum_{s=1, s \text{ is odd}}^{S} \boldsymbol{A}_s \circ \mathbb{E}\left[\mathbf{\Xi}_{(s)}^{\circ s}\ \mathbf{\Omega}^\top\right]\end{aligned} \tag{6}$$

and the expected value can be calculated using Isserlis' theorem mentioned above. Note that this term is nonzero only if t and s are odd, so the first two terms are

$$\mathbf{\Sigma_{YZ}} \approx \boldsymbol{A}_1^\top \circ \mathbf{\Sigma_{YX}} + 3\boldsymbol{A}_3^\top \circ \mathbf{\Sigma_{YX}} \circ \mathrm{diag}(\mathbf{\Sigma_X})^\top$$
$$\mathbf{\Sigma_{ZY}} \approx \boldsymbol{A}_1 \circ \mathbf{\Sigma_{XY}} + 3\boldsymbol{A}_3 \circ \mathbf{\Sigma_{XY}} \circ \mathrm{diag}(\mathbf{\Sigma_X})$$

An addition (e.g. residual) layer outputs the summation of activation of two (or more) layers, $\mathbf{Y}$ and $\mathbf{Z}$. Thus the covariance of $\mathbf{Y} + \mathbf{Z}$ is the sum of their covariance and cross-covariance.

$$\mathbf{\Sigma}(\mathbf{Y} + \mathbf{Z}) = \mathbf{\Sigma_Y} + \mathbf{\Sigma_{YZ}} + \mathbf{\Sigma_{ZY}} + \mathbf{\Sigma_Z}$$

A.2 Tanh layers †

We use a linear combination of **independent** error functions with different scaling factors to approximate **tanh** function. In our experiments, we choose a set of four scaling parameters, {0.5583, 0.8596, 0.8596, 1.2612}, using `fmincon` in MATLAB. In practice, one can add more terms for even higher accuracy without losing efficiency (depending on the computing resources), because the extra terms can be easily paralleled. We define a variance term considering the relation between error function and Gaussian cdf, such that

† For notational simplicity, all the product, division, and power operations that appear in and after this section are all element-wise.

$$\begin{aligned}\mathbf{tanh}(\mathbf{X}) &\approx \frac{1}{p}\sum_{j=1}^{p}\mathbf{erf}\left(\gamma_j\mathbf{X}\right)\\ \acute{\sigma}_j^2 &= \frac{1}{2\gamma_j^2}\end{aligned} \tag{7}$$

Thus, the factors of the pseudo Taylor polynomials are

$$\begin{aligned}\boldsymbol{A}_0 &= \mathbb{E}\left[\frac{1}{p}\sum_{j=1}^{p}\mathbf{erf}\left(\frac{\mathbf{X}}{\sqrt{2\acute{\sigma}_j^2}}\right)\right]\\ \boldsymbol{A}_1 &= \mathbb{E}\left[\nabla_{\mathbf{x}}\left(\frac{1}{p}\sum_{j=1}^{p}\mathbf{erf}\left(\frac{\mathbf{X}}{\sqrt{2\acute{\sigma}_j^2}}\right)\right)\right]\\ \boldsymbol{A}_2 &= \frac{1}{2!}\mathbb{E}\left[\nabla_{\mathbf{x}}^{\circ 2}\left(\frac{1}{p}\sum_{j=1}^{p}\mathbf{erf}\left(\frac{\mathbf{X}}{\sqrt{2\acute{\sigma}_j^2}}\right)\right)\right]\\ &\cdots\end{aligned}$$

Since all the operations in $\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_2, \cdots$ are element-wise, we only show the derivation for univariate case for notational simplicity in the following sections

A.2.1 Find A_0

$$\mathbb{E}\left[\mathbf{erf}\left(\frac{\mathrm{X}}{\sqrt{2\acute{\sigma}_j^2}}\right)\right] = \int_{-\infty}^{\infty}\mathbf{erf}\left(\frac{x}{\sqrt{2\acute{\sigma}_j^2}}\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x$$

This is a known integral Ng and Geller (1969)

$$= \mathbf{erf}\left(\frac{\tilde{\mu}}{\sqrt{2\tilde{\sigma}^2+2\acute{\sigma}_j^2}}\right)$$

We define

$$\hat{\sigma}_j^2 = \tilde{\sigma}^2 + \acute{\sigma}_j^2 \tag{8}$$

Thus,

$$\boxed{A_0 = \frac{1}{p}\sum_{j=1}^{p}\mathbf{erf}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)} \tag{9}$$

The usage of error function instead of Gaussian cdf may give A_0 a very distinctive form from those of the other factors. The reasons behind are purely out of considerations of numerical computing: calculating Gaussian cdf is computationally demanding, while the approximation algorithm of the error function is available Cody (1969).

A.2.2 Find A_1

Notice that

$$\frac{\partial}{\partial \mathrm{x}}\mathbf{erf}\left(\frac{\mathrm{X}}{\sqrt{2\acute{\sigma}_j^2}}\right) = \frac{\partial}{\partial x}\left(\int_0^{x/\sqrt{2\acute{\sigma}_j^2}}\frac{2}{\sqrt{\pi}}\mathbf{exp}(-t^2)\mathrm{d}t\right)$$

by Leibniz integral rule

$$= \frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{2\acute{\sigma}_j^2}} \mathbf{exp}\left(-\frac{x^2}{2\acute{\sigma}_j^2}\right)$$
$$= \frac{2}{\acute{\sigma}_j} \boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right)$$

where φ is the standard normal pdf. With the identity that the convolution of two Gaussians is still a Gaussian. (Bromiley (2003))

$$\mathbb{E}\left[\frac{\partial}{\partial \mathrm{x}} \mathbf{erf}\left(\frac{\mathrm{X}}{\sqrt{2\acute{\sigma}_j^2}}\right)\right] = \int_{-\infty}^{\infty} \frac{2}{\acute{\sigma}_j} \boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right) \frac{1}{\tilde{\sigma}} \boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right) \mathrm{d}x$$
$$= \frac{2}{\sqrt{\tilde{\sigma}^2 + \acute{\sigma}_j^2}} \boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\sqrt{\tilde{\sigma}^2 + \acute{\sigma}_j^2}}\right)$$

Therefore,

$$\boxed{A_1 = \frac{1}{p} \sum_{j=1}^{p} \frac{2}{\hat{\sigma}_j} \boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)} \tag{10}$$

and each term of the summation is a Gaussian function written in its standardized form.

A.2.3 Find A_2 and beyond

In previous section, we show that the first derivative of the error function is a Gaussian, thus the expected value of which is the convolution of two Gaussians. Similarly, we can obtain A_2, A_3, etc. by convolving the second, third, and higher order Gaussian derivatives with another Gaussian.

Gaussian derivatives can be represented by Hermite polynomial $\mathbf{H}_s(x)$.

$$\frac{d^s}{dx^s}\left[\frac{1}{\sigma}\boldsymbol{\varphi}\left(\frac{x}{\sigma}\right)\right] = \left(\frac{-1}{\sqrt{2\sigma^2}}\right)^s \mathbf{H}_s\left(\frac{x}{\sqrt{2\sigma^2}}\right) \frac{1}{\sigma}\boldsymbol{\varphi}\left(\frac{x}{\sigma}\right)$$

For examples,

$$\mathbf{H}_0(x) = 1$$
$$\mathbf{H}_1(x) = 2x$$
$$\mathbf{H}_2(x) = 4x^2 - 2$$
$$\mathbf{H}_3(x) = 8x^3 - 12x$$
$$\cdots$$

There are implemented functions for this from various scientific computing tools, such as hermiteH() from MATLAB and scipy.special.hermite() from SciPy.

Hence,

$$
\begin{aligned}
&\mathbb{E}\left[\frac{\partial^s}{\partial \mathrm{x}^s}\mathbf{erf}\left(\frac{\mathrm{X}}{\sqrt{2\acute{\sigma}_j^2}}\right)\right] \\
=&\int_{-\infty}^{\infty}\left[\frac{\partial^s}{\partial x^s}\mathbf{erf}\left(\frac{x}{\sqrt{2\acute{\sigma}_j^2}}\right)\right]p(x)\mathrm{d}x \\
=&2\int_{-\infty}^{\infty}\frac{\partial^{s-1}}{\partial x^{s-1}}\left[\frac{1}{\acute{\sigma}_j}\boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right)\right]\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
=&2\left(\frac{-1}{\sqrt{2\acute{\sigma}_j^2}}\right)^{s-1}\int_{-\infty}^{\infty}\mathbf{H}_{s-1}\left(\frac{x}{\sqrt{2\acute{\sigma}_j^2}}\right)\frac{1}{\acute{\sigma}_j}\boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)dx \\
=&2\left(\frac{-1}{\sqrt{2\acute{\sigma}_j^2}}\right)^{s-1}\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\int_{-\infty}^{\infty}\mathbf{H}_{s-1}\left(\frac{x}{\sqrt{2\acute{\sigma}_j^2}}\right)\frac{1}{\bar{\sigma}_j}\boldsymbol{\varphi}\left(\frac{x-\bar{\mu}}{\bar{\sigma}_j}\right)dx
\end{aligned}
$$

where $\bar{\mu}=\tilde{\mu}\frac{\acute{\sigma}_j^2}{\hat{\sigma}_j^2}\quad \bar{\sigma}_j^2=\tilde{\sigma}^2\frac{\acute{\sigma}_j^2}{\hat{\sigma}_j^2}$. The convolution of a Hermite polynomial and a Gaussian pdf is a known integral Gradshteyn and Ryzhik (2015)

$$
\begin{aligned}
=&2\left(\frac{-1}{\sqrt{2\acute{\sigma}_j^2}}\right)^{s-1}\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\left(1-2\bar{\sigma}^2\frac{1}{2\acute{\sigma}_j^2}\right)^{\frac{s-1}{2}}\mathbf{H}_{s-1}\left(\frac{\bar{\mu}/\sqrt{2\acute{\sigma}_j^2}}{\left(1-2\bar{\sigma}^2\frac{1}{2\acute{\sigma}_j^2}\right)^{\frac{1}{2}}}\right) \\
=&2\left(\frac{-1}{\sqrt{2\hat{\sigma}_j^2}}\right)^{s-1}\mathbf{H}_{s-1}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)
\end{aligned}
$$

Therefore, we can write the formula of A_s for $s\geq 1$

$$
\boxed{A_s(s\geq 1)=\frac{1}{s!\;p}\sum_{j=1}^{p}2\left(\frac{-1}{\sqrt{2\hat{\sigma}_j^2}}\right)^{s-1}\mathbf{H}_{s-1}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)} \tag{11}
$$

To give a few examples,

$$
\boxed{\begin{aligned}
A_2&=\frac{1}{2!\;p}\sum_{j=1}^{p}-2\frac{\tilde{\mu}}{\hat{\sigma}_j^2}\;\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\\
A_3&=\frac{1}{3!\;p}\sum_{j=1}^{p}2\frac{\tilde{\mu}^2-\hat{\sigma}_j^2}{\hat{\sigma}_j^4}\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\\
A_4&=\frac{1}{4!\;p}\sum_{j=1}^{p}2\left(\frac{-\tilde{\mu}^3+3\tilde{\mu}\hat{\sigma}_j^2}{\hat{\sigma}_j^6}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\\
&\dots
\end{aligned}} \tag{12}
$$

Note that we will reuse this relation in the following section

$$
\int_{-\infty}^{\infty}\left(\frac{-1}{\sqrt{2\acute{\sigma}_j^2}}\right)^{s}\mathbf{H}_s\left(\frac{x}{\sqrt{2\acute{\sigma}_j^2}}\right)\frac{1}{\acute{\sigma}_j}\boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)dx=\left(\frac{-1}{\sqrt{2\hat{\sigma}_j^2}}\right)^{s}\mathbf{H}_s\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \tag{13}
$$

A.3 Sigmoid layers

We can apply the same framework on **sigmoid** layers, with modifications

$$\mathbf{sigmoid}(x) = \frac{1}{1+\mathbf{exp}(-x)} \approx \frac{1}{2} + \frac{1}{2p}\sum_{j=1}^{p} \mathbf{erf}\left(\gamma_j x\right)$$

Using `fmincon` in MATLAB, we find a set of $\gamma = (0.2791, 0.4298, 0.4298, 0.6306)^\top$. Then the first four factors of the Taylor polynomials are listed below. A_0 is represented in complementary error function **erfc** to avoid subtractive cancellation that leads to inaccuracy in the tails. Note that except for A_0, all A_s of **sigmoid** layers are just $1/2$ of those of **tanh** layers.

$$\boxed{\begin{aligned}
A_0 &= \frac{1}{p}\sum_{j=1}^{p} \frac{1}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right) \\
A_1 &= \frac{1}{p}\sum_{j=1}^{p} \frac{1}{\hat{\sigma}_j}\,\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_2 &= \frac{1}{2!\;p}\sum_{j=1}^{p} -\frac{\tilde{\mu}}{\hat{\sigma}_j^2}\,\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_3 &= \frac{1}{3!\;p}\sum_{j=1}^{p} \frac{\tilde{\mu}^2-\hat{\sigma}_j^2}{\hat{\sigma}_j^4}\,\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_4 &= \frac{1}{4!\;p}\sum_{j=1}^{p} \left(\frac{-\tilde{\mu}^3+3\tilde{\mu}\hat{\sigma}_j^2}{\hat{\sigma}_j^6}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
&\cdots
\end{aligned}} \tag{14}$$

A.4 Softplus layers

The derivation of pseudo-Taylor polynomials for a **softplus** layer is related to that for a **sigmoid** layer, since the derivative of the **softplus** function is the **sigmoid** function with scaling factor β, and the latter can be approximated with a linear combination of Gaussian cdf (or error functions like we did in the previous section). We have

$$\mathbf{softplus}(x) = \frac{1}{\beta}\mathbf{log}\left(1+e^{\beta x}\right)$$

Then we use the approximation of sum of **independent** standard Gaussian cdf $\boldsymbol{\Phi}$

$$\frac{\partial}{\partial x}\mathbf{softplus}(x) = \frac{1}{1+e^{-\beta x}} \approx \frac{1}{p}\sum_{j=1}^{p} \boldsymbol{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right) \tag{15}$$

where we re-define

$$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2\beta^2} \tag{16}$$

Note that $\acute{\sigma}_j^2$ changes definition and should not be confused with that in the **tanh** and **sigmoid** sections.

