Input uncertainty propagation through trained neural networks

Paul Monchot, Loïc Coquelin, Sébastien J Petit, Sébastien Marmin, Erwan Le Pennec, Nicolas Fischer

Input uncertainty propagation through trained neural networks

Paul Monchot [1 2] **Loïc Coquelin** [2] **Sébastien J. Petit** [2] **Sébastien Marmin** [2] **Erwan Le Pennec** [1] **Nicolas Fischer** [2]

Abstract

When physical sensors are involved, such as image sensors, the uncertainty over the input data is often a major component of the output uncertainty of machine learning models. In this work, we address the problem of input uncertainty propagation through trained neural networks. We do not rely on a Gaussian distribution assumption of the output or of any intermediate layer. We propagate instead a Gaussian Mixture Model (GMM) that offers much more flexibility using the Split&Merge algorithm. This paper's main contribution is the computation of a Wasserstein criterion to control the Gaussian splitting procedure for which theoretical guarantees of convergence on the output distribution estimates are derived. The methodology is tested against a wide range of datasets and networks. It shows robustness, and genericity and offers highly accurate output probability density function estimation while maintaining a reasonable computational cost compared with the standard Monte Carlo (MC) approach.[1]

1. Introduction

Predictive uncertainty results from the combination of numerous sources. The most studied uncertainty source in the literature is the weight uncertainty (known in the community by epistemic uncertainty (Der Kiureghian & Ditlevsen, 2009), translating the lack of data (i.e. knowledge) in the training dataset. Among the most popular approaches, we can cite methods such as Monte Carlo Dropout (Gal & Ghahramani, 2016), variational inference with Bayes By Backprop (Blundell et al., 2015), or based on Markov Chain Monte Carlo (Welling & Teh, 2011; Chen et al., 2014). The second source of uncertainty in a deep neural network is the inherent uncertainty of the data composing the training dataset (known in the community as aleatoric uncertainty, and that cannot be reduced by increasing the number of training points). Proposed methods rely on adapted loss functions (Kendall & Gal, 2017), quantile regression (Tagasovska & Lopez-Paz, 2019), or ensemble distillation (Malinin et al., 2019) and try to capture the uncertainty induced by noisy trainable data points at inference time. This allows, in particular, to improve the prediction performance and increase the detection of out-of-distribution samples. The scope of this paper is to assess the predictive uncertainty induced by the input uncertainty (of the test data point) at inference time for a not-editable trained neural network. This amounts to estimating the output probability density function (PDF) for new samples not seen by the neural network. Such problem arises especially in a laboratory setting to investigate the robustness of trained neural networks to corrupted inputs. Among many other examples, it can be used in the field of autonomous driving to develop neural networks capable of learning proper driving maneuvers and thus be highly resilient to various types of input noise (brightness, blur, vibrations, sensor noise, ...). Equally, this issue is raised in an embedded context, where time constraints prevent MC sampling. Our proposed method allows for distribution propagation while staying within a designated time budget, producing an estimate and an upper bound of the error. Literature on this problem is less abundant and lacks generic methods with mathematical guarantees over the estimation.

We suggest building upon a long history of the Split&Merge paradigm to propagate a distribution through non-linear models. This paradigm was originally proposed by Sorenson & Alspach (1971), who use Kalman filters for linear filtering problems, with an extension to non-linear problems in (Alspach & Sorenson, 1972). This method was taken up again and improved as time went by. We can note in particular the integration of Unscented Transform (UT) sampling (Julier & Uhlmann, 2004) by Faubel et al. (2009), an update of adaptive weights in (Terefjanu, 2011), and finally the integration of an entropy-based criterion for nonlinear dynamical systems in (DeMars et al., 2013). This paradigm will be detailed in Section 3.2. The proposed methodology

[1]CMAP, CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France [2]Data Science and Uncertainty Department, National Laboratory of Metrology and Testing, Paris, France. Correspondence to: Paul Monchot <paul.monchot@lne.fr>, Paul Monchot <paul.monchot@polytechnique.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]The code to reproduce the experiments is available at https://github.com/PaulMcht/WGMprop.

uses a mixture of Gaussian distributions propagated through the neural network via a splitting procedure controlled by a Wasserstein distance criterion (see, e.g., Villani, 2009, Sect. 7.1). A mixture of Gaussian distributions allows one to estimate the propagated PDF faithfully. The introduction of the Wasserstein distance criterion offers a sound and suitable criterion for detecting non-linearity in neural networks. As a direct result, it helps control the number of Gaussian components in the mixture. Theoretical guarantees of convergence on the output distribution estimates are proposed in Section 3. Finally, extensive experimental results demonstrate that the proposed method performs favorably against the state-of-the art approaches on the MNIST (LeCun, 1998), CIFAR10 (Krizhevsky et al., 2009), CIFAR100 (Krizhevsky et al., 2009), and Camelyon (Litjens et al., 2018) datasets.

2. Related Work: Input uncertainty propagation through neural networks

Uncertainty propagated through a neural network under Gaussian assumption. This approach propagates the input uncertainty by modeling the input and output distributions as Gaussian. Following this approach, Astudillo & Neto (2011) propagate the first two moments (through non-linear layers) in a layer-wise fashion using UT sampling (Julier & Uhlmann, 2004). In their work, Abdelaziz et al. (2015) use a similar approach with additionally an analytical propagation through the sigmoid activation function alongside the propagation of uncertainty in a full network fashion using UT sampling. Later, Gast & Roth (2018) propagate the first two moments analytically while assuming diagonal covariance matrices, thereby neglecting the correlations within the layers (called LPN for Lightweight Probabilistic Network). Finally, Titensky et al. (2018) propagate the two first moments layer-by-layer using Extended Kalman Filtering. If relying on shape assumptions to model the output distribution helps make the methods scalable, it does not guarantee an accurate estimation of the output PDF. Indeed, when it comes to studying input noise with a complex covariance matrix structure, the unimodal Gaussian assumption over the neural network output does not hold, the output density being often multimodal or asymmetric. Thus, the uncertainty quantification of the prediction induced by the input distribution cannot be summarized with a single estimate (Cox & O'Hagan, 2022), but rather by its entire PDF, which, when accurately estimated, allows, among other things, to derive a $\alpha\%$ prediction interval, PI_α, within which falls a new prediction with a $\alpha\%$ probability.

Uncertainty propagated through neural networks using Gaussian mixture model (GMM). The first attempt to propagate a Gaussian mixture (GM) distribution through a neural network $f\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ using the Split&Merge paradigm (see Section 3.2 for details) was made by Zhang & Shin (2021). For a Gaussian variable $X \in \mathbb{R}^d$, the authors measure the similarity between $f(X)$ and a first-order Taylor approximation $\overline{f}(X)$ using the Kullback-Leibler divergence of $f(X)$ from $\overline{f}(X)$. Section 3.2 details how the distribution of X is *split* into a mixture of *smaller* Gaussians when the value of this divergence exceeds some threshold. The Kullback-Leibler divergence is expressed as an integral involving the densities of the distributions of $f(X)$ and $\overline{f}(X)$ with respect to some dominant measure. Unfortunately, expressing the distribution of $f(X)$ with a density is, in general, intractable for a neural net f. To circumvent this issue, Zhang & Shin (2021) use recursively the Split&Merge paradigm to propagate the distribution through the linear layers and the activation functions. Propagating (mixtures of) Gaussian distributions through linear layers is straightforward. For a smooth invertible activation function $\sigma\colon \mathbb{R}^q \to \mathbb{R}^q$ (acting typically componentwise), Zhang & Shin (2021) derive an upper bound on the Kullback-Leibler divergence using the formula $p(\sigma(A)) = \left|J_\sigma(A)^{-1} p(A)\right|$, with $J_\sigma(A)$ the Jacobian matrix of σ. However, the Rectified Linear Unit (ReLU, Maas et al., 2013) activation function is not invertible and, more fundamentally, the Kullback-Leibler divergence of a rectified Gaussian distribution from a continuous random variable is always infinite. The authors bypass this issue using the approximation given by a Leaky ReLU activation function with slope $\delta \ll 1$. However, Section 4 will show experimentally that this approach is impracticable in the presence of a noise significantly affected by the nonlinearity of the network. This is due to the fact that a Leaky ReLU activation function with slope $\delta \ll 1$ still concentrates a potentially significant fraction of probability mass in an interval of the form $[-O(\delta), 0]$. In this case, the Kullback-Leibler divergence of the leakily rectified Gaussian from a Gaussian with a variance larger than $O(\delta)$ becomes very large. Consequently, the recursive Split&Merge paradigm leads to an impractically high number of splits as soon as the first (Leaky) ReLU activation function is encountered. For these reasons, we were not able to implement their methodology in our experimental settings (Appendix A presents theoretical elements to support it). The Wasserstein-based criterion proposed in Section 3.3 helps crush these limits by: 1) not requiring the use of densities; 2) being less sensitive to small probability events; 3) making it possible to use the Split&Merge paradigm on the whole network; and 4) ensuring theoretical convergence guarantees given in Section 3.3.3.

3. Proposed Method (WGMprop)

In this section, we first define the notations that will be used in the rest of the paper, then the classical Split&Merge algorithm for Gaussian mixtures is introduced in section 3.2. Section 3.3 exposes the motivations for using the new cri-

terion based on the Wasserstein distance. Theoretical guarantees derived for this new criterion are presented in section 3.3.3 and finally in section 3.4, the practical implementation of the method.

3.1. Framework

Write $\boldsymbol{U} \in \mathbb{R}^d$ the multidimensional random variable of probability distribution $P_{\boldsymbol{U}}$ and with mean $\mu_{\boldsymbol{U}}$ and covariance matrix $\Sigma_{\boldsymbol{U}}$.

Let $\mathbb{M}$ be the space of mixtures of Gaussian distributions and, for $\boldsymbol{U}$ such that $P_{\boldsymbol{U}} \in \mathbb{M}$, write $P_{\boldsymbol{U}} = \sum_{i=1}^{M} w_i P_{\boldsymbol{U},i}$, with non-negative w_is summing to one and $P_{\boldsymbol{U},i} = \mathcal{N}(\mu_{\boldsymbol{U},i}, \Sigma_{\boldsymbol{U},i})$, and M the number of components in the mixture. We will keep the symbols M and w_i generic for any mixture, unless explicitly stated otherwise.

Furthermore, let $S^{[1]}(\cdot; n)$ be a splitting operator defined on the space of Gaussian distributions and extended to Gaussian mixtures by linearity. $S^{[1]}(\cdot; n)$ approximates $P_{\boldsymbol{U}}$ by a Gaussian mixture distribution of n components:

$$P_{\hat{\boldsymbol{U}}^{[1]}} = S^{[1]}(P_{\boldsymbol{U}}; n) = \sum_{i=1}^{n} w_i P_{\hat{\boldsymbol{U}}^{[1]},i}$$

Given some $\boldsymbol{X} \in \mathbb{R}^d$, the goal is to propagate $P_{\boldsymbol{X}}$ through a $L_{2,2}$-Lipschitz continuous neural network $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^{d'}$. (More precisely, all the activation functions of $\boldsymbol{f}$ are assumed to be Lipschitz continuous.) Leveraging the approximation power of mixtures of Gaussian distributions (see, e.g., Scott, 2015), we suppose from now on that $\boldsymbol{X} \in \mathbb{R}^d$ is a multivariate normal random variable.

Using the previous notations—omitting the dependence of M on s and n—, the s^{th} iteration

$$P_{\hat{\boldsymbol{X}}^{[s]}} = S^{[s]}(P_{\boldsymbol{X}}; n) = \sum_{i=1}^{M} w_i P_{\hat{\boldsymbol{X}}^{[s]},i}$$

of $S^{[1]}$ on $P_{\boldsymbol{X}}$ will play a major role in this work.

Finally, let $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}^{[s]}$ the random variables such that $\boldsymbol{Y} = f(\boldsymbol{X})$, $\hat{\boldsymbol{Y}}^{[s]} = f(\hat{\boldsymbol{X}}^{[s]})$ and let $\tilde{\boldsymbol{Y}}^{[s]}$ the random variable obtained by propagating each component of $P_{\hat{\boldsymbol{X}}^{[s]}}$ through $\boldsymbol{f}$ by moment matching.