A.4.1 Find A_0

First we apply substitution of variables $X = \tilde{\mu} + \Xi$, then

$$\mathbf{softplus}(x) = \mathbf{softplus}(\tilde{\mu}, \xi) = \frac{1}{\beta}\mathbf{log}\left(1+e^{\beta(\tilde{\mu}+\xi)}\right)$$

Notice that $\dfrac{\partial}{\partial x} = \dfrac{\partial}{\partial \xi}$ since $\tilde{\mu}$ is constant, then

$$\begin{aligned} A_0 &= \int_{-\infty}^{\infty} \textbf{softplus}(\tilde{\mu}, \xi)\ p(\xi)\ \mathrm{d}\xi \\ &= \int_{-\infty}^{\infty}\ p(\xi)\mathrm{d}\xi \int_{-\infty}^{\tilde{\mu}} \frac{\partial}{\partial \zeta}\textbf{softplus}(\zeta, \xi)\ \mathrm{d}\zeta \end{aligned}$$

by Fubini's theorem (Fubini, 1907)

$$\begin{aligned} &= \int_{-\infty}^{\tilde{\mu}}\ \mathrm{d}\zeta \int_{-\infty}^{\infty} \frac{\partial}{\partial \zeta}\textbf{softplus}(\zeta, \xi)\ p(\xi)\mathrm{d}\xi \\ &\approx \frac{1}{p}\sum_{j=1}^{p} \int_{-\infty}^{\tilde{\mu}}\ \mathrm{d}\zeta \int_{-\infty}^{\infty} \boldsymbol{\Phi}\left(\frac{\zeta+\xi}{\acute{\sigma}_j}\right) \frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\xi}{\tilde{\sigma}}\right) \mathrm{d}\xi \end{aligned}$$

using the known Gaussian integral identity $\int_{-\infty}^{\infty} \boldsymbol{\Phi}(ax+b)\boldsymbol{\varphi}(x)dx = \boldsymbol{\Phi}\left(\frac{b}{\sqrt{1+a^2}}\right)$

$$\begin{aligned} &= \frac{1}{p}\sum_{j=1}^{p} \int_{-\infty}^{\tilde{\mu}} \boldsymbol{\Phi}\left(\frac{\zeta}{\hat{\sigma}_j}\right) \mathrm{d}\zeta \\ &= \frac{1}{p}\sum_{j=1}^{p} \left[\tilde{\mu}\ \boldsymbol{\Phi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) + \hat{\sigma}_j\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\right] \\ &= \frac{1}{p}\sum_{j=1}^{p} \left[\frac{\tilde{\mu}}{2}\ \textbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right) + \hat{\sigma}_j\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\right] \end{aligned}$$

Or, with simplification

$$\boxed{A_0 = A_1\tilde{\mu} + \frac{1}{p}\sum_{j=1}^{p} \hat{\sigma}_j\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)} \tag{17}$$

A.4.2 Find A_1

Since the first derivative of the **softplus** function is just a **sigmoid** function with scaling factor β, we can immediately write A_1 using previous results

$$\boxed{A_1 = \frac{1}{p}\sum_{j=1}^{p} \frac{1}{2}\textbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)} \tag{18}$$

A.4.3 Find A_2 and beyond

In previous section, we find that $\nabla\ \textbf{softplus}(x)$ is approximately a Gaussian cdf. Subsequently, $\nabla^2\ \textbf{softplus}(x)$ is approximately a Gaussian. Since Gaussian function is infinitely differentiable, all $A_s(s > 2)$ can be found

using Gaussian derivatives, which can be represented by Hermite polynomial $\mathbf{H}_s(x)$ introduced above.

$$
\begin{aligned}
A_s &= \frac{1}{s!}\mathbb{E}\left[\frac{\partial^s}{\partial x^s}\mathbf{softplus}(x)\right] \\
&\approx \frac{1}{s!\ p}\sum_{j=1}^{p}\int_{-\infty}^{\infty}\frac{\partial^{s-2}}{\partial x^{s-2}}\left[\frac{1}{\acute{\sigma}_j}\boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right)\right]p(x)\ \mathrm{d}x \\
&= \frac{1}{s!\ p}\sum_{j=1}^{p}\left(\frac{-1}{\sqrt{2\acute{\sigma}_j^2}}\right)^{s-2}\int_{-\infty}^{\infty}\mathbf{H}_{s-2}\left(\frac{x}{\sqrt{2\acute{\sigma}_j^2}}\right)\frac{1}{\acute{\sigma}_j}\ \boldsymbol{\varphi}\left(\frac{x}{\acute{\sigma}_j}\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x
\end{aligned}
$$

we solved this integral in**tanh**section

$$
= \frac{1}{s!\ p}\sum_{j=1}^{p}\left(\frac{-1}{\sqrt{2\hat{\sigma}_j^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)
$$

To summarize, $A_s(s \geq 2)$ can be expressed as

$$
\boxed{A_s(s \geq 2) = \frac{1}{s!\ p}\sum_{j=1}^{p}\left(\frac{-1}{\sqrt{2\hat{\sigma}_j^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right)\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)} \tag{19}
$$

For examples,

$$
\boxed{\begin{aligned}
A_2 &= \frac{1}{2!\ p}\sum_{j=1}^{p}\frac{1}{\hat{\sigma}_j}\ \boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_3 &= \frac{1}{3!\ p}\sum_{j=1}^{p}-\frac{\tilde{\mu}}{\hat{\sigma}_j^2}\ \frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_4 &= \frac{1}{4!\ p}\sum_{j=1}^{p}\frac{\tilde{\mu}^2-\hat{\sigma}_j^2}{\hat{\sigma}_j^4}\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
&\dots
\end{aligned}} \tag{20}
$$

A.5 ReLU, Leaky ReLU, and Piece-wise Linear layers

Since ReLU function is only first-order differentiable ($x > 0$), we cannot do PTPE directly. However, given its relation to **softplus** function,

$$
\lim_{\beta\to\infty}\frac{1}{\beta}\mathbf{log}\left(1+e^{\beta x}\right) = \max\{0, x\}
$$

we can reuse the results for **softplus** layers by applying the limit

$$
\lim_{\beta\to\infty}\acute{\sigma}_j^2 = 0 \qquad \text{and} \qquad \lim_{\beta\to\infty}\hat{\sigma}_j^2 = \tilde{\sigma}^2
$$

Therefore,

$$\boxed{\begin{aligned}
A_0 &= A_1\tilde{\mu} + \tilde{\sigma}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
A_1 &= \frac{1}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\tilde{\sigma}^2}}\right) \\
A_2 &= \frac{1}{2!}\frac{1}{\tilde{\sigma}}\ \boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
A_3 &= \frac{1}{3!} - \frac{\tilde{\mu}}{\tilde{\sigma}^2}\ \frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
A_4 &= \frac{1}{4!}\frac{\tilde{\mu}^2 - \tilde{\sigma}^2}{\tilde{\sigma}^4}\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
&\dots
\end{aligned}} \tag{21}$$

and for $s \geq 2$ we have the general form of

$$\boxed{A_s(s \geq 2) = \frac{1}{s!}\left(\frac{-1}{\sqrt{2\tilde{\sigma}_j^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\tilde{\sigma}_j^2}}\right)\frac{1}{\tilde{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}_j}\right)} \tag{22}$$

On the other hand, leaky ReLU can be considered as superposition of two ReLU functions - consider a leaky ReLU with negative slope of θ

$$\mathbf{LeakyReLU}(x;\theta) = \mathbf{ReLU}(x) - \theta\ \mathbf{ReLU}(-x) \tag{23}$$

which can also be written as

$$\lim_{\beta\to\infty}\mathbf{softplus}(x) - \theta\ \mathbf{softplus}(-x)$$

Therefore,

$$\begin{aligned}
A_0 &= \lim_{\beta\to\infty}\frac{1}{p}\sum_{j=1}^{p}\left[\tilde{\mu}\boldsymbol{\Phi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) + \hat{\sigma}_j\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\right] - \theta\left[-\tilde{\mu}\boldsymbol{\Phi}\left(-\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) + \hat{\sigma}_j\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right)\right] \\
&= \theta\tilde{\mu} + (1-\theta)\left[\tilde{\mu}\boldsymbol{\Phi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) + \tilde{\sigma}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right)\right]
\end{aligned}$$

To find the expected value of the derivative of **LeakyReLU**, first we find the derivative

$$\begin{aligned}
\frac{\partial}{\partial x}\mathbf{LeakyReLU}(x\ ;\ \theta) &= \lim_{\beta\to\infty}\frac{\partial}{\partial x}\mathbf{softplus}(x) - \theta\frac{\partial}{\partial x}\mathbf{softplus}(-(x)) \\
&= \lim_{\beta\to\infty}\frac{1}{1+e^{-\beta(x)}} + \frac{\theta}{1+e^{\beta(x)}} \\
&\approx \frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right) + \theta\boldsymbol{\Phi}\left(\frac{-x}{\acute{\sigma}_j}\right) \\
&= \lim_{\beta\to\infty}\theta + \frac{1-\theta}{p}\sum_{j=1}^{p}\boldsymbol{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right)
\end{aligned}$$

Then we can write A_1 for **LeakyReLU** as

$$\begin{aligned}
A_1 &= \lim_{\beta\to\infty}\int_{-\infty}^{\infty}\left[\theta+\frac{1-\theta}{p}\sum_{j=1}^{p}\mathbf{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right)\right]\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
&= \theta+\lim_{\beta\to\infty}\frac{1-\theta}{p}\sum_{j=1}^{p}\int_{-\infty}^{\infty}\mathbf{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
&= \theta+\lim_{\beta\to\infty}\frac{1-\theta}{p}\sum_{j=1}^{p}\mathbf{\Phi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
&= \theta+(1-\theta)\mathbf{\Phi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right)
\end{aligned}$$

Rewrite in complementary error function

$$\boxed{A_1=\theta+\frac{1-\theta}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\tilde{\sigma}^2}}\right)} \tag{24}$$

Note that we can also rewrite A_0 using the result of A_1 to improve computational efficiency.

$$\boxed{A_0=A_1\tilde{\mu}+(1-\theta)\tilde{\sigma}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right)} \tag{25}$$

Note that starting from the second order, the derivative of **LeakyReLU** is just that of **ReLU** scaled by $1-\theta$. Therefore,

$$\boxed{\begin{aligned}
A_2 &= \frac{1-\theta}{2}\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
A_3 &= -\frac{1-\theta}{3!}\frac{\tilde{\mu}}{\tilde{\sigma}^3}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
A_4 &= \frac{1-\theta}{4!}\frac{\tilde{\mu}^2-\tilde{\sigma}^2}{\tilde{\sigma}^5}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right) \\
&\cdots
\end{aligned}} \tag{26}$$

and for $s\geq 2$, we have the general form of

$$\boxed{A_s(s\geq 2)=\frac{1-\theta}{s!}\left(\frac{-1}{\sqrt{2\tilde{\sigma}^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\tilde{\sigma}^2}}\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\tilde{\sigma}}\right)} \tag{27}$$

Similarly, any piece-wise linear activation function can be described as a combination of ReLU functions with different scaling, shifting, and/or mirroring. Thus, their pseudo Taylor coefficients can be found using the same methodology.

A.6 GELU layers

GELU (Gaussian Error Linear Unit) is defined as the product of input and a standard Gaussian cdf

$$\mathbf{GELU}(x)=x\ \mathbf{\Phi}(x)$$

and we can write the derivatives (with order $s\geq 1$) of **GELU** as

$$\frac{\partial^s}{\partial x^s}\mathbf{GELU}(x)=s\frac{\partial^{s-1}}{\partial x^{s-1}}\mathbf{\Phi}(x)+x\frac{\partial^s}{\partial x^s}\mathbf{\Phi}(x)$$

A.6.1 Find A_0

$$
\begin{aligned}
A_0 &= \mathbb{E}\left[\mathbf{GELU}(x)\right] \\
&= \int_{-\infty}^{\infty} x\boldsymbol{\Phi}(x)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
&= \int_{-\infty}^{\infty}(\tilde{\mu}+\xi)\int_{-\infty}^{\tilde{\mu}}\boldsymbol{\varphi}\left(\zeta+\xi\right)\mathrm{d}\zeta\ \frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\xi}{\tilde{\sigma}}\right)\mathrm{d}\xi \\
&= \tilde{\mu}\int_{-\infty}^{\tilde{\mu}}\int_{-\infty}^{\infty}\boldsymbol{\varphi}\left(\zeta+\xi\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\xi}{\tilde{\sigma}}\right)\mathrm{d}\xi\ \mathrm{d}\zeta\ + \int_{-\infty}^{\tilde{\mu}}\int_{-\infty}^{\infty}\xi\ \boldsymbol{\varphi}\left(\zeta+\xi\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{\xi}{\tilde{\sigma}}\right)\mathrm{d}\xi\ \mathrm{d}\zeta \\
&= \tilde{\mu}\int_{-\infty}^{\tilde{\mu}}\frac{1}{\sqrt{1+\tilde{\sigma}^2}}\boldsymbol{\varphi}\left(\frac{\zeta}{\sqrt{1+\tilde{\sigma}^2}}\right)\mathrm{d}\zeta\ + \int_{-\infty}^{\tilde{\mu}}\frac{1}{\sqrt{1+\tilde{\sigma}^2}}\boldsymbol{\varphi}\left(\frac{\zeta}{\sqrt{1+\tilde{\sigma}^2}}\right)\frac{-\zeta\tilde{\sigma}^2}{1+\tilde{\sigma}^2}\mathrm{d}\zeta \\
&= \tilde{\mu}\boldsymbol{\Phi}\left(\frac{\tilde{\mu}}{\sqrt{1+\tilde{\sigma}^2}}\right)+\frac{\tilde{\sigma}^2}{\sqrt{1+\tilde{\sigma}^2}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\sqrt{1+\tilde{\sigma}^2}}\right)
\end{aligned}
$$

We re-define $\hat{\sigma}^2$

$$\hat{\sigma}^2 = 1+\tilde{\sigma}^2 \tag{28}$$

and re-write the result with complementary error function

$$\boxed{A_0 = \frac{\tilde{\mu}}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)+\frac{\tilde{\sigma}^2}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)} \tag{29}$$

A.6.2 Find A_1

$$
\begin{aligned}
A_1 &= \mathbb{E}\left[\frac{\partial}{\partial x}\mathbf{GELU}(x)\right] \\
&= \int_{-\infty}^{\infty}\boldsymbol{\Phi}(x)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}(\frac{x-\tilde{\mu}}{\tilde{\sigma}})\mathrm{d}x\ + \int_{-\infty}^{\infty}x\boldsymbol{\varphi}(x)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}(\frac{x-\tilde{\mu}}{\tilde{\sigma}})\mathrm{d}x
\end{aligned}
$$

using results of previous section

$$= \boldsymbol{\Phi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)+\frac{\tilde{\mu}}{\hat{\sigma}^2}\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)$$

Therefore,

$$\boxed{A_1 = \frac{1}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)+\frac{\tilde{\mu}}{\hat{\sigma}^2}\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)} \tag{30}$$

A.6.3 Find A_2 and beyond

Higher order coefficients ($A_s(s \geq 2)$) all consist of two parts: (i) a term of expected value of a Gaussian derivative, (ii) a term of expected value of the product of x and a Gaussian derivative. We have already found a general form of the first term in the **tanh** section

$$\mathbb{E}\left[s\frac{\partial^{s-2}}{\partial x^{s-2}}\boldsymbol{\varphi}(x)\right] = s\left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s-2}\ \mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)\ \frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)$$

To solve the second part, we need to use the Hermite polynomial recurrence relation:

$$x\ \mathbf{H}_{s-1}(x) = \frac{1}{2}\mathbf{H}_s(x)+s\ \mathbf{H}_{s-2}(x)$$

$$
\begin{aligned}
&\mathbb{E}\left[x\frac{\partial^{s-1}}{\partial x^{s-1}}\boldsymbol{\varphi}(x)\right] \\
&= \left(\frac{-1}{\sqrt{2}}\right)^{s-1}\int_{-\infty}^{\infty} x\ \mathbf{H}_{s-1}\left(\frac{x}{\sqrt{2}}\right)\boldsymbol{\varphi}\left(x\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
&= \left(\frac{-1}{\sqrt{2}}\right)^{s-1}\sqrt{2}\int_{-\infty}^{\infty}\frac{x}{\sqrt{2}}\ \mathbf{H}_{s-1}\left(\frac{x}{\sqrt{2}}\right)\ \boldsymbol{\varphi}\left(x\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
&= \left(\frac{-1}{\sqrt{2}}\right)^{s-1}\sqrt{2}\int_{-\infty}^{\infty}\left[\frac{1}{2}\mathbf{H}_{s}\left(\frac{x}{\sqrt{2}}\right)+(s-1)\mathbf{H}_{s-2}\left(\frac{x}{\sqrt{2}}\right)\right]\boldsymbol{\varphi}\left(x\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x \\
&= -\left(\frac{-1}{\sqrt{2}}\right)^{s}\int_{-\infty}^{\infty}\mathbf{H}_{s}\left(\frac{x}{\sqrt{2}}\right)\boldsymbol{\varphi}\left(x\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x\ \cdots \\
&\quad -(s-1)\left(\frac{-1}{\sqrt{2}}\right)^{s-2}\int_{-\infty}^{\infty}\mathbf{H}_{s-2}\left(\frac{x}{\sqrt{2}}\right)\boldsymbol{\varphi}\left(x\right)\frac{1}{\tilde{\sigma}}\boldsymbol{\varphi}\left(\frac{x-\tilde{\mu}}{\tilde{\sigma}}\right)\mathrm{d}x
\end{aligned}
$$

by equation 13

$$
= \frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)\left[-\left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s}\mathbf{H}_{s}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)-(s-1)\left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)\right]
$$

Sum the two integral together, we get the general form of $A_s(s \geq 2)$

$$
\boxed{A_s(s\geq 2) = \frac{1}{s!}\left[\left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)-\left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s}\mathbf{H}_{s}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)\right]\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)} \tag{31}
$$

For examples,

$$
\boxed{\begin{aligned}
A_2 &= \frac{1}{2!}\left[1+\frac{1}{\hat{\sigma}^2}-\frac{\tilde{\mu}^2}{\hat{\sigma}^4}\right]\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right) \\
A_3 &= -\frac{1}{3!}\left[\frac{\tilde{\mu}}{\hat{\sigma}^2}+\frac{3\tilde{\mu}}{\hat{\sigma}^4}-\frac{\tilde{\mu}^3}{\hat{\sigma}^6}\right]\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right) \\
A_4 &= \frac{1}{4!}\left[-\frac{1}{\hat{\sigma}^2}+\frac{\tilde{\mu}^2-3}{\hat{\sigma}^4}+\frac{6\tilde{\mu}^2}{\hat{\sigma}^6}-\frac{\tilde{\mu}^4}{\hat{\sigma}^8}\right]\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right) \\
&\cdots
\end{aligned}} \tag{32}
$$

A.7 SiLU layers

SiLU (Sigmoid Linear Unit), equivalent to **Swish** when $\beta = 1$, is defined as the product of input and a **sigmoid** function

$$
\mathbf{SiLU}(x) = x\ \mathbf{Sigmoid}(x)
$$

In the previous section, we approximate **Sigmoid** function with error functions so that we can reuse derivations from the **Tanh** section. Here we approximate **Sigmoid** function with Gaussian cdf's in order to reuse derivations from the **GELU** section. With γ as a numerically optimized scalar vector, let

$$
\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2} \quad , \quad j \in \{1, \cdots, p\}
$$

Then, we approximate **SiLU** as

$$\mathbf{SiLU}(x) \approx \frac{x}{p}\sum_{j=1}^{p}\mathbf{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right)$$

and we can write the derivatives (with order $s \geq 1$) of **SiLU** as

$$\frac{\partial^s}{\partial u^s}\mathbf{SiLU}(x) = \frac{s}{p}\sum_{j=1}^{p}\frac{\partial^{s-1}}{\partial u^{s-1}}\mathbf{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right) + \frac{x}{p}\sum_{j=1}^{p}\frac{\partial^s}{\partial u^s}\mathbf{\Phi}\left(\frac{x}{\acute{\sigma}_j}\right)$$

The rest of the derivation is very similar to that of **GELU**, so we only list the final results. With

$$\hat{\sigma}_j^2 = \tilde{\sigma}^2 + \acute{\sigma}_j^2$$

$$\boxed{\begin{aligned}
A_0 &= \frac{1}{p}\sum_{j=1}^{p}\frac{\tilde{\mu}}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right) + \frac{\tilde{\sigma}^2}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_1 &= \frac{1}{p}\sum_{j=1}^{p}\frac{1}{2}\mathbf{erfc}\left(-\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}_j^2}}\right) + \tilde{\mu}\frac{\acute{\sigma}_j^2}{\hat{\sigma}_j^2}\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_2 &= \frac{1}{2!\ p}\sum_{j=1}^{p}\left[1 + \frac{\acute{\sigma}_j^2}{\hat{\sigma}_j^2} - \frac{\tilde{\mu}^2\acute{\sigma}_j^2}{\hat{\sigma}_j^4}\right]\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_3 &= -\frac{1}{3!\ p}\sum_{j=1}^{p}\left[\frac{\tilde{\mu}}{\hat{\sigma}_j^2} + \frac{3\tilde{\mu}\acute{\sigma}_j^2}{\hat{\sigma}_j^4} - \frac{\tilde{\mu}^3\acute{\sigma}_j^2}{\hat{\sigma}_j^6}\right]\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
A_4 &= \frac{1}{4!\ p}\sum_{j=1}^{p}\left[-\frac{1}{\hat{\sigma}_j^2} + \frac{\tilde{\mu}^2 - 3\acute{\sigma}_j^2}{\hat{\sigma}_j^4} + \frac{6\tilde{\mu}^2\acute{\sigma}_j^2}{\hat{\sigma}_j^6} - \frac{\tilde{\mu}^4\acute{\sigma}_j^2}{\hat{\sigma}_j^8}\right]\frac{1}{\hat{\sigma}_j}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}_j}\right) \\
&\dots
\end{aligned}} \tag{33}$$

and the general form of $A_s (s \geq 2)$ is

$$\boxed{A_s(s \geq 2) = \frac{1}{s!\ p}\sum_{j=1}^{p}\left[\left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s-2}\mathbf{H}_{s-2}\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right) - \left(\frac{-1}{\sqrt{2\hat{\sigma}^2}}\right)^{s}\mathbf{H}_s\left(\frac{\tilde{\mu}}{\sqrt{2\hat{\sigma}^2}}\right)\right]\frac{1}{\hat{\sigma}}\boldsymbol{\varphi}\left(\frac{\tilde{\mu}}{\hat{\sigma}}\right)} \tag{34}$$

Table 4: First four coefficients of the polynomials for seven commonly used nonlinearities. For notational simplicity, all the product, division, and power operations are element-wise. The method for determining γ_j is outlined in Table 1.