3.2. Classical Split&Merge for Gaussian Mixture Model

The Split&Merge algorithm is an iterative process built on 4 steps. At iteration $s \in \mathbb{N}$, with $P_{\hat{\boldsymbol{X}}^{[0]}} = P_{\boldsymbol{X}}$, we apply the following steps for each component $P_{\hat{\boldsymbol{X}}^{[s]},i}$ of $P_{\hat{\boldsymbol{X}}^{[s]}}$:

- step1/**propagation**: it propagates, under linear assumption, the component through $\boldsymbol{f}$. Since normal distributions are stable by linear operation, the propagation step is reduced to the estimation of the output first two moments of such Gaussian, from which we obtain the estimate $P_{\tilde{\boldsymbol{Y}}^{[s]},i}$ obtained by moment matching of the true output distribution $P_{\hat{\boldsymbol{Y}}^{[s]},i}$. The moments can be estimated by sampling techniques or analytically derived when formulas are available.

- step2/**non-linearity detection**: it quantifies how strongly the Gaussian distribution $P_{\hat{\boldsymbol{X}}^{[s]},i}$ was affected by the non-linearity during the propagation utilizing a thresholded discrepancy measure (threshold noted T_{split}).

- step3/**splitting**: it splits the input Gaussian distribution $P_{\hat{\boldsymbol{X}}^{[s]},i}$ using a splitting operator $S(\cdot, n)$ in case the local linearity assumption is invalidated in step2, then restarts from step1 on the split components, otherwise $P_{\hat{\boldsymbol{X}}^{[s]},i}$ remains unchanged. The splitting of a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ consists in splitting the distribution in a specific single direction, following the formula $S^{[1]}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}); n) = \sum_{i=1}^{n} w_i \mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$ with:

$$w_i = w\tilde{w}_i,\ \boldsymbol{\mu_i} = \boldsymbol{\mu} + \tilde{\beta}_i \sqrt{\lambda_j}\boldsymbol{\nu_j}, \quad (1)$$

$$\tilde{\boldsymbol{\Lambda}} = \text{diag}(\lambda_1, \ldots, \tilde{\sigma}^2\lambda_j, \ldots, \lambda_R) \quad (2)$$

  where $(\lambda_r)_{1 \le r \le R}$ are the eigenvalues of $\boldsymbol{\Sigma}$, j is the splitting direction[2], $\tilde{w}_i$, $\tilde{\beta}_i$ and $\tilde{\sigma}$ are fixed parameters (cf Appendix E)[3]. The split operator introduces an approximation at each iteration, related to the standard error $\epsilon_{0,1} = W_2(\mathcal{N}(0,1), S^{[1]}(\mathcal{N}(0,1); n)) > 0$. However, we can find a Gaussian mixture rendering $\epsilon_{0,1}$ arbitrarily small, as long as a sufficient number of components, with a reduced variance, are used in the mixture. The splitting procedure stops when the criterion computed in step2 becomes smaller than the threshold.

- step4/**merging**: it merges similar components using a thresholded discrepancy measure to reduce the number of components of the output estimated PDF $P_{\tilde{\boldsymbol{Y}}^{[s]}}$. This step is not applied in our work.

The iterative process stops when all components of the input mixture distribution satisfy the local linearity hypotheses made in step1. Provided this happens at iteration s, the resulting GMM $P_{\hat{\boldsymbol{X}}^{[s]}}$, approximate of $P_{\boldsymbol{X}}$, is given by $P_{\hat{\boldsymbol{X}}^{[s]}} = S^{[s]}(P_{\boldsymbol{X}}; n) = \sum_{i=1}^{M} w_i P_{\hat{\boldsymbol{X}}^{[s]},i} = \sum_{i=1}^{M} w_i \mathcal{N}(\mu_{\hat{\boldsymbol{X}}^{[s]},i}, \Sigma_{\hat{\boldsymbol{X}}^{[s]},i})$. Then, by propagating each component of $P_{\hat{\boldsymbol{X}}^{[s]}}$ by moment matching, one can easily obtain an estimate of $P_{\boldsymbol{Y}}$, denoted $P_{\tilde{\boldsymbol{Y}}^{[s]}}$, with $P_{\tilde{\boldsymbol{Y}}^{[s]}} =$

[2] the axis along which the Gaussian mixture PDF terms are split

[3] It is worth noting that all components split the same amount of time have the same covariance matrix, and so the same eigenvalues

$\sum_{i=1}^{M} w_i P_{\tilde{\boldsymbol{Y}}^{[s]},i}$.
The essential matter when using this iterative algorithm is the choice of the criterion acting as the non-linearity detector. In the following section, we present a criterion based on the Wasserstein distance.

3.3. The Wasserstein metric as a splitting criterion

3.3.1. The Wasserstein metric

The Wasserstein metric (see, e.g., Villani, 2009, p.118 and references therein) is a popular tool used notably in transportation theory. Given a strictly positive integer p, the Wasserstein distance between the measures P and Q is defined as

$$W_p(P, Q) = \left(\inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(X,Y)\sim\gamma} \left(\|X - Y\|^p \right) \right)^{1/p}, \tag{3}$$

where Γ is the set of probability measures on $\mathbb{R}^d \times \mathbb{R}^d$ having marginals P and Q.

Wasserstein distances enjoy several appealing properties. First, they are proper distances and satisfy, in particular, the triangle inequality. It will be beneficial for computing distances between mixtures of Gaussian distributions. Second, they also have interesting theoretical properties. Indeed, it is well known (see, e.g., Villani, 2009, Proposition 7.29) that the p^{th} Wasserstein distance controls the moments up to the p-th order. Therefore, when $p \geq 2$, $W_p(\cdot,\cdot)$ can be used to bound the error on the first two moments, which are sometimes a primary interest.

Proposition 3.1. *Using the previous notations, we have*

$$\left\| \mu_{\boldsymbol{Y}} - \mu_{\tilde{\boldsymbol{Y}}^{[s]}} \right\| \leq W_p(P_{\boldsymbol{Y}}, P_{\tilde{\boldsymbol{Y}}^{[s]}}).$$

Also, let $\sigma_{\boldsymbol{Y},i}$ and $\sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}$ be the standard deviations of the i-th marginals of $P_{\boldsymbol{Y}}$ and $P_{\tilde{\boldsymbol{Y}}^{[s]}}$. If $p \geq 2$, then

$$\left| \sigma_{\boldsymbol{Y},i} - \sigma_{\tilde{\boldsymbol{Y}}^{[s]},i} \right| \leq W_p(P_{\boldsymbol{Y}}, P_{\tilde{\boldsymbol{Y}}^{[s]}})$$

The proof of proposition 3.1 can be found in appendice D.

Furthermore, Wasserstein distances metrize weak convergence (see, e.g., Villani, 2009, Theorem 6.9). Consequently, the convergence in the Wasserstein sense implies the convergence of many interesting statistical quantities. The following proposition shows notably that Wasserstein distances bound the errors on the marginal quantile functions.

Proposition 3.2. *Let $Q_{\boldsymbol{Y}}$ (respectively $Q_{\tilde{\boldsymbol{Y}}^{[s]}}$) be quantile functions of Y (respectively $\tilde{Y}^{[s]}$), then:*

$$\frac{1}{\sqrt{d}} \sum_{i=1}^{d} \int_0^1 |Q_{\boldsymbol{Y}}(q) - Q_{\tilde{\boldsymbol{Y}}^{[s]}}(q)| dq \leq W_p(P_{\boldsymbol{Y}}, P_{\tilde{\boldsymbol{Y}}^{[s]}})$$

The proof of proposition 3.2 can be found in appendice D.

This can be interpreted as a bound on the average quantiles of the marginal distribution. Using the Wasserstein distance does not allow bounding an arbitrary quantile in general. Unlike the KL metric, Wasserstein distances are relevant for comparing probability measures when the reference distribution is not dominated by the other one. This situation arises, e.g., when comparing $P_{\boldsymbol{Y}}$ and $P_{\tilde{\boldsymbol{Y}}^{[s]}}$ for a neural network using ReLU activation functions. As detailed in Appendix A, in this setting, the KL distance between the distribution is always $+\infty$ which makes it impossible to obtain any bounds. Even in the leaky ReLU setting, the bounds that could be obtained would be meaningless when the leaky ReLU becomes close to the classical one.

3.3.2. Upper bound criterion

We propose to rely on a 2-Wasserstein distance criterion between $P_{\boldsymbol{Y}}$ and $P_{\tilde{\boldsymbol{Y}}^{[s]}}$ for the non-linearity detection. Computing directly the distance is hard, so we will rely on an upper bound obtained by introducing an intermediary distribution $P_{\bar{\boldsymbol{Y}}^{[s]}}$ defined by linearizing the function $\boldsymbol{f}$.

Proposition 3.3. *Let $W_2(\cdot,\cdot)$ the 2-Wasserstein distance with respect to the usual Euclidean norm on $\mathbb{R}^d$. Then, for $s \in \mathbb{N}$ and after applying $S^{[s]}(\cdot, n)$ to $P_{\boldsymbol{X}}$, we have:*

$$W_2(P_{\boldsymbol{Y}}, P_{\tilde{\boldsymbol{Y}}^{[s]}}) \quad \leq \quad B_s, \tag{4}$$

with

$$B_s = L_{2,2}\epsilon_{0,1} \sum_{k=0}^{s} \sqrt{\lambda_{\infty}^{[k]}} + \tag{5}$$

$$\sum_i w_i E_{\boldsymbol{X}\sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(\boldsymbol{X}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X})\|_2^2 \right]^{\frac{1}{2}} +$$

$$\left[\|\mu_{\tilde{\boldsymbol{Y}}^{[s]},i} - \mu_{\bar{\boldsymbol{Y}}^{[s]},i}\|_2^2 + \right.$$

$$\left. \mathrm{Tr}\left(\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} + \Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i} - 2\left(\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}^{\frac{1}{2}} \Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} \Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}^{\frac{1}{2}} \right)^{\frac{1}{2}} \right) \right]^{\frac{1}{2}}$$

where $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ is the first order Taylor linearization of $\boldsymbol{f}$ at location $\mu_{\hat{\boldsymbol{X}}^{[s]},i}$ and $\lambda_{\infty}^{[k]}$ is the highest eigenvalue of $\Sigma_{\hat{\boldsymbol{X}}^{[k]},\boldsymbol{i}}$ after applying $S^{[k]}(\cdot, n)$ to $P_{\boldsymbol{X}}$. $\mu_{\tilde{\boldsymbol{Y}}^{[s]},i}$ and $\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}$ (respectively $\mu_{\bar{\boldsymbol{Y}}^{[s]},i}$ and $\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i}$) are the first two moments of $P_{\tilde{\boldsymbol{Y}}^{[s]},i}$ (respectively $P_{\bar{\boldsymbol{Y}}^{[s]},i}$). $L_{2,2}$ is the Lipschitz constant (w.r.t. to the L_2 norms) of $\boldsymbol{f}$, and $\epsilon_{0,1}$ is the standard error of the splitting operator S.

The proof of proposition 3.3 can be found in appendice D.2.2.

The local linearity hypothesis is here used in two distinct ways. The first is by first-order Taylor linearization, and the second using the Gaussian stability by linear application. The upper bound is composed of three distinct parts.

The first part translates the output error due to the error performed by splitting the input distribution ($\epsilon_{0,1} > 0$). The second part evaluates the error due to the approximation after Taylor linearization. The final part quantifies the moment drifting between $P_{\tilde{\boldsymbol{Y}}^{[s]}}$ and $P_{\bar{\boldsymbol{Y}}^{[s]}}$ induced by the linearization hypothesis. We can note that no analytical formula permits us to compute this upper bound. However, techniques to estimate this quantity are presented in Section 3.4.

3.3.3. Theoretical guarantees

We show here the convergence of this upper bound in Proposition 3.4.

Proposition 3.4. *Let $R = rank(\Sigma_{\boldsymbol{X}})$ and $(\lambda_1, ..., \lambda_R)$ the non-null eigenvalues of $\Sigma_{\boldsymbol{X}}$ in descending order, $\exists C \in \mathbb{R}_+$ such that:*

$$B_s \underset{s\to\infty}{\longrightarrow} L_{2,2}\epsilon_{0,1}\left(C + \frac{R\sqrt{\lambda_R}}{1-\tilde{\sigma}}\right),$$

with $C = \sum_{k=0}^{r_R}\sqrt{\lambda_\infty^{[k]}}$, and $r_R = \sum_{i=1}^{R-1}\left\lceil \frac{\ln\lambda_i - \ln\lambda_R}{\ln\tilde{\sigma}^2}\right\rceil$

This gives immediately the following corollary.

Corollary 3.5. *With the notations from Proposition 3.4:*

$$\limsup W_2(P_{\boldsymbol{Y}}, P_{\tilde{\boldsymbol{Y}}^{[s]}}) \leq L_{2,2}\epsilon_{0,1}\left(C + \frac{R\sqrt{\lambda_R}}{1-\tilde{\sigma}}\right).$$

The proof of proposition 3.4 can be found in appendice D.3.