	$\boldsymbol{A}_0$	$\boldsymbol{A}_1$	$\boldsymbol{A}_2$	$\boldsymbol{A}_3$	$\hat{\boldsymbol{\sigma}}_j^2 = \tilde{\boldsymbol{\sigma}}^2 + \acute{\sigma}_j^2$
Tanh	$\frac{1}{p}\sum_{j=1}^{p}(2\boldsymbol{\mathcal{F}} - 1)$	$\frac{1}{p}\sum_{j=1}^{p} 2\boldsymbol{\mathcal{L}}$	$\frac{1}{2}\frac{1}{p}\sum_{j=1}^{p} -2\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}_j^2}$	$\frac{1}{3!}\frac{1}{p}\sum_{j=1}^{p} 2\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}^2 - \hat{\boldsymbol{\sigma}}_j^2}{\hat{\boldsymbol{\sigma}}_j^4}$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2}$
Sigmoid	$\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{F}}$	$\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{L}}$	$\frac{1}{2}\frac{1}{p}\sum_{j=1}^{p} -\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}_j^2}$	$\frac{1}{3!}\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}^2 - \hat{\boldsymbol{\sigma}}_j^2}{\hat{\boldsymbol{\sigma}}_j^4}$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2}$
Softplus	$\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{F}}\tilde{\boldsymbol{\mu}} + \boldsymbol{\mathcal{L}}\hat{\boldsymbol{\sigma}}_j^2$	$\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{F}}$	$\frac{1}{2}\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{L}}$	$\frac{1}{3!}\frac{1}{p}\sum_{j=1}^{p} -\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}_j^2}$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2\beta^2}$
ReLU	$\boldsymbol{\mathcal{F}}\tilde{\boldsymbol{\mu}} + \boldsymbol{\mathcal{L}}\tilde{\boldsymbol{\sigma}}^2$	$\boldsymbol{\mathcal{F}}$	$\frac{1}{2}\boldsymbol{\mathcal{L}}$	$\frac{1}{3!}\left(-\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}}{\tilde{\boldsymbol{\sigma}}^2}\right)$	$\acute{\sigma}^2 = 0$
LeakyReLU (θ)	$\theta\tilde{\boldsymbol{\mu}} + (1-\theta)\left(\boldsymbol{\mathcal{F}}\tilde{\boldsymbol{\mu}} + \boldsymbol{\mathcal{L}}\tilde{\boldsymbol{\sigma}}^2\right)$	$\theta + (1-\theta)\boldsymbol{\mathcal{F}}$	$\frac{1-\theta}{2}\boldsymbol{\mathcal{L}}$	$\frac{1-\theta}{3!}\left(-\boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}}{\tilde{\boldsymbol{\sigma}}^2}\right)$	$\acute{\sigma}^2 = 0$
GELU	$\boldsymbol{\mathcal{F}}\tilde{\boldsymbol{\mu}} + \boldsymbol{\mathcal{L}}\tilde{\boldsymbol{\sigma}}^2$	$\boldsymbol{\mathcal{F}} + \boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}^2}$	$\frac{1}{2}\boldsymbol{\mathcal{L}}\left(1 + \frac{1}{\hat{\boldsymbol{\sigma}}^2} - \frac{\tilde{\boldsymbol{\mu}}^2}{\hat{\boldsymbol{\sigma}}^4}\right)$	$\frac{1}{3!}\boldsymbol{\mathcal{L}}\left(-\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}^2} - \frac{3\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}^4} + \frac{\tilde{\boldsymbol{\mu}}^3}{\hat{\boldsymbol{\sigma}}^6}\right)$	$\acute{\sigma}^2 = 1$
SiLU	$\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{F}}\tilde{\boldsymbol{\mu}} + \boldsymbol{\mathcal{L}}\tilde{\boldsymbol{\sigma}}^2$	$\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{F}} + \boldsymbol{\mathcal{L}}\frac{\tilde{\boldsymbol{\mu}}\acute{\sigma}_j^2}{\hat{\boldsymbol{\sigma}}_j^2}$	$\frac{1}{2}\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{L}}\left(1 + \frac{\acute{\sigma}_j^2}{\hat{\boldsymbol{\sigma}}_j^2} - \frac{\tilde{\boldsymbol{\mu}}^2\acute{\sigma}_j^2}{\hat{\boldsymbol{\sigma}}_j^4}\right)$	$\frac{1}{3!}\frac{1}{p}\sum_{j=1}^{p}\boldsymbol{\mathcal{L}}\left(-\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}_j^2} - \frac{3\tilde{\boldsymbol{\mu}}\acute{\sigma}_j^2}{\hat{\boldsymbol{\sigma}}_j^4} + \frac{\tilde{\boldsymbol{\mu}}^3\acute{\sigma}_j^2}{\hat{\boldsymbol{\sigma}}_j^6}\right)$	$\acute{\sigma}_j^2 = \frac{1}{2\gamma_j^2}$

where $\boldsymbol{\mathcal{L}} = \frac{1}{\hat{\boldsymbol{\sigma}}_j}\,\boldsymbol{\varphi}\left(\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}_j}\right)$ and $\boldsymbol{\mathcal{F}} = \boldsymbol{\Phi}\left(\frac{\tilde{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}_j}\right) = \frac{1}{2}\mathbf{erfc}\left(-\frac{\tilde{\boldsymbol{\mu}}}{\sqrt{2\hat{\boldsymbol{\sigma}}_j^2}}\right)$

A.8 Emperical distributions of ResNets show Gaussianity

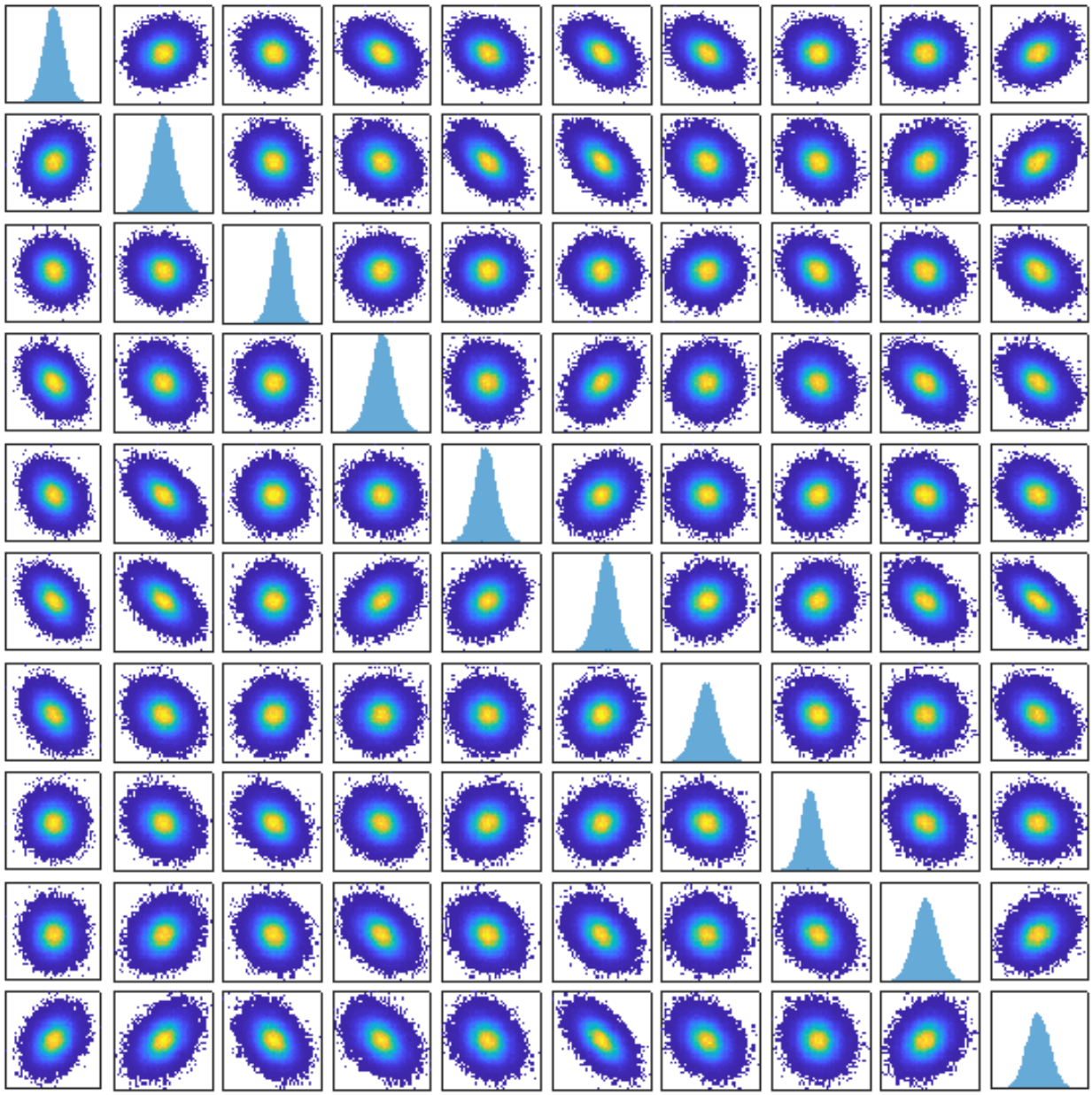

Figure 6: Empirical distributions of all units before the final softmax layer of the resnet13(ReLU).

A.9 Approximation accuracy on other non-linearity

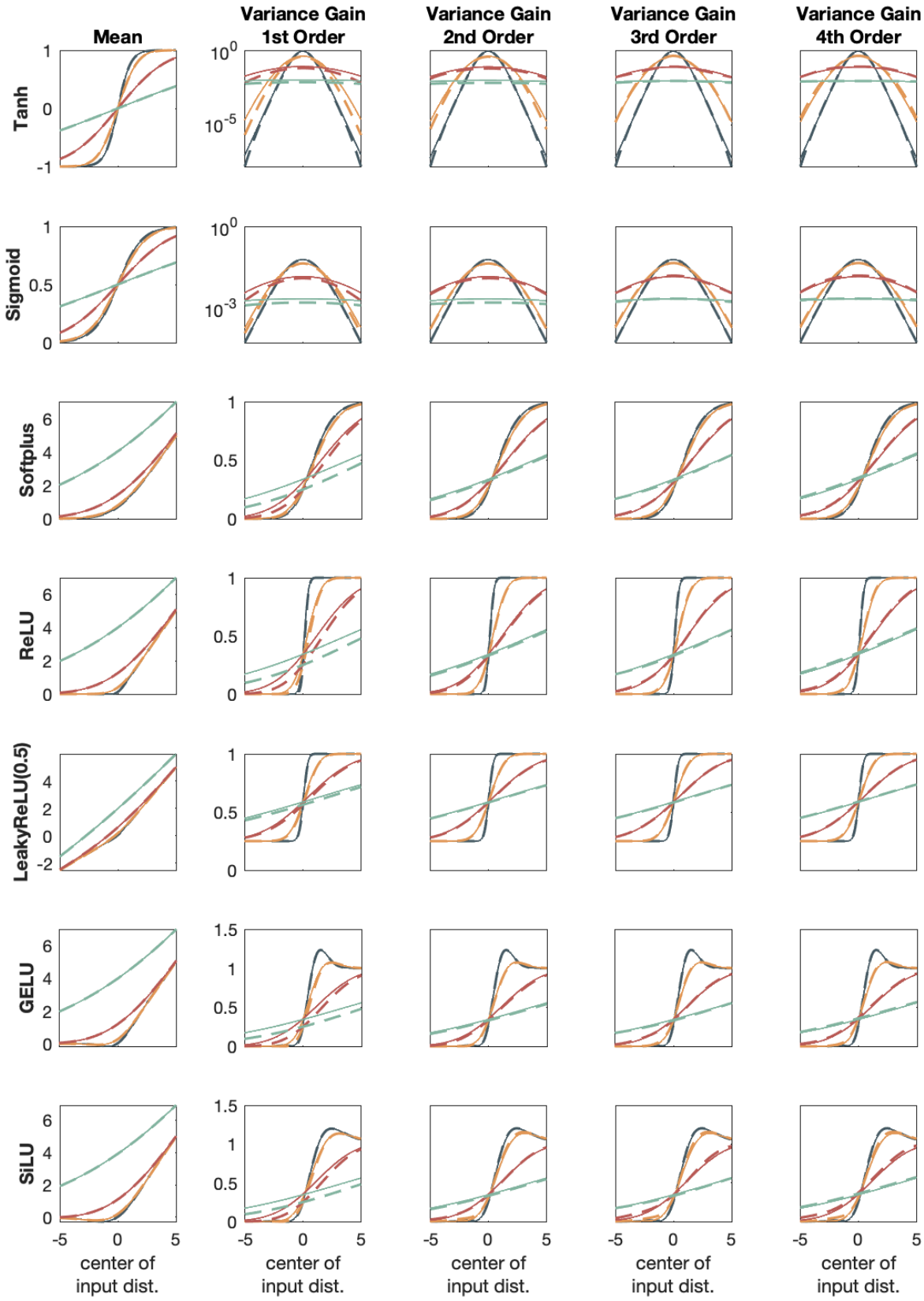

Figure 7: The meaning of different colors and line styles is the same as Fig. 1. The higher the input variance, the more significant is the benefit of using higher order Stochastic Taylor expansion.