Applying an infinite time, the split operator guides the criterion's upper bound convergence to a constant. This constant depends on the standard error of the splitting operator $\epsilon_{0,1}$, the Lipschitz constant $L_{2,2}$ of the network $\boldsymbol{f}$, the splitting level n and the initial input uncertainty translated by the initial eigenvalues of $\Sigma_{\boldsymbol{X}}$. We show that this convergence bound depends on the initial error committed when splitting the initial distribution as a Gaussian mixture. $\epsilon_{0,1}$ can be reduced using a split operator with more components (higher n) and smaller variances (lower $\tilde{\sigma}$). The number of iterations of the new scheme will also be reduced, and thus the bound holds with the new $\epsilon_{0,1}$ and the old constant.

3.4. Practical implementation

The following subsections contains the practical implementation details of our proposed method.

3.4.1. Gaussian Splitting

The splitting hyper-parameters from Equations (1)-(2) (provided in Appendix E) were obtained by solving the following optimization problem:

$$\min_{\tilde{w}_i,\tilde{\beta}_i,\tilde{\sigma}} D_{KL}(\mathcal{N}(0,1))\|S^{[1]}(\mathcal{N}(0,1))) + \alpha n\tilde{\sigma}$$
$$\text{subject to } \sum_{i=1}^{n}\tilde{w}_i = 1, \quad 0 < \tilde{\sigma} < 1 \qquad (6)$$

where α is a scaling factor and n stands for the number of components. The splitting direction corresponds here to the direction of the largest uncertainty, i.e. the direction having the highest eigenvalue. Other splitting directions could have been chosen, such as the direction of largest non-linearity (Faubel & Klakow, 2010) or the direction of largest non-Gaussianity (Straka et al., 2016). These last two solutions are not retained in our case because they introduce an additional computational cost.

3.4.2. Moment matching and Upper bound estimation

Unscented Transform: To estimate the upper bound of Proposition 3.3, we propose to rely on UT sampling. UT is a sampling technique which propagates the first two moments of the distribution $P_{\hat{\boldsymbol{X}}^{[s]},i} = \mathcal{N}(\mu_{\hat{\boldsymbol{X}}^{[s]},i}, \Sigma_{\hat{\boldsymbol{X}}^{[s]},i})$ through nonlinear transformation $\boldsymbol{f}$ by using a $2d+1$ weighted samples called sigma points, defined as:

$$Z_{i,[0]} = \mu_{\hat{\boldsymbol{X}}^{[s]},i} \qquad \boldsymbol{\omega}_0 = \frac{\kappa}{n+\kappa}$$
$$Z_{i,[j]} = \mu_{\hat{\boldsymbol{X}}^{[s]},i} + (\sqrt{(d+\kappa)}\boldsymbol{U}_{\hat{\boldsymbol{X}}^{[s]},i})_{[j]} \qquad \boldsymbol{\omega}_j = \frac{1}{2(n+\kappa)}$$
$$Z_{i,[j+d]} = \mu_{\hat{\boldsymbol{X}}^{[s]},i} - (\sqrt{(d+\kappa)}\boldsymbol{U}_{\hat{\boldsymbol{X}}^{[s]},i})_{[j]} \qquad \boldsymbol{\omega}_{j+d} = \frac{1}{2(d+\kappa)}$$

$\forall j \in \{1, \cdots, d\}$, where $\Sigma_{\hat{\boldsymbol{X}}^{[s]},i} = \boldsymbol{U}_{\hat{\boldsymbol{X}}^{[s]},\boldsymbol{i}}\boldsymbol{U}^{\boldsymbol{T}}_{\hat{\boldsymbol{X}}^{[s]},\boldsymbol{i}}$, $(\sqrt{(d+\kappa)}\boldsymbol{U}_{\hat{\boldsymbol{X}}^{[s]},i})_{[j]}$ is the j-th column of $\sqrt{(d+\kappa)}\boldsymbol{U}_{\hat{\boldsymbol{X}}^{[s]},i}$, $\boldsymbol{\omega}_j$ is the weight associated with the j-th sigma point, and κ is a free parameter, usually set as $\kappa = d-3$. The sigma points are used to estimate $\mu_{\tilde{\boldsymbol{Y}}^{[s]},i}$, $\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}$, $\mu_{\bar{\boldsymbol{Y}}^{[s]},i}$ and $\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i}$ by propagation through $\boldsymbol{f}$ and $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ following the procedure detailed in Section 3.4.3.

3.4.3. How to obtain $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$:

Formally, a neural network can be considered as a non-linear parametric function $\boldsymbol{f}(\cdot, \boldsymbol{\theta})$, results of the concatenation of L elementary blocks h_i, each composed by a linear transformation and a non-linear activation function: $\mathbf{Y} = h_{L-1}\left(..h_1\left(h_0\left(\mathbf{X}, \theta_0\right)\right)\right) = \boldsymbol{f}(\mathbf{X}, \boldsymbol{\theta})$.

To obtain $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$, we perform a succession of first order Taylor linearization: $\bar{\boldsymbol{Y}} = \bar{h}_{L-1}\left(..\bar{h}_1\left(\bar{h}_0\left(\mathbf{X}, \theta_0\right)\right)\right) = \bar{\boldsymbol{f}}(\mathbf{X}, \boldsymbol{\theta})$, using the $2d+1$ sigma points $Z_{i,[j]}$ obtained in Section 3.4.2. By noting $Y_{i,[j],l} = h_l(h_{l-1}(..h_0(Z_{i,[j]}, \theta_0)))$ and $\mu_{i,j,l} = \sum_{j=1}^{2d+1} w_i Y_{i,[j],l}$, and $Y_{i,[j],0} = Z_{i,[j]}$ then for the layer l, we obtain $\bar{h}_l$ by first order Taylor linearization:

$$\bar{h}_l(X) = h_l(\mu_{i,j,l-1}) + J_{h_l}(\mu_{i,j,l-1})(X - \mu_{i,j,l-1}) \quad (7)$$

where $J_{h_l}(\mu_{i,j,l-1})$[4] denotes the Jacobian matrix of h_l at

[4]The Jacobian matrix is not defined everywhere for neural networks using ReLU or Leaky ReLU, it is computed in practice using classical computing library (Agrawal et al., 2019)

location $\mu_{i,j,l-1}$.

Finally:

$$\mu_{\bar{\boldsymbol{Y}}^{[s]},i} \approx \sum_{j=1}^{2d+1} \boldsymbol{\omega}_j \bar{\boldsymbol{f}}_{\boldsymbol{i}}(Z_{i,[j]}), \qquad \mu_{\hat{\boldsymbol{Y}}^{[s]},i} \approx \sum_{j=1}^{2d+1} \boldsymbol{\omega}_j \boldsymbol{f}(Z_{i,[j]})$$

$$\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} \approx \sum_{j=1}^{2d+1} \boldsymbol{\omega}_j (\bar{\boldsymbol{f}}_{\boldsymbol{i}}(Z_{i,[j]}) - \mu_{\bar{\boldsymbol{Y}}^{[s]},i})(\bar{\boldsymbol{f}}_{\boldsymbol{i}}(Z_{i,[j]}) - \mu_{\bar{\boldsymbol{Y}}^{[s]},i})^T$$

$$\Sigma_{\hat{\boldsymbol{Y}}^{[s]},i} \approx \sum_{j=1}^{2d+1} \boldsymbol{\omega}_j (\boldsymbol{f}(Z_{i,[j]}) - \mu_{\hat{\boldsymbol{Y}}^{[s]},i})(\boldsymbol{f}(Z_{i,[j]}) - \mu_{\hat{\boldsymbol{Y}}^{[s]},i})^T$$

$$\mathbb{E}_{X \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(X) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(X)\|_2^2 \right] \approx \sum_{j=1}^{2d+1} \omega_j \|\boldsymbol{f}(Z_{i,[j]}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(Z_{i,[j]})\|_2^2$$

3.4.4. High dimensionality

As the proposed methodology considers the neural network as a black box, the scalability bottleneck in terms of memory usage and computation time resides in the input and output dimensions (independently of network width), which can be elevated. This problem is first and foremost encountered in image processing and convolutional neural network architectures. More specifically, our method relies on the inverse of the input covariance matrix $\Sigma_{\boldsymbol{X}}$ with computational complexity in $\mathcal{O}(d^3)$. To improve the scalability of the proposed method, we apply the methodology over the active subspace of $\Sigma_{\boldsymbol{X}}$ (Constantine et al., 2014). The selection of the active subspace consists in reducing the dimension by keeping the r ($r \ll d$) first largest eigenvalues of $\Sigma_{\boldsymbol{X}}$ using algorithms such as the Implicitly Restarted Lanczos Method (Calvetti et al., 1994). To further improve the performances, we can apply a burn-in stage consisting of initially performing s_0 splitting iterations, where s_0 is manually set.

Algorithm 1 presents the full algorithm.

Algorithm 1 WGMprop

Input:
- Input Gaussian distribution, $P_{\boldsymbol{X}}$
- Splitting threshold, T_{split}
- Number of burn-in stages, s_0
- Subspace maximal dimension, r
- Level of split, n

Output:
- GMM estimate $P_{\hat{\boldsymbol{Y}}^{[s]}} = (\omega_{\hat{\boldsymbol{Y}}^{[s]},i}, \mu_{\hat{\boldsymbol{Y}}^{[s]},i}, \Sigma_{\hat{\boldsymbol{Y}}^{[s]},i})$

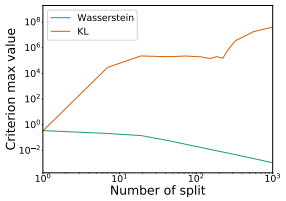
1: **Active subspace selection:** $\Sigma_{\boldsymbol{X}} = U_r \Lambda_r U_r^T$
2: **Burn in:** Apply s_0 split operations $S^{[s_0]}(P_{\boldsymbol{X}}; n) = P_{\hat{\boldsymbol{X}}^{[\boldsymbol{s_0}]}} = (\omega_{\hat{\boldsymbol{X}}^{[\boldsymbol{s_0}]},i}, \mu_{\hat{\boldsymbol{X}}^{[\boldsymbol{s_0}]},i}, \Sigma_{\hat{\boldsymbol{X}}^{[\boldsymbol{s_0}]},i})$ (Eq. (1-2))
3: **Split&Merge:** For each components $P_{\hat{\boldsymbol{X}}^{[\boldsymbol{s_0}]},i}$ of $P_{\hat{\boldsymbol{X}}^{[\boldsymbol{s_0}]}}$:
4: **UT sampling:** Generate $Z_{i,[j]}$ and ω_j (Sec. 3.4.2)
5: **Propagation:** Estimate $P_{\bar{\boldsymbol{Y}}^{[s]},i}$ and $P_{\hat{\boldsymbol{Y}}^{[s]},i}$ (Sec. 3.4.3)
6: **Criteria estimation:** Compute B_s using Equation (5)
7: **Splitting:** If $B_s > T_{split}$ then split $P_{\hat{\boldsymbol{X}}^{[s]},i}$ (Eq. (1-2))
8: **Iteration:** Apply the steps (4)-(7) to all splitted components of $P_{\hat{\boldsymbol{X}}^{[s]},i}$ until $B_s < T_{split}$
9: **Output estimation:** Estimate $(\mu_{\hat{\boldsymbol{Y}}^{[s]},i}, \Sigma_{\hat{\boldsymbol{Y}}^{[s]},i})_{1 \leq i \leq M}$ by moment matching

4. Experiments

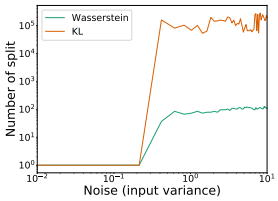
We first present a short single ReLU propagation experiment before detailling our experiment performed over the MNIST dataset, where the input dimension allow a deep comparison with a MC reference. Then, we present experiments conducted over the CIFAR-10, CIFAR-100 and Camelyon datasets providing a wide range of scenarios.

4.1. Single ReLU experiment

To illustrate the theoretical limitations exposed in Section 2 of the use of the KL divergence to propagate Gaussian distribution through neural networks using ReLU activation function, we conduct an experiment consisting in propagating a univariate Gaussian distribution through a ReLU activation function using the KL divergence based criterion (using the Leaky-ReLU approximation with slope $\lambda = 10^{-3}$), which is a heavily-used key step in the approach of Zhang & Shin (2021). Our approach is also tested (without the Leaky-ReLU approximation) for comparison, although it can be applied directly to the whole network.



(a) Evolution of the splitting criterion at each step of the iterating splitting procedure.



(b) Number of splits as a function of the input variance required to propagate $X \sim \mathcal{N}(1.0, \sigma^2)$ through a ReLU.

Figure 1. Comparison between the KL and the Wasserstein based criteria to propagate $X \sim \mathcal{N}(1.0, \sigma^2)$ through a ReLU (Leaky-ReLU approximation with slope $\lambda = 10^{-3}$ for the KL criterion)

First, Figure 1a shows the evolution of the maximum criterion value at each step of the iterating procedure. Our proposed Wasserstein-based criterion decreases as the number of splits increases until it falls below the specified threshold, ending the procedure. On the contrary, the criterion based on the KL divergence becomes very large and thus the procedure does not end. More theoretical information on this phenomenon is provided in Appendix A.1. Finally, Figure 1b shows the evolution of the number of necessary

splits when propagating a Gaussian $\mathcal{N}(1.0, \sigma^2)$ through a ReLU. Observe that it leads to a very large number of splits, making the propagation of a Gaussian distribution through the first layer of a ReLU network already impracticable. Appendix A.2 also provides the numerical details of the first two iterations of this procedure using the KL divergence and the Leaky-ReLU approximation with a slope $\lambda = 10^{-3}$.

4.2. MNIST dataset

In this section, our method is tested against a MNIST regression task by looking at the law of logits where input images are corrupted with different types of noise (Gaussian noise, blur and contrast noise) at different levels of intensity (I_1, I_2, I_3[5] referring to low, medium and high intensity, respectively). These noises were selected in order to cover a wide range of scenarios in terms of covariance amplitude. Indeed, while the Gaussian noise corrupts the pixels independently (weak covariances), the blur corrupts each pixel according to its neighborhood (medium covariances) and the contrast noise affects the whole image (large covariances). The neural network is a 4-layers perceptron with 200 neurons each. Practically, we generate 10^4 noisy samples, from which we extract a mean vector and the associated variance covariance matrix. Then, we propagate the resulting normal distribution through the network by comparing our method (WGMprop) to the LPN method (Gast & Roth, 2018) and full network UT propagation paradigm (Abdelaziz et al., 2015), referenced as UT@FN in the following. We have not been able to integrate in our experiments, the work of (Zhang & Shin, 2021) for the reasons previously presented. Figure 2 presents an input sample corrupted with the different types of noises and levels of intensity. The MC reference is built over 10^6 generated samples for all experiments.

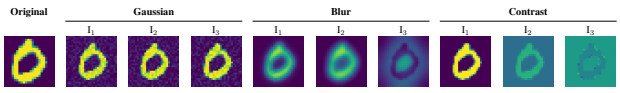


Figure 2. Input sample image corrupted with different types of noise (Gaussian noise, blur and contrast noise) at different levels of intensity (I_1, I_2, I_3).

4.2.1. Results

A quantitative performance evaluation was performed over 500 test images from the MNIST dataset, comparing their performance against the MC reference, which shows a mean prediction time of 15.91 seconds ($\sim$ 6.5s for sample generation and $\sim$ 9s for the propagation). We are studying here the distribution of the class probabilities provided by the fixed network. The comparison between the reference distribution obtained by Monte Carlo Sampling and the one obtained by the various methods is performed using a probabilistic metric such as the 2-Wasserstein distance (2W) but also by comparing the quality of the estimation, using the Mean absolute percentage error, of the mean probability($\mathbf{MAPE}_{\text{MEAN}} = 100(|\mu_{MC} - \mu_{WGMprop}|)/\mu_{MC}$), and of the standard deviation($\mathbf{MAPE}_{\text{STD}}$). We also compute the 95% confidence interval (IOU_{95}) and the mean prediction time (TIME). Table 1 presents the results for the highest noise intensity as this represents the most challenging settings for our method (requiring the highest number of splits).

Table 1. Numerical comparison of the performance of state-of-the-art methods and WGMprop on the three studied noises (Gaussian, Blur and Contrast) at intensity I_3. Standard deviation in ().

			PERFORMANCE CRITERIA				
Noise	**Method**	**# Gaussian**	**2W** ($\times 10^3$)	$\mathbf{MAPE}_{\text{MEAN}}$	$\mathbf{MAPE}_{\text{STD}}$	$\mathbf{IOU}_{95}$	**TIME (s)**
Gaussian	LPN	1	2.39(3.11)	4.26(4.82)	197.08(411.19)	0.342(0.156)	0.02(0.00)
	UT	1	0.03(0.05)	0.21(0.26)	6.07(7.98)	0.859(0.117)	0.06(0.01)
	Zhang et al.	10000*	-/-	-/-	-/-	-/-	-/-
	WGMprop	349	0.02(0.04)	0.16(0.20)	5.69(8.44)	0.876(0.113)	0.48(0.04)
Blur	LPN	1	39.77(19.38)	13.61(9.48)	88.52(3.43)	0.145(0.037)	0.02(0.00)
	UT	1	17.44(12.76)	17.00(10.89)	30.74(17.03)	0.604(0.101)	0.05(0.01)
	Zhang et al.	10000*	-/-	-/-	-/-	-/-	-/-
	WGMprop	635	0.26(0.21)	0.17(0.15)	0.48(0.51)	0.920(0.034)	0.58(0.03)
Contrast	LPN	1	88.15(33.15)	53.66(23.12)	66.55(10.03)	0.343(0.099)	0.02(0.00)
	UT	1	95.85(45.12)	66.88(25.57)	44.18(26.74)	0.446(0.162)	0.04(0.01)
	Zhang et al.	10000*	-/-	-/-	-/-	-/-	-/-
	WGMprop	691	0.25(0.23)	0.18(0.16)	0.26(0.29)	0.938(0.041)	0.81(0.10)

*Number of splits reached the memory limit at the first ReLU.

Here, the 'true' input noise distribution was fitted using a single Gaussian distribution. However, it is possible to fit this input distribution using a GMM with K components. All results for all noises and intensity and for $K = 1, 2, 3$ are presented in Appendix B.

From an overall perspective, LPN and UT@FN methods are faster than WGMprop at the cost of accuracy. Indeed, no matter the input noise complexity, the mean prediction time of these methods stays constant (≈ 0.02s) while their prediction errors rise dramatically (the IOU_{95} drops from 0.859 for a Gaussian noise and a UT@FN propagation to 0.446 for a contrast noise). In fact, in the case of an input corrupted with a Gaussian noise, the output PDF is closely Gaussian (2-Wasserstein value of 0.03×10^{-5} for the UT@FN method) whereas, for more complex noises, the Gaussian assumption on the output PDF does not hold.

To illustrate this statement, Figure 3 displays both the estimated output PDFs and the reference PDF (MC) over sample image n°66 (of the test dataset) corrupted with the whole set of noises at the highest intensity (I_3). In the case of blur and contrast noise, UT@FN and LPN methods suffer from shape misrepresentation due to their Gaussian assumption. We can infer that neglecting the covariances in the

[5] **Gaussian Noise**: $\mu = (0, 0, 0)$, $\sigma^2 = (0.002, 0.01, 0.05)$; **Blur**: $\mu = (0, 0, 0)$, $\sigma^2 = (1, 16, 49)$; **Contrast Noise**: $\mu = (1, 1, 1)$, $\sigma^2 = (0.04, 1, 25)$. (μ, σ) correspond to the parameters set for I_1, I_2 and I_3.

layer propagation proposed by LPN results in low estimation accuracy of the output distribution statistics. On the other hand, the WGMprop method captures the shape of the output PDF well. It is, therefore, a good candidate to obtain a reliable estimate of the 95% prediction interval while maintaining a reasonable execution time (a maximum of 0.81s for the contrast noise).

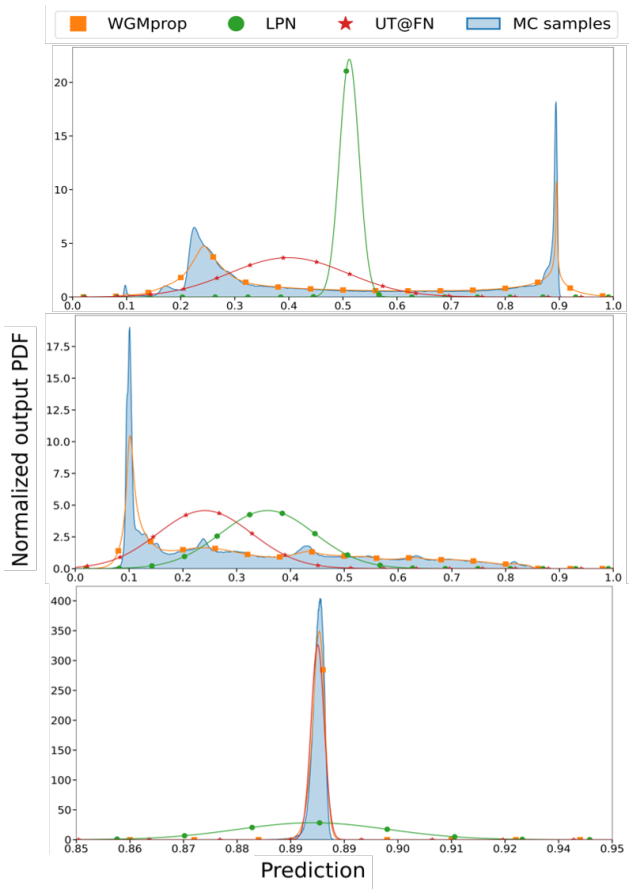


Figure 3. Estimated output PDFs (main marginal) for sample n°66 corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

Although the mean prediction time slightly increases with the input noise complexity (more Gaussian components in the mixture are needed), WGMprop method predictions remain highly accurate: $\mathbf{IOU}_{95}$ superior to 0.876, $\mathbf{MAPE}_{\text{MEAN}}$ lower than 0.18%. In addition, the boxplot of the MAPE of several predicted percentiles (1, 2.5, 25, 25, 97.5 and 99) are shown in Figure 4. These boxplots present the MAPE distributions over the different percentiles of the output PDF for the 3 compared methods for a contrast noise of high intensity. As expected, the methods under Gaussian assumption present high errors on the considered set. Using Gaussian mixture allows a faithful representation of the output PDF and thus allows a reliable estimation of the different percentiles. We note a higher error towards the low percentiles. For higher percentiles, our approach shows small MAPE values as well as small dispersion of these errors.

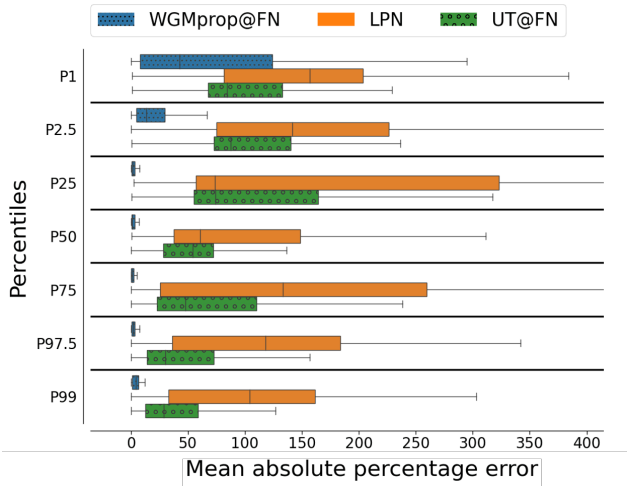


Figure 4. MAPE boxplots for percentiles 1, 2.5, 25, 50, 75, 97.5 and 99. Input images were degraded by a large contrast kernel (I_3).

4.3. More complex dataset

In this section, we test our methodology over more complex datasets and network architectures, detailed in Table 2.

Table 2. Experiments details.

Dataset	Architecture	Input dimension	Output dimension	Last Layer	Nb parameters	s_0
CIFAR-10	$VGG8$	$32 \times 32 \times 3$	10	Linear	$2,397,216$	3
CIFAR-100	$VGG13$	$32 \times 32 \times 3$	100	Softmax	$10,171,936$	1
Camelyon	$VGG10$	$96 \times 96 \times 3$	2	Sigmoïd	$10,615,328$	1

This set of experiments allows us to explore different scenarios by varying the input and output dimensions, the number of parameters in the network, and the kind of last layer used in the network (linear layer for regression case, softmax for multiclass classification, and sigmoid for binary classification). For these experiments, we maintain Monte Carlo as a reference and implement UT@FN in addition to our method (WGMprop). In the following section, we present high-intensity blur noise results. For these high-dimensional problems, we set the maximum dimension of the subspace to 30. The number of burns is indicated by the column s_0.

4.3.1. Results

Table 3 presents the results using the same metrics as for the MNIST classification problem.