A.10 Pseudo Code

Algorithm 1 Propagating a multi-variate Gaussian distribution through a pretrained ResNet

$\mu \leftarrow$ Input mean
$\Sigma \leftarrow$ Input covariance
$\mu_{res} \leftarrow$ Storage for mean of the output of residual layer
$\Sigma_{res} \leftarrow$ Storage for covariance of the output of residual layer
$\Sigma_{cross} \leftarrow$ Storage for cross-covariance between the two input of the residual layer
for layer in neural network **do**
  if layer is linear **then**
    if layer is addition (residual) **then**
      $\mu \leftarrow \mu + \mu_{res}$
      $\Sigma \leftarrow \Sigma + \Sigma_{res} + \Sigma_{cross} + \Sigma_{cross}^{\top}$
      empty $\mu_{res}, \Sigma_{res}, \Sigma_{cross}$
    else
      find effective weight W and bias b
      $\mu \leftarrow W^{\top}\mu + b$
      $\Sigma \leftarrow W^{\top}\Sigma W$
      $\Sigma_{cross} \leftarrow W^{\top}\Sigma_{cross}$
      if residual connection starts from here **then**
        $\mu_{res} \leftarrow \mu$
        $\Sigma_{res}, \Sigma_{cross} \leftarrow \Sigma$
      end if
    end if
  else
    $\mu, \Sigma, \Sigma_{cross} \leftarrow$ PTPE(nonlinearity, $\mu, \Sigma, \Sigma_{cross}$)
    if residual connection starts from here **then**
      $\mu_{res} \leftarrow \mu$
      $\Sigma_{res}, \Sigma_{cross} \leftarrow \Sigma$
    end if
  end if
end for
$\mu_{output} \leftarrow \mu$
$\Sigma_{output} \leftarrow \Sigma$

A.11 Additional results on comparing PTPE and other expectation propagation methods.

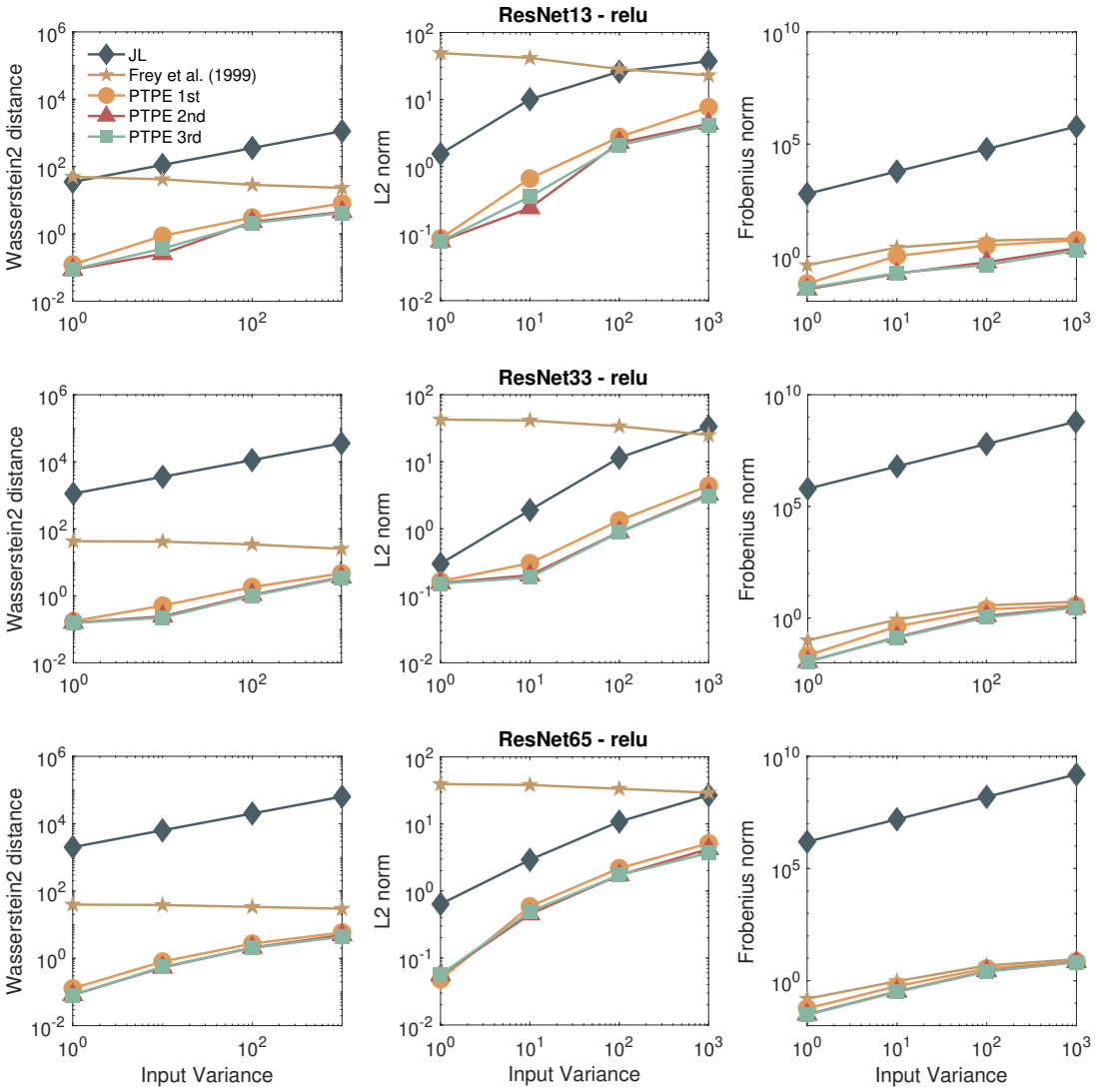

Figure 8: Similar to Figure 3, except that the models employ ReLU as the nonlinearity. The method proposed by Frey and Hinton (1999) performed poorly as it disregards correlation, a critical factor in networks with overlapping convolutional kernels and residual connections.

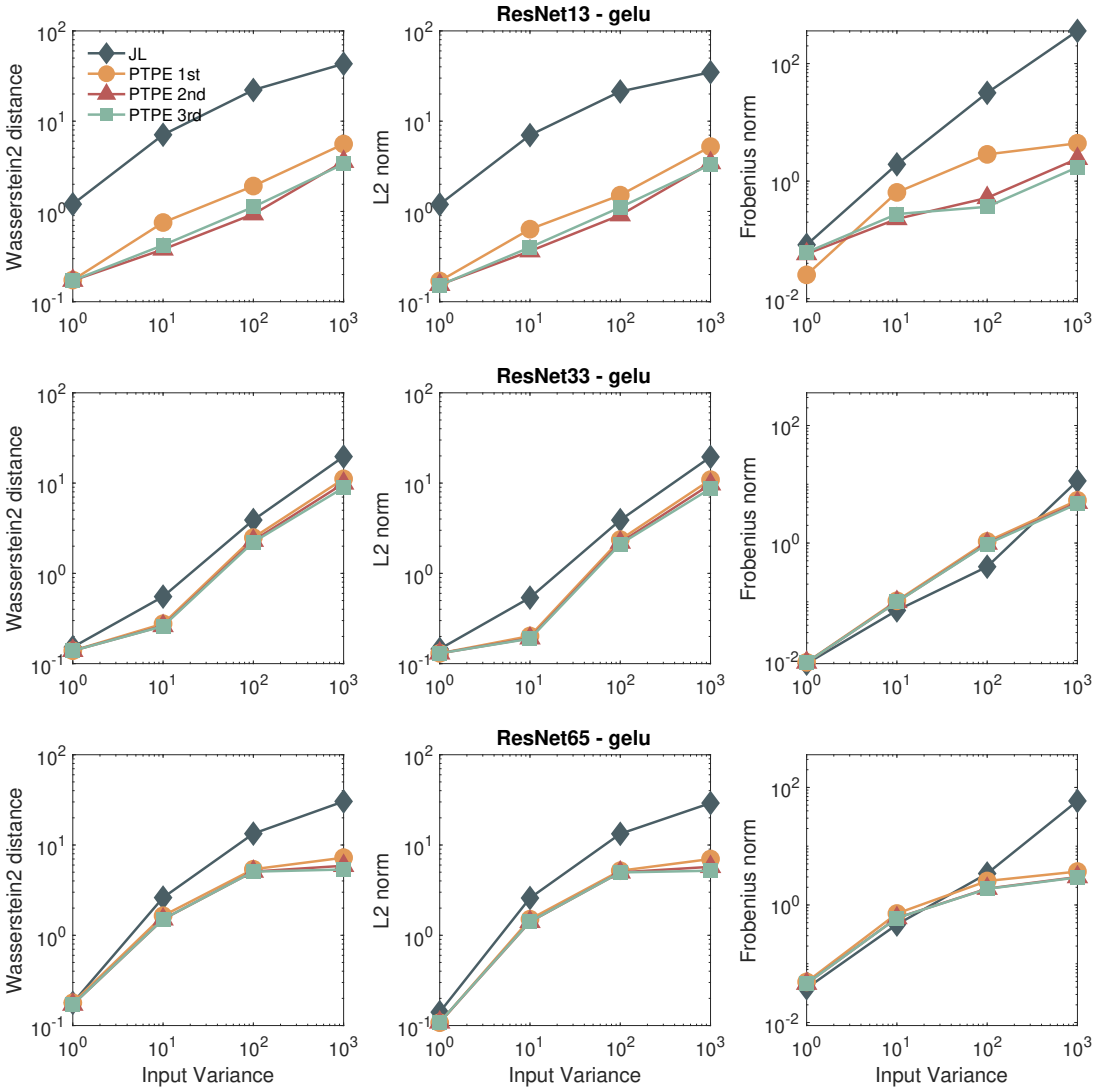

Figure 9: Similar to Figure 3, except that the models employ GELU as the nonlinearity.

A.12 Additional results on the selection of the number of scaling factors in the reformulation of the**tanh**function

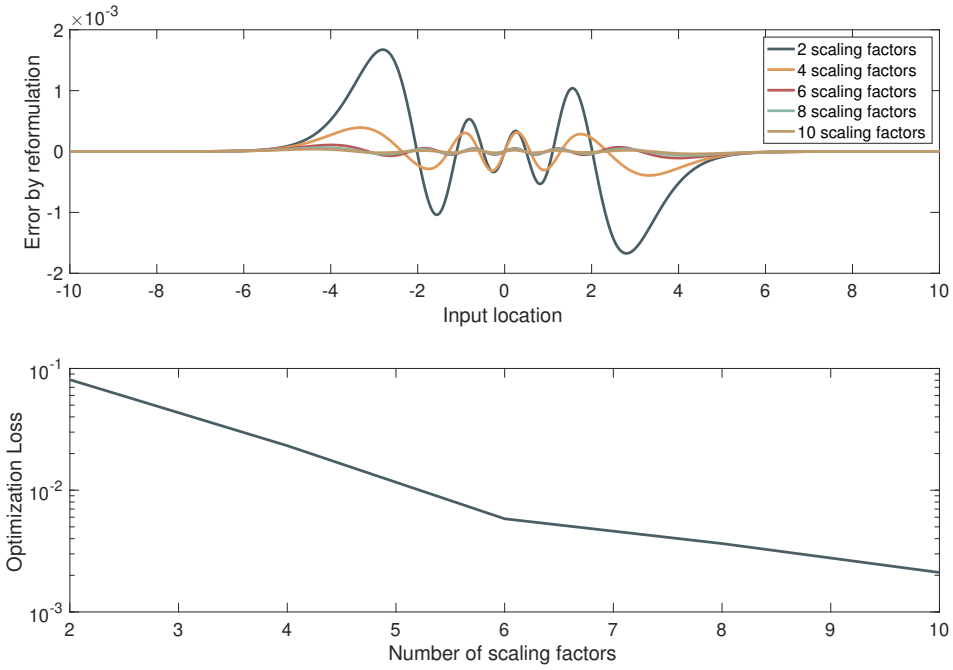

Figure 10: (Top) The error in approximating the**tanh**function using a linear combination of error functions with different scaling factors was analyzed. We selected four terms, as this configuration yields a maximum approximation error on the scale of 10^{-4}. Importantly, increasing the number of terms enhances accuracy while only increasing computational complexity linearly, making this approach both flexible and computationally efficient. (Bottom) The optimization loss for determining the scaling factors was computed using `fmincon` in MATLAB. This loss is defined as the root mean square error (RMSE) of the approximation.