Table 3. Numerical comparison of the performance of state-of-the-art methods and WGMprop on Blur noise at the highest level of intensity I_3. Standard deviation are presented in ().

			PERFORMANCE CRITERIA				
Dataset	**Method**	**# Gaussian**	**2W** ($\times 10^3$)	**MAPE**$_{\text{MEAN}}$	**MAPE**$_{\text{STD}}$	**IOU**$_{95}$	**TIME (s)**
CIFAR-10	MC (reference)						38.83
	UT@FN	1	12.15(14.32)	16.96(13.58)	30.60(16.74)	0.634(0.122)	0.39(0.10)
	WGMprop@FN	629	0.64(0.88)	0.63(0.63)	1.73(1.65)	0.877(0.061)	1.77(0.16)
CIFAR-100	MC (reference)						49.92
	UT@FN	1	1.39(2.18)	60.33(43.67)	35.61(20.33)	0.320(0.227)	0.41(0.12)
	WGMprop@FN	709	0.16(0.32)	3.45(3.33)	2.84(2.99)	0.783(0.121)	6.54(0.55)
Camelyon	MC (reference)						359.21
	UT@FN	1	6.96(12.32)	43.93(87.48)	145.78(345.12)	0.131(0.181)	23.04(5.72)
	WGMprop@FN	309	0.25(0.40)	3.80(12.93)	25.14(59.98)	0.783(0.184)	28.68(7.18)

We observe a net increase in execution time for both the MC reference and the two compared methods. Indeed, all three share an incompressible time in selecting the active subspace (about 0.35 seconds for CIFAR10 and CIFAR100 datasets and about 22 seconds for the Camelyon dataset). WGMprop maintains a competitive propagation time compared to UT@FN, which requires the propagation of only $2r+1$ samples, where r denotes the dimension of the active subspace. WGMprop maintains high performance over all metrics for the CIFAR-10 and CIFAR-100 datasets, being 22 times faster than Monte Carlo in the case of the CIFAR-10 and eight times faster for the CIFAR-100. Finally, for the Camelyon dataset, which is a binary classification task with a sigmoid used as the last layer of the neural network, WGMprop shows good performances for the 2W and $\mathbf{MAPE}_{\text{STD}}$ metrics. However, the binary classification configuration can produce output PDFs extremely concentrated around its mean, where the notion of standard deviation and percentile does not seem relevant anymore. This most likely explains the results obtained for $\mathbf{MAPE}_{\text{STD}}$ and IOU_{95}.

5. Limitations

While precise, robust, and generic, our method built upon GM models requires the empirical adjustment of hyperparameters for the splitting stage. A poor choice of threshold conducts in a significant increase in memory usage and computational time, automating its setting for the expected performances in terms of accuracy could be a significant improvement of the proposed method. Indeed, in the presented experiments, hyperparameters have been fixed empirically. Nonetheless, through experimentation, we were able to identify a threshold value of $T_{split} = 10^{-4}$ that worked well for most experiments. We also recommend the use of a 7-level split as it is the one showing the smallest $\epsilon_{0,1}$ (see Table 8 in Appendix E). Moreover, a trade-off between accuracy and performance must be addressed for a covariance matrix with a high dimensional active subspace (setting of the r value). Finally, the number of burn-in stages depends on the application and noise intensity. A value between 0 and 3 seems reasonable.

6. Conclusion

In this work, we proposed a Wasserstein-based Gaussian Mixture propagation method for input uncertainty propagation in Neural Networks. WGMprop provides a reliable estimate of the 95% prediction interval in case of highly corrupted input images. The proposed method is generic by considering the neural network as a black box while using its specificities (strongly linear by part) by computing an adapted Wasserstein-based criterion. Moreover, the method becomes even more relevant compared to standard Monte Carlo methods when the cost of a single call to the neural network increases. Finally, we provide a theoretical convergence guarantee of our Wasserstein-based input uncertainty propagation paradigm. This allows to effectively estimate the statistical parameters of the output distribution. In addition to meeting the previously detailed limitations, the following steps will focus on integrating weight uncertainty to provide a more comprehensive uncertainty budget.

7. Acknowledgements

This work has been founded by the French National Association for Research and Technology, the Laboratoire Nationale de Métrologie et d'Essais (LNE) and the Centre de Mathématiques Appliquées (CMAP) of the Ecole Polytechnique.

References

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

Agrawal, A., Modi, A., Passos, A., Lavoie, A., Agarwal, A., Shankar, A., Ganichev, I., Levenberg, J., Hong, M., Monga, R., et al. Tensorflow eager: A multi-stage, python-embedded dsl for machine learning. *Proceedings of Machine Learning and Systems*, 1:178–189, 2019.

Alspach, D. and Sorenson, H. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE transactions on automatic control*, 17(4):439–448, 1972.

Astudillo, R. F. and Neto, J. P. d. S. Propagation of uncertainty through multilayer perceptrons for robust automatic speech recognition. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

Bae, H., Jang, J., Jung, D., Jang, H., Ha, H., Lee, H., and Yoon, S. Security and privacy issues in deep learning. *arXiv preprint arXiv:1807.11655*, 2018.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra,

D. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Calvetti, D., Reichel, L., and Sorensen, D. C. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2(1):21, 1994.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pp. 1683–1691. PMLR, 2014.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Constantine, P. G., Dow, E., and Wang, Q. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4): A1500–A1524, 2014.

Cox, M. and O'Hagan, A. Meaningful expression of uncertainty in measurement. *Accreditation and Quality Assurance*, 27(1):19–37, 2022.

DeMars, K. J., Bishop, R. H., and Jah, M. K. Entropy-based approach for uncertainty propagation of nonlinear dynamical systems. *Journal of Guidance, Control, and Dynamics*, 36(4):1047–1057, 2013.

Der Kiureghian, A. and Ditlevsen, O. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

Faubel, F. and Klakow, D. Further improvement of the adaptive level of detail transform: Splitting in direction of the nonlinearity. In *2010 18th European Signal Processing Conference*, pp. 850–854. IEEE, 2010.

Faubel, F., McDonough, J., and Klakow, D. The split and merge unscented gaussian mixture filter. *IEEE Signal Processing Letters*, 16(9):786–789, 2009.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Gast, J. and Roth, S. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3369–3378, 2018.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Julier, S. J. and Uhlmann, J. K. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3): 401–422, 2004.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Litjens, G., Bandi, P., Ehteshami Bejnordi, B., Geessink, O., Balkenhol, M., Bult, P., Halilovic, A., Hermsen, M., van de Loo, R., Vogels, R., et al. 1399 h&e-stained sentinel lymph node sections of breast cancer patients: the camelyon dataset. *GigaScience*, 7(6):giy065, 2018.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, pp. 3, 2013.

Malinin, A., Mlodozeniec, B., and Gales, M. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076*, 2019.

MATLAB version 9.3.0.713579 (R2017b). The Mathworks, Inc., Natick, Massachusetts, 2017.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P.,

Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Scott, D. W. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

Sorenson, H. W. and Alspach, D. L. Recursive bayesian estimation using gaussian sums. *Automatica*, 7(4):465–479, 1971.

Straka, O., Duník, J., and Punčochář, I. Directional splitting for structure adaptation of bayesian filters. In *2016 American Control Conference (ACC)*, pp. 2705–2710. IEEE, 2016.

Tagasovska, N. and Lopez-Paz, D. Single-model uncertainties for deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Terefjanu, G. An adaptive split-merge scheme for uncertainty propagation using gaussian mixture models. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pp. 890, 2011.

Titensky, J. S., Jananthan, H., and Kepner, J. Uncertainty propagation in deep neural networks using extended kalman filtering. In *2018 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pp. 1–4. IEEE, 2018.

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., and Yu, T. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

Van Rossum, G. and Drake, F. L. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Villani, C. *Optimal transport: old and new*, volume 338. Springer, 2009.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Zhang, B. and Shin, Y. C. An adaptive gaussian mixture method for nonlinear uncertainty propagation in neural networks. *Neurocomputing*, 458:170–183, 2021.

A. Analysis of KL divergence as statistical distance in the Split&Merge paradigm for ReLU neural networks.

A.1. Analyse of the KL-based splitting criterion

Definition A.1. (KL divergence) Let P and Q be distributions on the same set U. Then the KL divergence of p from q, denoted $D(P\|Q)$, is defined as

$$D_{KL}(P\|Q) = \int_U \frac{dP}{d\lambda}(x) \log \frac{\frac{dP}{d\lambda}(x)}{\frac{dQ}{d\lambda}(x)} d\lambda(x)$$

where $\frac{dP}{d\lambda}$ and $\frac{dQ}{d\lambda}$ are the densities of P and Q with respect to a common dominating measure (which always exists). In the discrete setting, this amounts to

$$D_{KL}(P\|Q) = \sum_{x\in U} P(x) \log \frac{P(x)}{Q(x)}.$$

When both P and Q have a density with respect to the Lebesgue measure, we obtain

$$D_{KL}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx.$$

The KL divergence is a particular case of the family of f-divergence. It is not a proper distance since it is not symmetric and does not verify the triangular inequality (which is primordial in our case to compute distance with mixture distribution). In a more general aspect, the KL divergence becomes infinite when Q does not dominate P, i.e. it exists an event A, such that $P(A) \neq 0$ and $Q(A) = 0$. This happens, for instance, if the support of P is not included in the one Q when or if there is Q has a density with respect to the Lebesgue measure while P is a mixture of such a density and one putting mass on singletons. This situation arises for neural networks using ReLU activation functions. Zhang & Shin (2021) proposed to overcome this problem by approximating the ReLU activation function by a Leaky ReLU with slope $\lambda = 10^{-K}$ and to control the criteria $D_{KL}(p\|\hat{p})$ by computing the following upper bound (equation (9) in their paper):

$$D_{KL}(p\|\hat{p}) \leq \sum_{i=1}^{M} w_{s,i} D_{KL}\left(p_i\|\hat{p}_i\right) \tag{8}$$

with $D_{KL}\left(p_i\|\hat{p}_i\right)$ defined as:

$$\begin{aligned} D_{KL}(p\|\hat{p}) = &-\frac{k}{2} + \sum_{i=1}^{k} \log\left(f'\left(\mu_i\right)\right) - \sum_{i=1}^{k} E_{p(z_i)}\left[\log\left(f'\left(z_i\right)\right)\right] \\ &+ \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\left(\frac{c_{ij}}{f'\left(\mu_i\right) f'\left(\mu_j\right)} E_{p(z_i,z_j)}\left[\left(f\left(z_i\right) - f\left(\mu_i\right)\right)\left(f\left(z_j\right) - f\left(\mu_j\right)\right)\right]\right). \end{aligned} \tag{9}$$

Theoretically, this upper bound is not defined for the ReLU activation function. Indeed, when $\mu_i < 0$ and $\mu_j < 0$, $\frac{1}{f'(\mu_i)f'(\mu_j)} = \infty \approx 10^{2K}$. To circumvent this issue, the authors reduce the upper bound by reducing each term of the sum $\sum_{i=1}^{M} w_{s,i} D_{KL}\left(p_i\|\hat{p}_i\right)$. As a direct consequence, $w_{s,i} D_{KL}\left(p_i\|\hat{p}_i\right)$ becomes small when $w_{s,i}$ is very small. So, the proposed strategy artificially increases the number of Gaussian components in the mixture (increasing memory usage and computation time drastically).

This large number of components makes the KL-based criterion unusable in practice. Indeed, as soon as the first activation function (ReLU) is output, a costly merge step must be applied (detailed below). As each of the remaining components has a different variance-covariance matrix, it is also necessary to perform as many inverses (the complexity scales with the cube of the dimension of the current layer), which is impossible for classical CNN architectures.