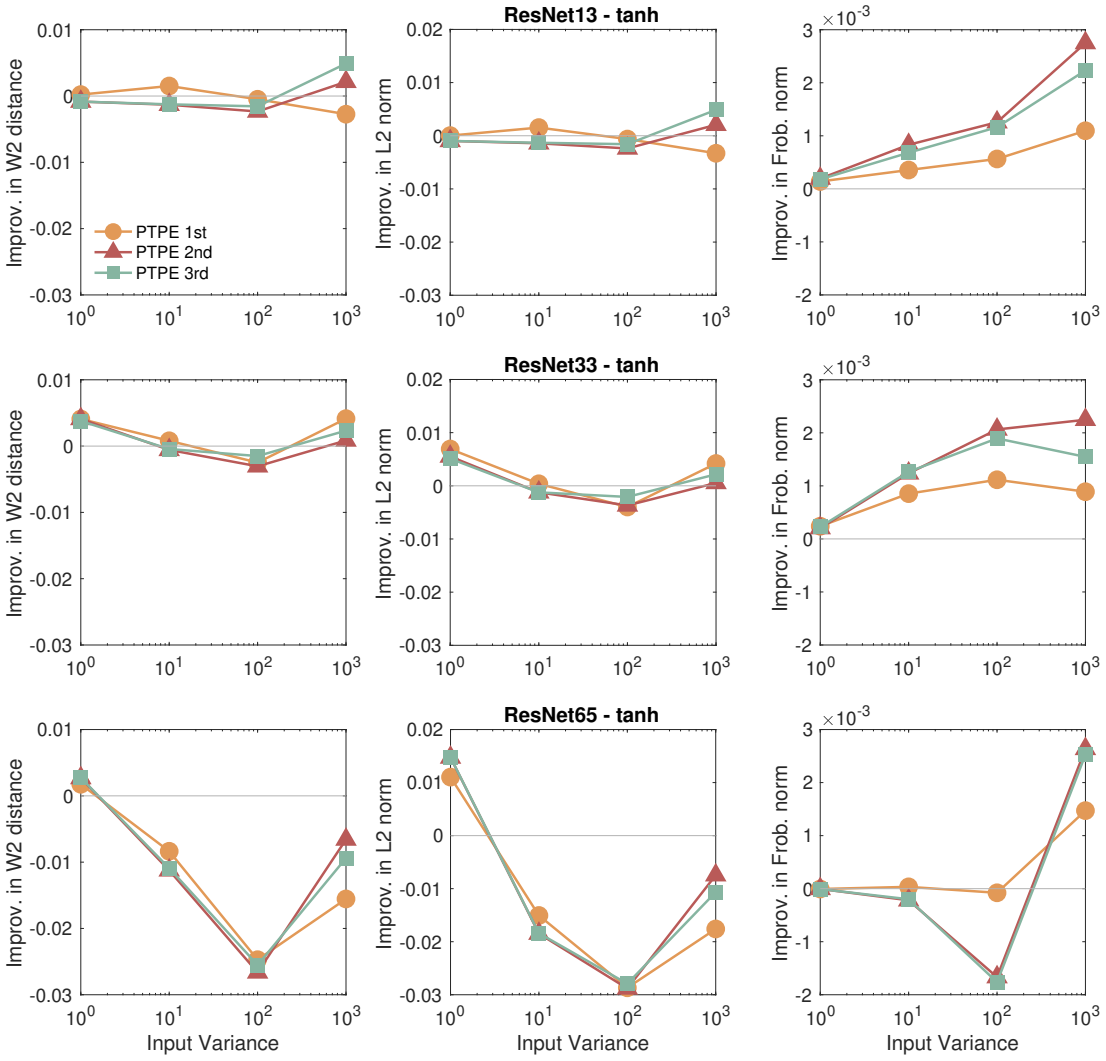

Figure 11: Improvements in the three metrics—Wasserstein-2 distance, Euclidean distance, and Frobenius norm—when using eight scaling factors (γ_j) to approximate the **tanh** function, compared to using four, are reported. Positive values indicate improvement, while negative values denote deterioration. Notably, there is no clear dichotomy in predictive accuracy, suggesting that the approximation error of our method is not a dominant factor. Alternatively, Monte Carlo sampling with 10^6 data points is insufficient to accurately estimate the true covariance of a 10-dimensional multivariate distribution. As a result, the reference moments may deviate from the true values. We argue This finding supports the justification for using four scaling factors rather than a larger number.

A.13 Comparison with Gauss-Hermite Quadrature

Gauss-Hermite quadrature (GHQ) is a numerical integration method used to approximate integrals of the form:

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

where w_i are scalar weights, and n is the number of quadrature points x_i (also referred to as sigma points). In this context, $f(\cdot)$ can represent a nonlinearity or its square, corresponding to the mean and covariance integrals. While infinitely many linear combinations of $f(x)$ can approximate this integral, GHQ selects quadrature points and weights based on Hermite polynomials. These polynomials are orthogonal, making GHQ computationally efficient. This method also serves as the foundation of the cubature Kalman filter (CKF).

We compare the predictive accuracy of PTPE and GHQ across the first four orders (Fig. 12 to 15), using absolute error as the evaluation metric against a reference mean and variance obtained from 10^7 Monte Carlo samples. Since Monte Carlo estimates with 10^7 samples typically fluctuate on the order of 10^{-4}, absolute errors of magnitude $\leq 10^{-4}$ are considered negligible.

PTPE outperforms GHQ in the first three orders. At the fourth order, PTPE provides evidently more accurate estimates than GHQ when the input variance is 1 or higher. Notably, 4th-order PTPE surpasses 4th-order GHQ on ReLU and LeakyReLU even at a variance of 0.1.

To determine how many GHQ orders are needed to surpass 4th-order PTPE, we compare 10th-, 20th-, 30th-, and 40th-order GHQ to 4th-order PTPE (Figures 16 to 19). There is no clear threshold at which GHQ consistently outperforms 4th-order PTPE, as performance depends on the type of nonlinearity. Among the seven types, PTPE is particularly effective for ReLU and LeakyReLU. However, we observe a general trend: higher-order GHQ is required to match PTPE, particularly at higher input variances.

In summary, although GHQ efficiently selects sampling points, it requires an increasing number of points for accurate estimation at higher input variances. More critically, the number of GHQ sampling points grows rapidly with dimensionality, making high-dimensional integration computationally expensive. In contrast, PTPE's complexity is at most $\mathcal{O}(n^2)$, making it more scalable. However, GHQ is expected to outperform PTPE in narrow but deep neural networks. Unlike PTPE, GHQ does not require layer-wise propagation of moments. Furthermore, in narrow networks, the Gaussianity assumption of hidden layers often breaks down, making it difficult for PTPE to provide accurate estimates.

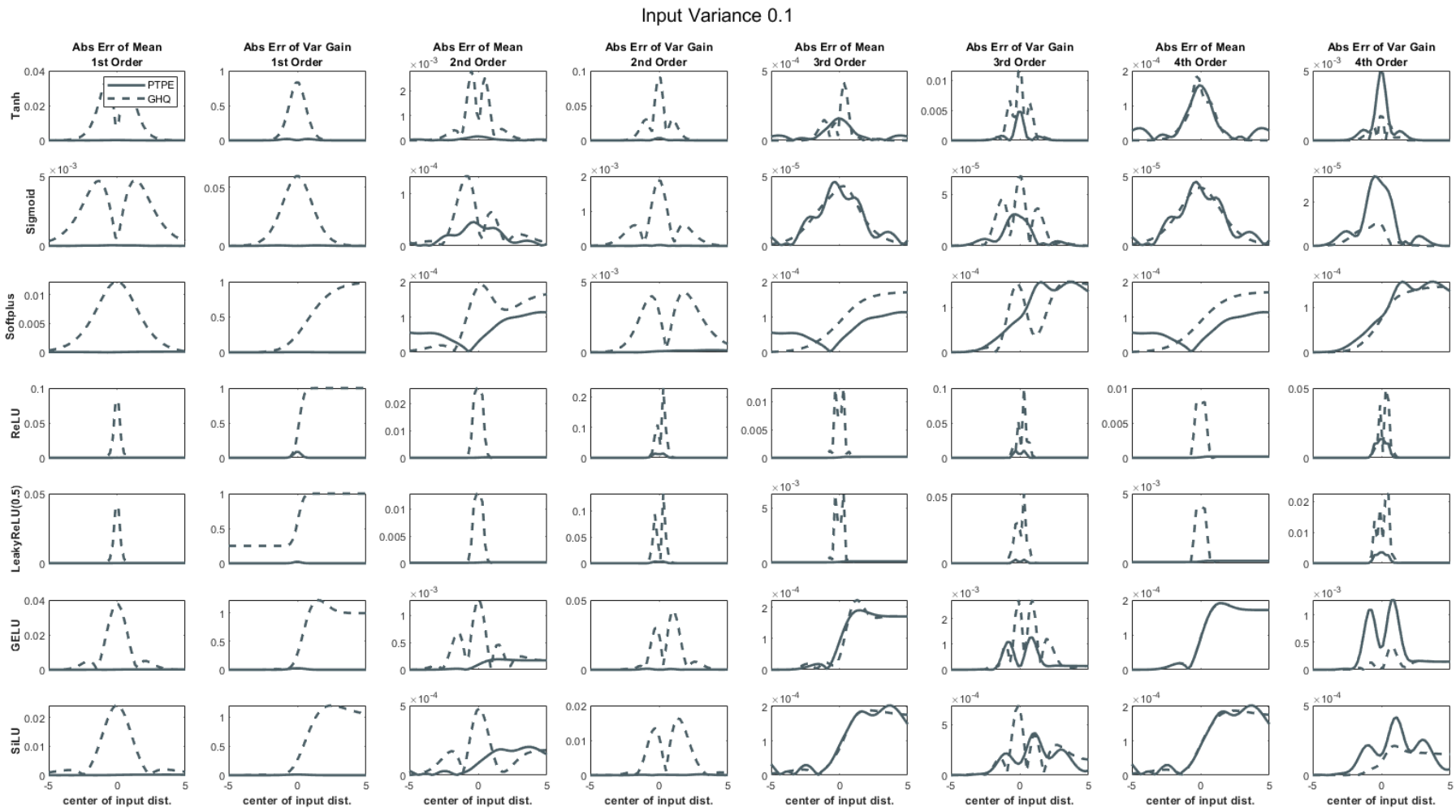

Figure 12: Comparing the absolute approximation error of PTPE and GHQ across the first four orders. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 0.1.

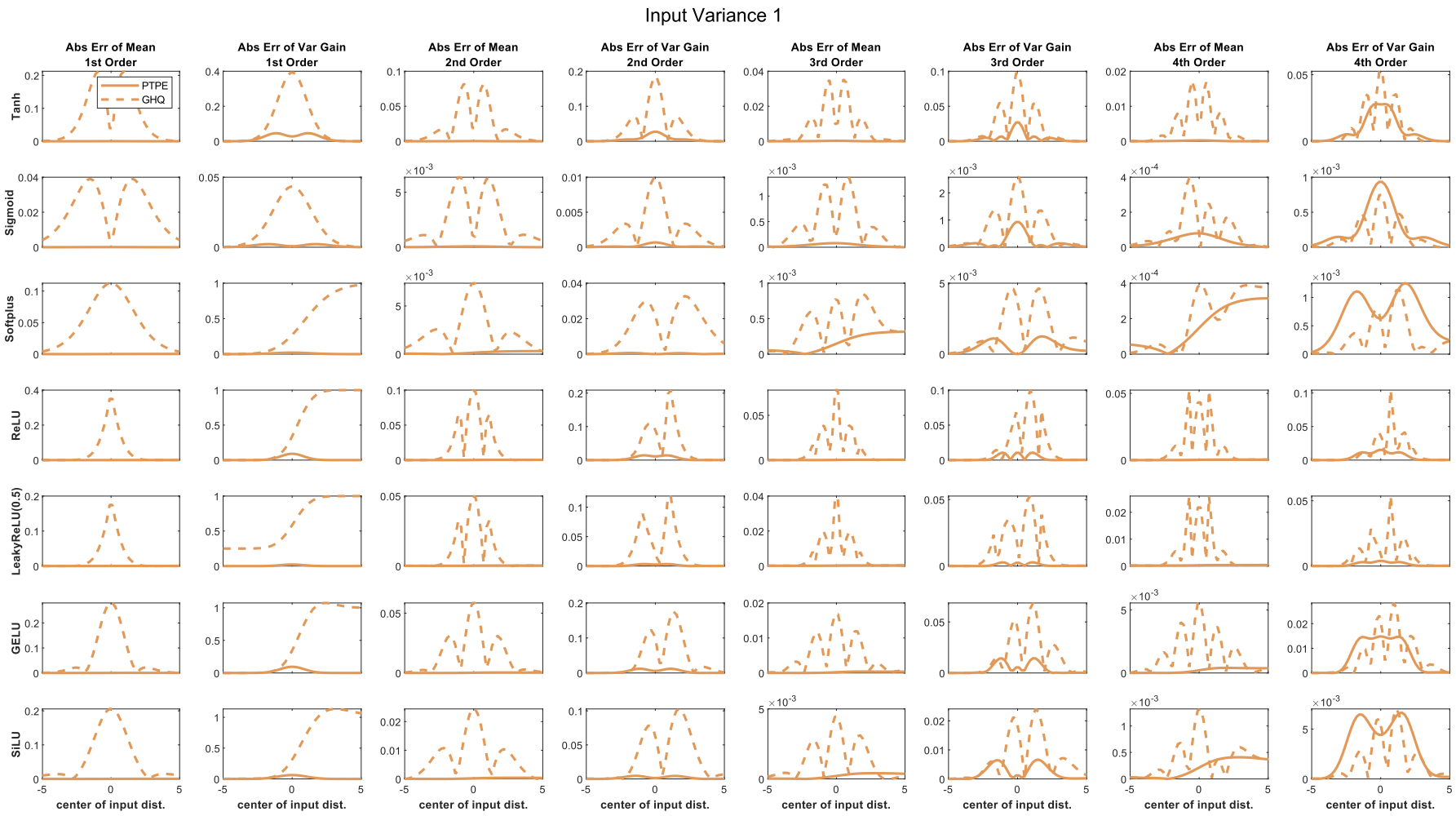

Figure 13: Comparing the absolute approximation error of PTPE and GHQ across the first four orders. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 1.