A.2. Numerical details of the experiments 4

In the single ReLU experiment presented in Section 4, the first value of the criterion for a univariate Gaussian distribution $\mathcal{N}(1.0, 1.0)$ is 0.333. It is then higher than the threshold value 0.001 and a step of splitting is applied. In the case of a 7-split

using the hyperparameters provided in Appendix E, one of the splitted Gaussian components is $\mathcal{N}(-0.4992, 0.4389)$. For this component, the new criterion value explodes to a value of 29450. The following provides the details of this calculation: Consider the univariate Gaussian distribution $\mathcal{N}(1.0, 1.0)$. Let z_i its UT samples, with $z_0 = 1.0$, $z_1 = 1.0 + \sqrt{3}\sigma = 1.0 + \sqrt{3} * 1.0 = 2.7320$ and $z_2 = 1.0 - \sqrt{3}\sigma = -0.7320$. Thus:

$$D_{KL} \leq -\frac{1}{2} + log(1.0) - (\frac{2}{3}log(1.0) + \frac{1}{6}log(1.0) + \frac{1}{6}log(0.001) + \frac{1}{2}(\frac{1.0}{1.0 * 1.0}(\frac{2}{3}(1.0-1.0)^2 + \frac{1}{6}(2.7320-1.0)^2 + \frac{1}{6}(0-1.0)^2$$

$$D_{KL} \leq 0.3333$$

Then, one of the 7-splitted components is $\mathcal{N}(-0.4992, 0.1926)$. Following the same formula, we have $z_0 = -0.4992$, $z_1 = -0.4992 + \sqrt{3}\sigma = -0.4992 + \sqrt{3} * 0.4389 = 0.2609$ and $z_2 = -0.4992 - \sqrt{3}\sigma = -1.2594$. Finally:

$$D_{KL} \leq -\frac{1}{2} + log(0.001) - (\frac{2}{3}log(0.001) + \frac{1}{6}log(1.0) + \frac{1}{6}log(0.001) + \frac{1}{2}(\frac{0.1926^{-1}}{0.001^2}(\frac{2}{3}(0-0)^2 + \frac{1}{6}(0.2609-0)^2 + \frac{1}{6}(0-0)^2$$

$$D_{KL} \leq 29450$$

This KL based criterion explosion is due to the fact that for this Gaussian we have its mean $\mu \in [-\sqrt{3}\sigma, 0]$. In practice, this situation always arises due to the UT sampling deterministic equations.

A.3. Analyse of the KL-based merging criterion

As mentioned in Sect. 2, the use of a KL based criterion requires a layer wise propagation through the neural network. It would be possible to manually stop the splitting procedure by introducing a maximum number of components per layer. However, a merging stage is required before propagating through the next layer. To do so, Zhang & Shin (2021) propose to rely on the computation of an upper bound of the KL divergence between the Gaussian mixture composed of the components (i, j) and the resulting merge Gaussian distribution. (Equation 21 in their paper).

$$B_{i,j} = \frac{1}{2}\left[(w_i + w_j)\log|\mathbf{\Sigma}_{ij}| - w_i \log|\mathbf{\Sigma}_i| - w_j \log|\mathbf{\Sigma}_j|\right] \tag{10}$$

This upper bound needs the computation of the log of the determinant of the covariance matrix of the different Gaussian distributions. Unfortunately, the determinants in such Gaussian distributions are generally close to 0 (or even equal to 0) which conducts the absolute value of the logarithm to be very large (or even to diverge). Note that

$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)),$$

for A a $K \times L$ matrix and B an $L \times M$ matrix. Practically, it is not rare that the covariance matrix of a Gaussian component of the mixture is singular. Empirically, we also observe that a few eigenvalues of the covariance matrices are $\gg 0$.

In a layer-wise paradigm, the computational cost also depends on the width of each layer of the neural network. Using our Wasserstein-based approach, it allows propagating the input distribution directly through the whole network avoiding these problems.

B. All results

In this section, we display all the results from all experimental settings and additional figures in order to illustrate the performances of our methodology. Best performing methods for a given performance criterion are highlighted using green rectangles.

B.1. Experiment settings

In the main body, the experiments were conducted by fitting the noisy input distribution by a single Gaussian distribution. However, this input distribution can be fitted by a Gaussian mixture distribution with K components. Moreover, our proposed method considers the neural network as a black box and thus the propagation is performed in a full network fashion using UT sampling for output moment estimation. We wish to perform a comparison with a layer-wise implementation of our approach (rendered possible by the sound and suitable Wasserstein criterion presented in Sect. 3). We additionally study the impact of analytical moment estimations using the theorems provided in Appendix C. Thus, we provide in this section the results on the MNIST experiment for different values of K (here $K \in \{1, 2, 3\}$ (Pedregosa et al., 2011)) and for the different variants of WGMprop displayed in Table 4:

- LPN: layer-wise propagation of a single Gaussian diagonal distribution using analytical formula
- HPN: LPN: layer-wise propagation of a single Gaussian distribution using analytical formula (LPN + covariance estimations)
- UT@LW: layer-wise propagation of a single Gaussian distribution using UT sampling
- UT@FN: full network propagation of a single Gaussian distribution using UT sampling
- WGMProp@LPN: layer wise propagation of a GMM using analytical formula (no estimation of covariances)
- WGMProp@HPN: layer wise propagation of a GMM using analytical formula (estimation of covariances)
- WGMProp@UT: layer wise propagation of a GMM using UT sampling
- WGMProp@FN: our main methodology detailed in the main body of the paper.

Table 4. Propagation characteristics for both state-of-the-art methods and the various methods of the WGMprop framework.

	Propagation		**Output PDF**		**Moment estimation**		**Covariance**		**Criterion**	
	Layer-wise	**Full Network**	**Single Gaussian**	**Gaussian Mixture**	**UT Sampling**	**Analytic**	**With**	**Without**	**Wasserstein**	**KL**
LPN	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗
HPN	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗
UT@LW	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗
UT@FN	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗
WGMProp@LPN	✓	✗	✗	✓	✗	✓	✗	✓	✓	✗
WGMProp@HPN	✓	✗	✗	✓	✗	✓	✓	✗	✓	✗
WGMProp@UT	✓	✗	✗	✓	✓	✗	✓	✗	✓	✗
WGMProp@FN	✗	✓	✗	✓	✓	✗	✓	✗	✓	✗

Hyperparameters: In the following of this section experimental results are displayed using the following parameters: $s_0 = 0$, $T_{\text{split}} = 0.0001$, no subspace selection ($r =$ input dimension). In this experimental setting, the MC reference shows a mean prediction time of 74.53 seconds ($\sim$ 65s for sample generation and $\sim$ 9s for sample propagation) In the main body of this paper, the execution time was drastically reduced by setting $r = 30$.
Noise parameters are the following:
Gaussian Noise: $\mu = (0, 0, 0)$, $\sigma^2 = (0.002, 0.01, 0.05)$; **Blur**: $\mu = (0, 0, 0)$, $\sigma^2 = (1, 16, 49)$; **Contrast Noise**: $\mu = (1, 1, 1)$, $\sigma^2 = (0.04, 1, 25)$. (μ, σ) correspond to the parameters set for I_1, I_2 and I_3.

Hardware and software: The experiments were conducted using a GPU NVIDIA Telsa V100 32GO HBM2 and a CPU Intel Xeon Silver 4210. All the codes were written using Python 3 (Van Rossum & Drake, 2009) and Tensorflow 2 (Agrawal et al., 2019). The implementation of our method was not optimized in terms of computation time.

B.2. MNIST dataset

B.2.1. METRIC TABLES

Table 5. Performance comparison for all type of noise and intensity, and with $K = 1$. Standard deviations are presented in $(\pm)$.

				PERFORMANCE CRITERIA					
Noise	**Intensity**	**Method**	**# Gaussian**	**2W**	**KL**	**MEAN**	**STD**	**IOU_{95}**	**TIME (s)**
Gaussian Nois	I_1	LPN	1	0.090(±0.130)	0.41(±0.34)	0.16(±0.13)	6255.53(±1570.23)	0.023(±0.010)	0.02(±0.01)
		HPN	1	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.60)	0.948(±0.039)	0.12(±0.01)
		UT@LW	1	0.009(±0.008)	0.02(±0.02)	0.08(±0.09)	4.04(±2.96)	0.931(±0.039)	0.07(±0.01)
		UT@FN	1	0.019(±0.029)	0.04(±0.03)	0.06(±0.07)	7.84(±7.75)	0.904(±0.077)	0.09(±0.01)
		WGMprop@LPN	10.14(±6.18)	0.389(±0.351)	0.29(±0.26)	0.15(±0.12)	1227.99(±1576.32)	0.167(±0.117)	0.79(±0.39)
		WGMprop@HPN	10.14(±6.18)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.61(±0.56)	0.952(±0.036)	1.48(±0.81)
		WGMprop@UT	1.91(±2.15)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.98(±2.90)	0.932(±0.038)	0.30(±0.20)
		WGMprop@FN	6.02(±2.22)	0.019(±0.029)	0.04(±0.03)	0.06(±0.07)	7.75(±7.70)	0.905(±0.076)	0.67(±0.23)
	I_2	LPN	1	0.090(±0.130)	0.41(±0.34)	0.16(±0.13)	6255.53(±1570.23)	0.023(±0.010)	0.02(±0.01)
		HPN	1	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.60)	0.948(±0.039)	0.12(±0.01)
		UT@LW	1	0.009(±0.008)	0.02(±0.02)	0.08(±0.09)	4.04(±2.96)	0.931(±0.039)	0.07(±0.01)
		UT@FN	1	0.019(±0.029)	0.04(±0.03)	0.06(±0.07)	7.84(±7.75)	0.904(±0.077)	0.09(±0.01)
		WGMprop@LPN	10.14(±6.18)	0.389(±0.351)	0.29(±0.26)	0.15(±0.12)	1227.99(±1576.32)	0.167(±0.117)	0.79(±0.39)
		WGMprop@HPN	10.14(±6.18)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.61(±0.56)	0.952(±0.036)	1.48(±0.81)
		WGMprop@UT	1.91(±2.15)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.98(±2.90)	0.932(±0.038)	0.30(±0.20)
		WGMprop@FN	6.02(±2.22)	0.019(±0.029)	0.04(±0.03)	0.06(±0.07)	7.75(±7.70)	0.905(±0.076)	0.67(±0.23)
	I_3	LPN	1	0.090(±0.130)	0.41(±0.34)	0.16(±0.13)	6255.53(±1570.23)	0.023(±0.010)	0.02(±0.01)
		HPN	1	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.60)	0.948(±0.039)	0.12(±0.01)
		UT@LW	1	0.009(±0.008)	0.02(±0.02)	0.08(±0.09)	4.04(±2.96)	0.931(±0.039)	0.07(±0.01)
		UT@FN	1	0.019(±0.029)	0.04(±0.03)	0.06(±0.07)	7.84(±7.75)	0.904(±0.077)	0.09(±0.01)
		WGMprop@LPN	10.14(±6.18)	0.389(±0.351)	0.29(±0.26)	0.15(±0.12)	1227.99(±1576.32)	0.167(±0.117)	0.79(±0.39)
		WGMprop@HPN	10.14(±6.18)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.61(±0.56)	0.952(±0.036)	1.48(±0.81)
		WGMprop@UT	1.91(±2.15)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.98(±2.90)	0.932(±0.038)	0.30(±0.20)
		WGMprop@FN	6.02(±2.22)	0.019(±0.029)	0.04(±0.03)	0.06(±0.07)	7.75(±7.70)	0.905(±0.076)	0.67(±0.23)
Blur	I_1	LPN	1	0.019(±0.132)	13.53(±12.40)	2.20(±1.62)	12922.73(±6028.60)	0.011(±0.009)	0.02(±0.01)
		HPN	1	0.129(±0.109)	1.89(±1.60)	0.08(±0.09)	5.04(±7.73)	0.746(±0.097)	0.12(±0.01)
		UT@LW	1	0.151(±0.114)	1.97(±1.58)	0.88(±0.88)	10.26(±6.68)	0.729(±0.092)	0.07(±0.01)
		UT@FN	1	0.193(±0.156)	2.37(±1.73)	1.18(±1.13)	18.00(±12.24)	0.711(±0.092)	0.08(±0.01)
		WGMprop@LPN	12.52(±9.95)	0.150(±0.325)	0.74(±0.65)	0.32(±0.21)	13.21(±11.50)	0.794(±0.092)	0.92(±0.60)
		WGMprop@HPN	10.04(±4.13)	0.007(±0.014)	0.11(±0.10)	0.01(±0.01)	0.19(±0.26)	0.983(±0.016)	1.40(±0.51)
		WGMprop@UT	7.46(±1.79)	0.011(±0.024)	0.13(±0.11)	0.04(±0.04)	0.68(±0.52)	0.970(±0.021)	0.76(±0.15)
		WGMprop@FN	77.73(±46.18)	0.042(±0.135)	0.18(±0.17)	0.07(±0.08)	3.21(±8.65)	0.952(±0.102)	7.04(±4.12)
	I_2	LPN	1	−0.011(±0.065)	45.62(±22.88)	5.25(±4.98)	11079.69(±3116.66)	0.013(±0.004)	0.02(±0.00)
		HPN	1	0.319(±0.146)	7.35(±3.62)	3.55(±2.42)	8.38(±6.99)	0.766(±0.076)	0.12(±0.01)
		UT@LW	1	0.435(±0.180)	8.17(±4.24)	4.58(±3.35)	32.74(±13.20)	0.810(±0.108)	0.06(±0.01)
		UT@FN	1	0.428(±0.162)	8.14(±3.21)	4.63(±4.65)	29.84(±11.98)	0.770(±0.140)	0.08(±0.00)
		WGMprop@LPN	155.03(±104.93)	0.410(±0.229)	1.76(±14.55)	1.20(±9.27)	2.58(±4.03)	0.945(±0.058)	9.38(±5.88)
		WGMprop@HPN	64.16(±17.25)	0.017(±0.016)	0.34(±0.24)	0.05(±0.04)	0.15(±0.25)	0.980(±0.017)	8.20(±2.16)
		WGMprop@UT	43.25(±11.87)	0.034(±0.033)	0.39(±0.24)	0.20(±0.17)	0.52(±0.70)	0.961(±0.026)	3.73(±0.98)
		WGMprop@FN	219.15(±65.99)	0.045(±0.036)	0.43(±0.25)	0.13(±0.13)	1.01(±1.80)	0.953(±0.043)	19.33(±5.77)
	I_3	LPN	1	−0.003(±0.055)	32.79(±17.67)	4.00(±2.71)	9148.25(±2335.73)	0.014(±0.003)	0.02(±0.00)
		HPN	1	0.311(±0.116)	5.84(±3.01)	4.31(±2.47)	11.44(±8.78)	0.760(±0.080)	0.12(±0.01)
		UT@LW	1	0.365(±0.158)	6.04(±3.31)	2.46(±2.38)	24.07(±15.87)	0.763(±0.116)	0.06(±0.00)
		UT@FN	1	0.393(±0.139)	8.12(±3.25)	8.21(±5.39)	17.86(±10.25)	0.668(±0.133)	0.08(±0.00)
		WGMprop@LPN	235.25(±190.77)	0.498(±0.428)	10.11(±41.46)	4.08(±12.83)	7.54(±8.57)	0.878(±0.106)	14.04(±9.96)
		WGMprop@HPN	88.19(±23.17)	0.030(±0.023)	0.47(±0.31)	0.10(±0.07)	0.43(±0.43)	0.952(±0.030)	11.81(±3.05)
		WGMprop@UT	62.61(±15.17)	0.053(±0.040)	0.62(±0.33)	0.25(±0.19)	0.95(±0.93)	0.927(±0.042)	5.46(±1.27)
		WGMprop@FN	218.48(±55.67)	0.063(±0.043)	0.77(±0.50)	0.25(±0.24)	1.15(±1.21)	0.909(±0.051)	20.08(±5.08)
Contrast	I_1	LPN	1	0.018(±0.170)	18.47(±17.70)	2.90(±2.18)	14683.21(±6712.73)	0.011(±0.011)	0.02(±0.00)
		HPN	1	0.187(±0.185)	2.63(±2.22)	0.11(±0.16)	7.11(±13.77)	0.724(±0.119)	0.12(±0.01)
		UT@LW	1	0.218(±0.191)	2.79(±2.28)	1.28(±1.27)	12.81(±9.04)	0.704(±0.114)	0.06(±0.00)
		UT@FN	1	0.189(±0.190)	2.88(±2.56)	0.22(±0.26)	4.96(±10.58)	0.729(±0.111)	0.08(±0.00)
		WGMprop@LPN	14.06(±9.89)	0.186(±0.348)	0.85(±0.81)	0.33(±0.27)	10.76(±5.89)	0.803(±0.082)	1.02(±0.61)
		WGMprop@HPN	11.97(±5.01)	0.018(±0.031)	0.17(±0.17)	0.02(±0.02)	0.29(±0.68)	0.977(±0.025)	1.63(±0.61)
		WGMprop@UT	8.13(±2.58)	0.030(±0.052)	0.20(±0.19)	0.06(±0.06)	0.85(±0.71)	0.958(±0.035)	0.79(±0.20)
		WGMprop@FN	103.61(±71.68)	0.039(±0.121)	0.13(±0.14)	0.02(±0.02)	2.08(±9.51)	0.952(±0.120)	9.49(±6.50)
	I_2	LPN	1	−0.015(±0.060)	126.98(±65.74)	17.18(±16.74)	10196.32(±3056.44)	0.016(±0.005)	0.02(±0.00)
		HPN	1	0.928(±0.362)	33.22(±16.45)	12.40(±9.17)	11.31(±14.12)	0.670(±0.088)	0.12(±0.01)
		UT@LW	1	1.070(±0.355)	32.05(±14.82)	10.43(±8.52)	36.93(±16.77)	0.748(±0.131)	0.07(±0.01)
		UT@FN	1	0.920(±0.342)	31.64(±18.92)	6.11(±4.95)	11.71(±17.75)	0.680(±0.081)	0.08(±0.01)
		WGMprop@LPN	281.92(±39.79)	0.267(±0.125)	0.63(±1.17)	0.09(±0.08)	0.24(±0.22)	0.992(±0.008)	16.82(±2.34)
		WGMprop@HPN	84.25(±13.48)	0.019(±0.012)	1.45(±2.21)	0.07(±0.06)	0.06(±0.07)	0.993(±0.007)	10.58(±1.66)
		WGMprop@UT	68.63(±12.98)	0.040(±0.029)	1.47(±2.34)	0.17(±0.15)	0.23(±0.19)	0.985(±0.015)	5.65(±1.03)
		WGMprop@FN	383.63(±99.77)	0.018(±0.020)	1.45(±2.14)	0.07(±0.07)	0.09(±0.14)	0.991(±0.016)	34.34(±8.96)
	I_3	LPN	1	0.007(±0.159)	101.48(±56.83)	31.69(±34.55)	6007.26(±1659.82)	0.026(±0.006)	0.02(±0.00)
		HPN	1	0.991(±0.378)	38.38(±17.90)	27.57(±13.04)	11.10(±9.06)	0.640(±0.089)	0.12(±0.01)
		UT@LW	1	0.966(±0.410)	31.27(±16.59)	14.00(±9.41)	15.75(±14.28)	0.633(±0.096)	0.06(±0.00)
		UT@FN	1	0.924(±0.376)	33.98(±19.47)	6.97(±4.87)	9.65(±8.45)	0.604(±0.058)	0.07(±0.00)
		WGMprop@LPN	670.06(±148.56)	0.551(±0.316)	32.22(±42.94)	22.27(±30.94)	13.27(±20.02)	0.790(±0.131)	35.92(±6.72)
		WGMprop@HPN	108.17(±9.88)	0.022(±0.012)	1.46(±3.43)	0.11(±0.07)	0.12(±0.10)	0.989(±0.006)	14.08(±1.30)
		WGMprop@UT	97.02(±9.56)	0.044(±0.026)	1.23(±3.52)	0.24(±0.16)	0.31(±0.24)	0.980(±0.012)	8.02(±0.77)
		WGMprop@FN	492.59(±91.99)	0.022(±0.015)	1.36(±3.04)	0.09(±0.07)	0.08(±0.10)	0.985(±0.015)	44.72(±8.36)

Table 6. Performance comparison for all type of noise and intensity, and with $K = 2$. Standard deviations are presented in $(\pm)$.