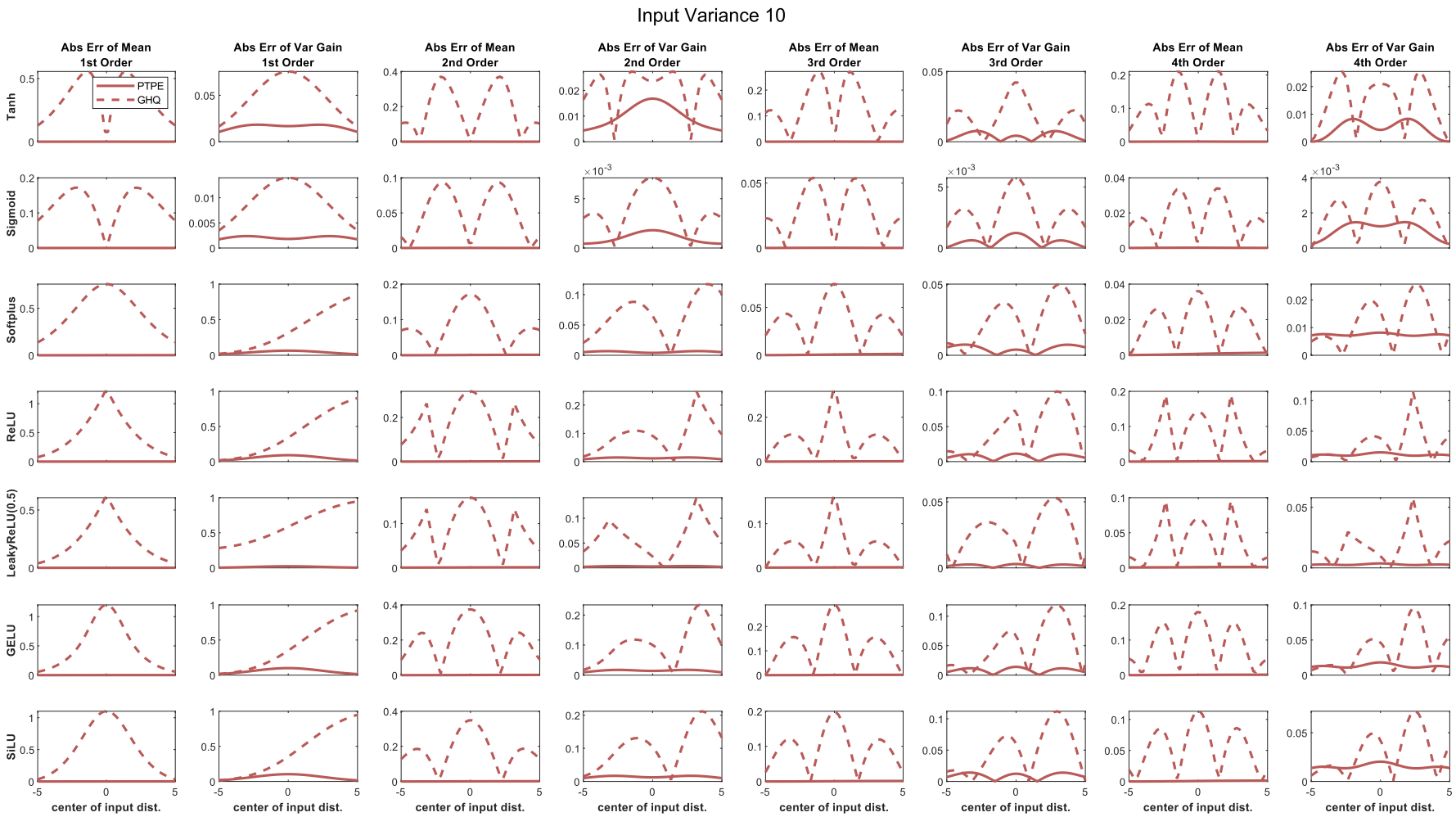

Figure 14: Comparing the absolute approximation error of PTPE and GHQ across the first four orders. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 10.

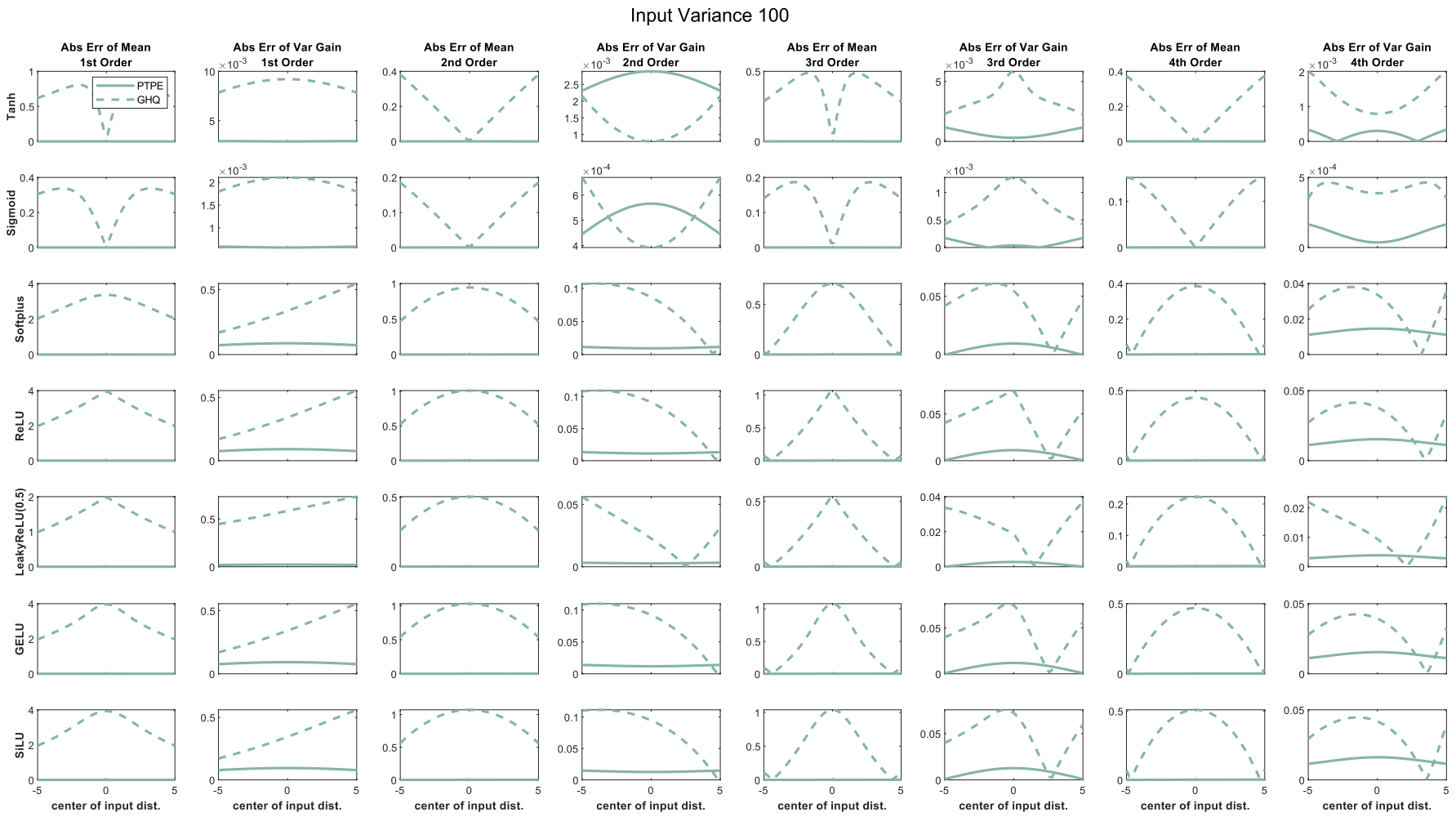

Figure 15: Comparing the absolute approximation error of PTPE and GHQ across the first four orders. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 100.

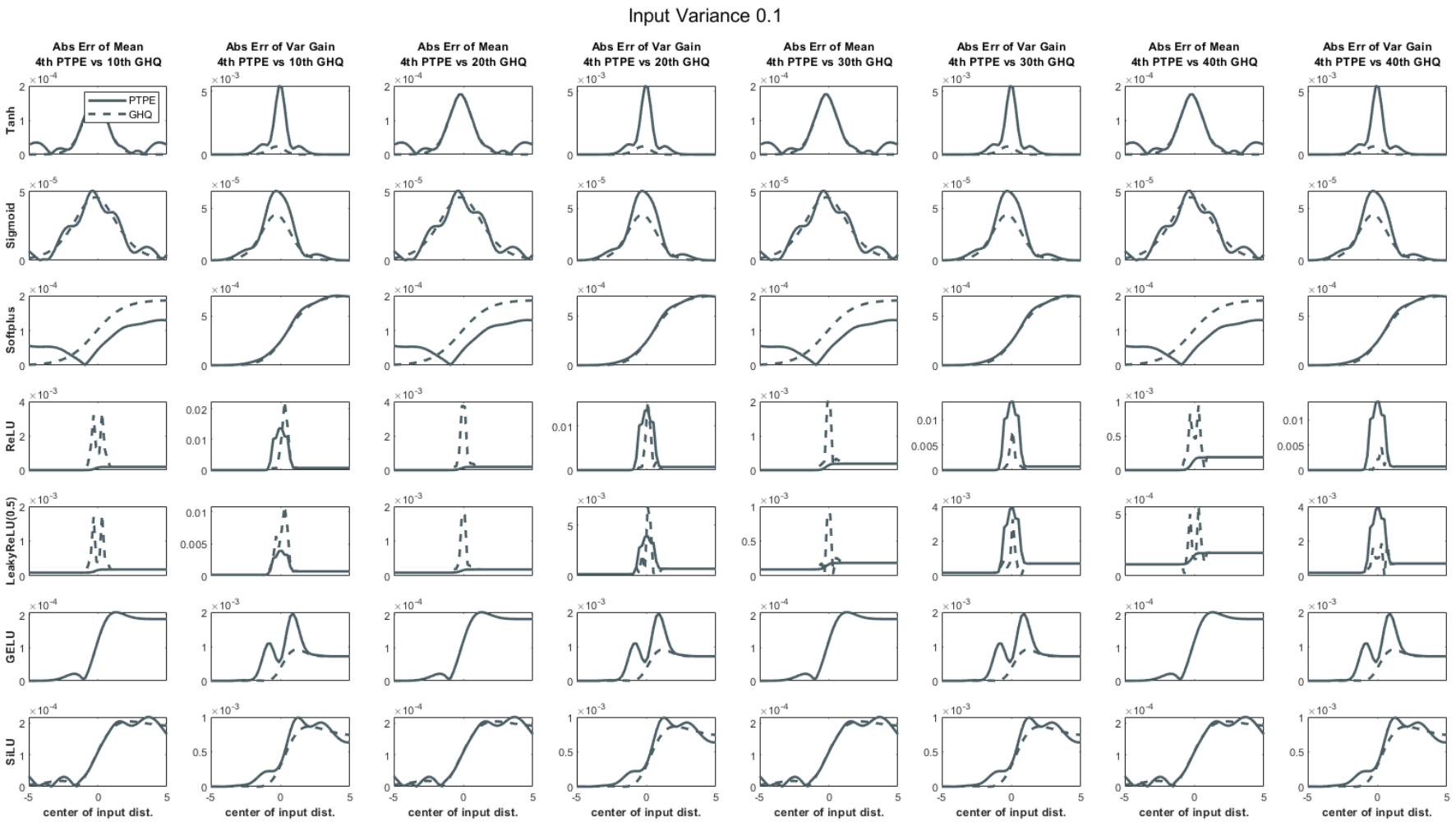

Figure 16: Comparing the absolute approximation error of 4th-order PTPE and 10th-, 20th-, 30th-, and 40th-order GHQ. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 0.1.

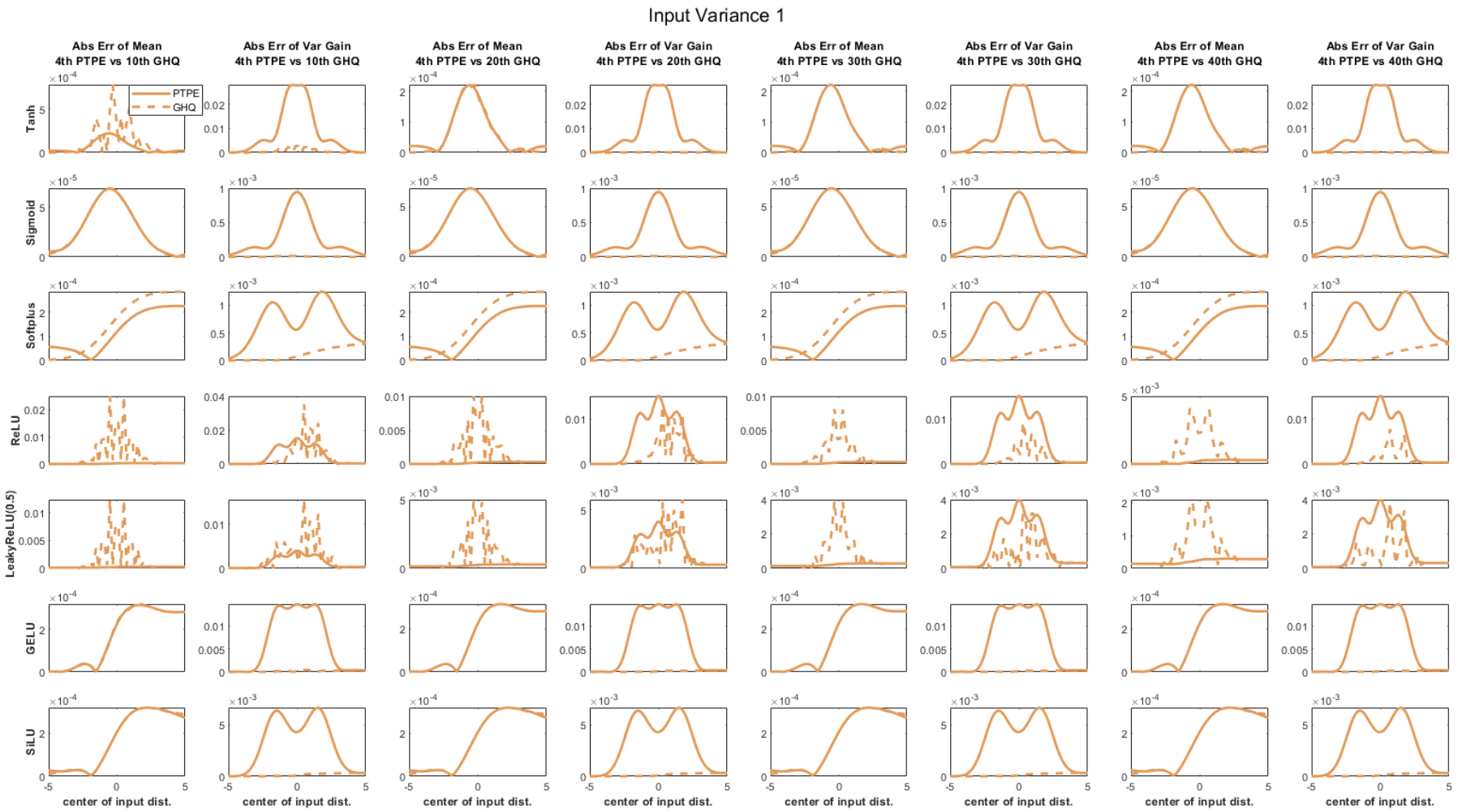

Figure 17: Comparing the absolute approximation error of 4th-order PTPE and 10th-, 20th-, 30th-, and 40th-order GHQ. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 1.

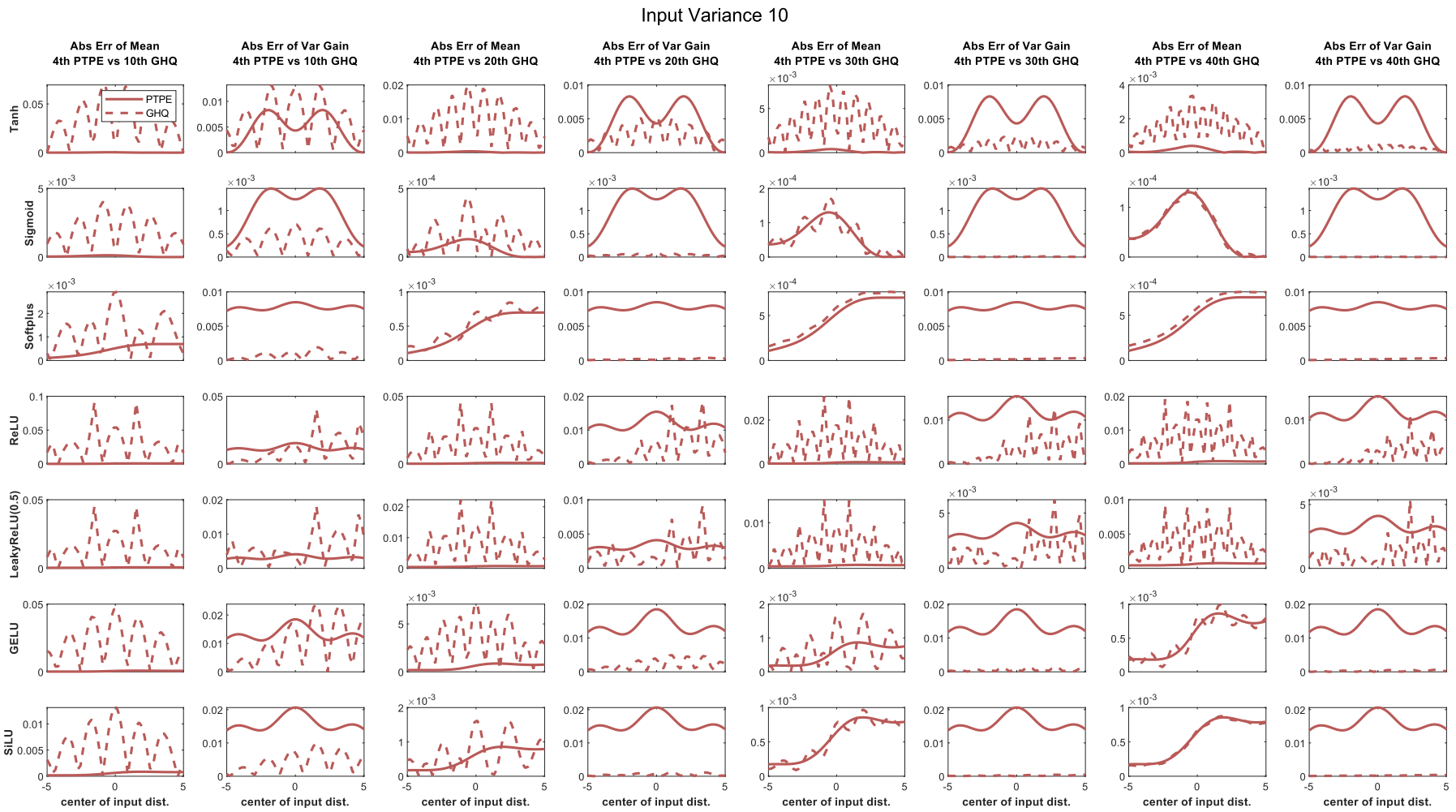

Figure 18: Comparing the absolute approximation error of 4th-order PTPE and 10th-, 20th-, 30th-, and 40th-order GHQ. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 10.

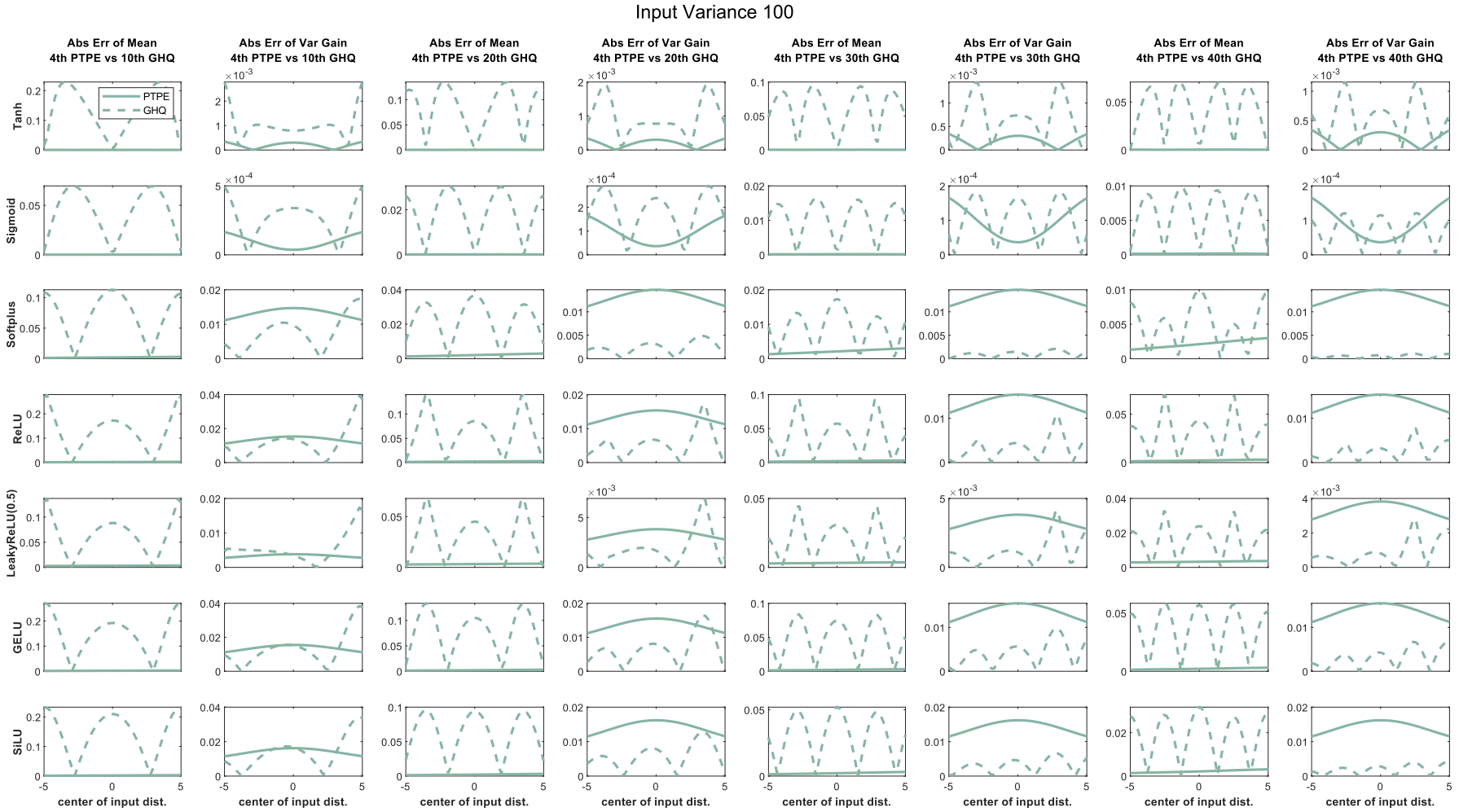

Figure 19: Comparing the absolute approximation error of 4th-order PTPE and 10th-, 20th-, 30th-, and 40th-order GHQ. The odd number of columns show the error of mean, and even number of columns show the error of variance, the smaller the better. In this panel, all input variance is 100.