			PERFORMANCE CRITERIA						
Noise	**Intensity**	**Method**	**# Gaussian**	**2W**	**KL**	**MEAN**	**STD**	**IOU$_{95}$**	**TIME (s)**
ussian Noise	I_1	LPN	2	0.087(±0.141)	0.39(±0.32)	0.16(±0.14)	94.98(±3.25)	0.041(±0.019)	0.02(±0.00)
		HPN	2	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.62)	0.948(±0.039)	0.22(±0.01)
		UT@LW	2	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.91(±2.80)	0.931(±0.039)	0.10(±0.00)
		UT@FN	2	0.020(±0.045)	0.04(±0.03)	0.06(±0.10)	9.13(±10.68)	0.905(±0.077)	0.14(±0.00)
		WGMprop@LPN	11.04(±5.12)	0.370(±0.340)	0.30(±0.25)	0.16(±0.13)	84.16(±11.00)	0.155(±0.111)	0.84(±0.33)
		WGMprop@HPN	11.04(±5.12)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.62(±0.57)	0.951(±0.037)	1.63(±0.70)
		WGMprop@UT	2.01(±0.19)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.91(±2.80)	0.931(±0.039)	0.31(±0.02)
		WGMprop@FN	5.99(±2.83)	0.018(±0.028)	0.04(±0.03)	0.06(±0.07)	9.05(±10.58)	0.906(±0.075)	0.71(±0.30)
	I_2	LPN	2	0.087(±0.141)	0.39(±0.32)	0.16(±0.14)	94.98(±3.25)	0.041(±0.019)	0.02(±0.00)
		HPN	2	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.62)	0.948(±0.039)	0.22(±0.01)
		UT@LW	2	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.91(±2.80)	0.931(±0.039)	0.10(±0.00)
		UT@FN	2	0.020(±0.045)	0.04(±0.03)	0.06(±0.10)	9.13(±10.68)	0.905(±0.077)	0.14(±0.00)
		WGMprop@LPN	11.04(±5.12)	0.370(±0.340)	0.30(±0.25)	0.16(±0.13)	84.16(±11.00)	0.155(±0.111)	0.84(±0.33)
		WGMprop@HPN	11.04(±5.12)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.62(±0.57)	0.951(±0.037)	1.63(±0.70)
		WGMprop@UT	2.01(±0.19)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.91(±2.80)	0.931(±0.039)	0.31(±0.02)
		WGMprop@FN	5.99(±2.83)	0.018(±0.028)	0.04(±0.03)	0.06(±0.07)	9.05(±10.58)	0.906(±0.075)	0.71(±0.30)
	I_3	LPN	2	0.087(±0.141)	0.39(±0.32)	0.16(±0.14)	94.98(±3.25)	0.041(±0.019)	0.02(±0.00)
		HPN	2	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.62)	0.948(±0.039)	0.22(±0.01)
		UT@LW	2	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.91(±2.80)	0.931(±0.039)	0.10(±0.00)
		UT@FN	2	0.020(±0.045)	0.04(±0.03)	0.06(±0.10)	9.13(±10.68)	0.905(±0.077)	0.14(±0.00)
		WGMprop@LPN	11.04(±5.12)	0.370(±0.340)	0.30(±0.25)	0.16(±0.13)	84.16(±11.00)	0.155(±0.111)	0.84(±0.33)
		WGMprop@HPN	11.04(±5.12)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.62(±0.57)	0.951(±0.037)	1.63(±0.70)
		WGMprop@UT	2.01(±0.19)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.91(±2.80)	0.931(±0.039)	0.31(±0.02)
		WGMprop@FN	5.99(±2.83)	0.018(±0.028)	0.04(±0.03)	0.06(±0.07)	9.05(±10.58)	0.906(±0.075)	0.71(±0.30)
Blur	I_1	LPN	2	0.103(±0.398)	5.43(±4.04)	0.61(±0.39)	64.94(±204.91)	0.418(±0.096)	0.02(±0.00)
		HPN	2	0.062(±0.124)	0.96(±0.75)	0.03(±0.05)	2.19(±10.58)	0.894(±0.074)	0.21(±0.01)
		UT@LW	2	0.061(±0.061)	0.81(±0.59)	0.23(±0.19)	5.39(±3.20)	0.858(±0.071)	0.09(±0.01)
		UT@FN	2	0.131(±0.108)	3.15(±4.03)	3.18(±4.15)	6.74(±6.72)	0.844(±0.092)	0.11(±0.00)
		WGMprop@LPN	9.07(±2.41)	0.278(±0.433)	0.61(±0.57)	0.18(±0.12)	6.80(±6.36)	0.896(±0.071)	0.70(±0.14)
		WGMprop@HPN	8.99(±2.63)	0.030(±0.120)	0.09(±0.10)	0.01(±0.02)	1.29(±9.10)	0.979(±0.023)	1.29(±0.33)
		WGMprop@UT	8.01(±2.11)	0.025(±0.046)	0.11(±0.13)	0.04(±0.06)	0.69(±1.21)	0.969(±0.031)	0.79(±0.18)
		WGMprop@FN	55.78(±30.05)	0.045(±0.093)	0.09(±0.08)	0.04(±0.04)	1.08(±4.35)	0.962(±0.069)	4.92(±2.61)
	I_2	LPN	2	0.028(±0.307)	8.01(±3.60)	2.00(±1.61)	19.79(±169.47)	0.681(±0.139)	0.02(±0.00)
		HPN	2	0.227(±0.130)	1.45(±0.82)	0.21(±0.17)	1.64(±1.91)	0.880(±0.047)	0.21(±0.01)
		UT@LW	2	0.282(±0.145)	1.86(±0.91)	1.35(±1.15)	2.40(±2.78)	0.859(±0.061)	0.09(±0.01)
		UT@FN	2	0.336(±0.178)	7.62(±7.92)	7.41(±7.85)	3.57(±3.71)	0.823(±0.071)	0.12(±0.00)
		WGMprop@LPN	32.76(±16.69)	0.236(±0.273)	0.43(±0.24)	0.22(±0.17)	1.20(±1.93)	0.955(±0.033)	2.14(±0.99)
		WGMprop@HPN	22.98(±6.48)	0.023(±0.018)	0.18(±0.16)	0.02(±0.02)	0.09(±0.08)	0.984(±0.010)	3.06(±0.83)
		WGMprop@UT	15.88(±3.39)	0.043(±0.036)	0.18(±0.12)	0.09(±0.07)	0.23(±0.21)	0.972(±0.019)	1.44(±0.29)
		WGMprop@FN	137.44(±42.71)	0.042(±0.054)	0.17(±0.15)	0.05(±0.04)	0.27(±0.71)	0.970(±0.034)	12.22(±3.76)
	I_3	LPN	2	0.010(±0.213)	6.92(±2.51)	2.50(±2.00)	65.02(±475.44)	0.599(±0.217)	0.03(±0.01)
		HPN	2	0.316(±0.157)	2.36(±1.20)	0.46(±0.37)	2.22(±1.78)	0.823(±0.050)	0.21(±0.02)
		UT@LW	2	0.363(±0.178)	2.53(±1.11)	1.94(±1.30)	4.97(±4.21)	0.788(±0.064)	0.10(±0.01)
		UT@FN	2	0.476(±0.198)	11.05(±12.43)	13.31(±15.69)	7.57(±6.63)	0.768(±0.070)	0.12(±0.00)
		WGMprop@LPN	52.53(±13.76)	0.499(±0.325)	0.24(±0.11)	0.11(±0.10)	1.82(±1.86)	0.954(±0.039)	3.36(±0.83)
		WGMprop@HPN	28.74(±7.94)	0.027(±0.016)	0.31(±0.24)	0.02(±0.02)	0.11(±0.09)	0.982(±0.008)	3.80(±0.99)
		WGMprop@UT	19.33(±5.36)	0.061(±0.042)	0.29(±0.16)	0.13(±0.10)	0.46(±0.47)	0.960(±0.020)	1.75(±0.45)
		WGMprop@FN	174.10(±46.61)	0.031(±0.028)	0.19(±0.15)	0.03(±0.03)	0.20(±0.25)	0.973(±0.023)	15.59(±4.13)
Contrast	I_1	LPN	2	0.265(±0.496)	11.52(±9.90)	1.26(±0.94)	102.79(±226.88)	0.327(±0.087)	0.02(±0.00)
		HPN	2	0.125(±0.158)	2.41(±1.90)	0.15(±0.18)	7.20(±34.32)	0.794(±0.100)	0.22(±0.01)
		UT@LW	2	0.106(±0.085)	2.14(±1.59)	0.68(±0.57)	11.14(±6.59)	0.754(±0.091)	0.10(±0.00)
		UT@FN	2	0.084(±0.084)	2.71(±2.25)	0.17(±0.19)	2.73(±5.81)	0.797(±0.095)	0.12(±0.00)
		WGMprop@LPN	10.53(±5.81)	0.396(±0.527)	0.58(±0.45)	0.22(±0.15)	7.40(±4.85)	0.868(±0.052)	0.81(±0.37)
		WGMprop@HPN	9.49(±3.01)	0.060(±0.146)	0.24(±0.23)	0.03(±0.02)	4.34(±38.48)	0.972(±0.025)	1.40(±0.38)
		WGMprop@UT	8.07(±0.71)	0.020(±0.034)	0.25(±0.23)	0.06(±0.06)	0.81(±0.61)	0.963(±0.030)	0.82(±0.06)
		WGMprop@FN	89.94(±57.08)	0.016(±0.046)	0.15(±0.16)	0.02(±0.01)	0.73(±4.59)	0.974(±0.072)	8.25(±5.18)
	I_2	LPN	2	0.149(±0.653)	31.74(±18.54)	7.11(±5.42)	83.05(±359.62)	0.521(±0.145)	0.03(±0.01)
		HPN	2	0.845(±0.296)	23.87(±11.23)	6.36(±4.41)	11.51(±8.08)	0.732(±0.086)	0.22(±0.02)
		UT@LW	2	0.795(±0.265)	17.77(±7.63)	4.62(±4.89)	12.09(±6.35)	0.754(±0.094)	0.10(±0.01)
		UT@FN	2	0.864(±0.306)	30.22(±15.86)	4.87(±3.99)	4.87(±4.70)	0.654(±0.078)	0.12(±0.01)
		WGMprop@LPN	370.66(±96.08)	0.265(±0.239)	7.06(±36.28)	4.81(±27.88)	3.86(±28.30)	0.981(±0.073)	21.98(±5.41)
		WGMprop@HPN	86.94(±14.74)	0.026(±0.029)	1.46(±1.79)	0.09(±0.04)	0.11(±0.12)	0.991(±0.010)	11.11(±1.87)
		WGMprop@UT	72.18(±11.22)	0.044(±0.028)	1.51(±2.00)	0.18(±0.11)	0.38(±0.31)	0.983(±0.016)	5.97(±0.91)
		WGMprop@FN	433.62(±119.44)	0.021(±0.025)	1.12(±1.53)	0.06(±0.03)	0.10(±0.49)	0.991(±0.019)	38.18(±10.52)
	I_3	LPN	2	0.101(±0.518)	71.86(±61.45)	24.81(±31.99)	1571.97(±2001.32)	0.301(±0.347)	0.02(±0.00)
		HPN	2	0.859(±0.416)	29.50(±20.17)	19.91(±15.17)	9.11(±8.48)	0.705(±0.136)	0.21(±0.01)
		UT@LW	2	0.916(±0.388)	25.29(±17.43)	11.11(±9.79)	12.46(±13.13)	0.685(±0.124)	0.09(±0.01)
		UT@FN	2	0.805(±0.442)	31.99(±23.11)	7.24(±6.33)	8.51(±9.57)	0.678(±0.140)	0.11(±0.00)
		WGMprop@LPN	558.15(±143.52)	1.545(±0.963)	114.32(±88.74)	39.09(±60.09)	35.73(±139.52)	0.693(±0.220)	27.66(±5.31)
		WGMprop@HPN	138.44(±50.48)	0.027(±0.031)	2.45(±4.40)	0.16(±0.10)	0.13(±0.17)	0.987(±0.009)	17.39(±6.26)
		WGMprop@UT	122.07(±46.22)	0.049(±0.033)	2.20(±4.38)	0.31(±0.21)	0.31(±0.27)	0.976(±0.017)	9.98(±3.71)
		WGMprop@FN	371.46(±196.76)	0.179(±0.208)	12.76(±16.35)	2.06(±3.10)	2.79(±4.62)	0.844(±0.172)	32.44(±17.04)

Table 7. Performance comparison for all type of noise and intensity, and with $K = 3$. Standard deviations are presented in $(\pm)$.

				PERFORMANCE CRITERIA					
Noise	**Intensity**	**Method**	**# Gaussian**	**2W**	**KL**	**MEAN**	**STD**	**IOU$_{95}$**	**TIME (s)**
Gaussian Noise	I_1	LPN	3	0.068(±0.138)	0.37(±0.30)	0.16(±0.13)	93.03(±3.49)	0.055(±0.021)	0.03(±0.00)
		HPN	3	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.61)	0.948(±0.039)	0.31(±0.02)
		UT@LW	3	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.88(±2.79)	0.931(±0.039)	0.13(±0.01)
		UT@FN	3	0.019(±0.034)	0.04(±0.03)	0.06(±0.10)	8.88(±10.38)	0.907(±0.074)	0.20(±0.01)
		WGMprop@LPN	10.86(±8.17)	0.292(±0.327)	0.32(±0.27)	0.16(±0.13)	87.01(±9.72)	0.125(±0.100)	0.82(±0.51)
		WGMprop@HPN	10.85(±8.16)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.64(±0.58)	0.950(±0.037)	1.54(±1.06)
		WGMprop@UT	3.00(±00)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.88(±2.79)	0.931(±0.039)	0.40(±0.01)
		WGMprop@FN	6.44(±2.97)	0.018(±0.027)	0.04(±0.03)	0.06(±0.06)	8.85(±10.38)	0.908(±0.074)	0.77(±0.31)
	I_2	LPN	3	0.068(±0.138)	0.37(±0.30)	0.16(±0.13)	93.03(±3.49)	0.055(±0.021)	0.03(±0.00)
		HPN	3	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.61)	0.948(±0.039)	0.31(±0.02)
		UT@LW	3	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.88(±2.79)	0.931(±0.039)	0.13(±0.01)
		UT@FN	3	0.019(±0.034)	0.04(±0.03)	0.06(±0.10)	8.88(±10.38)	0.907(±0.074)	0.21(±0.01)
		WGMprop@LPN	10.86(±8.17)	0.292(±0.327)	0.32(±0.27)	0.16(±0.13)	87.01(±9.72)	0.125(±0.100)	0.82(±0.51)
		WGMprop@HPN	10.85(±8.16)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.64(±0.58)	0.950(±0.037)	1.54(±1.05)
		WGMprop@UT	3.00(±00)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.88(±2.79)	0.931(±0.039)	0.40(±0.01)
		WGMprop@FN	6.44(±2.97)	0.018(±0.027)	0.04(±0.03)	0.06(±0.06)	8.85(±10.38)	0.908(±0.074)	0.78(±0.31)
	I_3	LPN	3	0.068(±0.138)	0.37(±0.30)	0.16(±0.13)	93.03(±3.49)	0.055(±0.021)	0.03(±0.01)
		HPN	3	0.005(±0.007)	0.03(±0.02)	0.00(±0.00)	0.65(±0.61)	0.948(±0.039)	0.31(±0.02)
		UT@LW	3	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.88(±2.79)	0.931(±0.039)	0.41(±0.01)
		UT@FN	6.44(±2.97)	0.018(±0.027)	0.04(±0.03)	0.06(±0.06)	8.85(±10.38)	0.908(±0.074)	0.77(±0.31)
		WGMprop@LPN	10.86(±8.17)	0.292(±0.327)	0.32(±0.27)	0.16(±0.13)	87.01(±9.72)	0.125(±0.100)	0.83(±0.52)
		WGMprop@HPN	10.85(±8.16)	0.005(±0.006)	0.02(±0.02)	0.00(±0.00)	0.64(±0.58)	0.950(±0.037)	1.57(±1.07)
		WGMprop@UT	3.00(±00)	0.008(±0.008)	0.02(±0.02)	0.08(±0.09)	3.88(±2.79)	0.931(±0.039)	0.13(±0.01)
		WGMprop@FN	6.44(±2.97)	0.018(±0.027)	0.04(±0.03)	0.06(±0.06)	8.85(±10.38)	0.908(±0.074)	0.77(±0.31)
Blur	I_1	LPN	3	0.190(±0.408)	2.63(±1.70)	0.28(±0.17)	18.46(±14.03)	0.631(±0.094)	0.02(±0.00)
		HPN	3	0.093(±0.191)	0.41(±0.35)	0.02(±0.03)	4.80(±17.61)	0.958(±0.040)	0.31(±0.01)
		UT@LW	3	0.029(±0.030)	0.33(±0.26)	0.10(±0.08)	2.63(±1.75)	0.936(±0.038)	0.12(±0.00)
		UT@FN	3	0.108(±0.074)	3.87(±5.43)	4.65(±5.13)	9.47(±6.61)	0.926(±0.078)	0.16(±0.01)
		WGMprop@LPN	8.47(±2.23)	0.259(±0.432)	0.73(±0.77)	0.16(±0.12)	5.17(±5.17)	0.801(±0.075)	0.68(±0.14)
		WGMprop@HPN	7.23(±3.15)	0.085(±0.192)	0.15(±0.20)	0.02(±0.01)	4.39(±16.40)	0.975(±0.033)	1.07(±0.42)
		WGMprop@UT	5.37(±2.99)	0.023(±0.028)	0.20(±0.21)	0.07(±0.07)	1.63(±1.68)	0.958(±0.036)	0.58(±0.25)
		WGMprop@FN	51.66(±25.87)	0.018(±0.044)	0.02(±0.03)	0.02(±0.01)	0.44(±2.07)	0.983(±0.046)	4.75(±2.33)
	I_2	LPN	3	0.007(±0.300)	5.44(±2.95)	0.55(±0.53)	6.14(±6.48)	0.846(±0.096)	0.03(±0.00)
		HPN	3	0.123(±0.082)	1.09(±0.66)	0.14(±0.15)	0.69(±0.81)	0.938(±0.041)	0.32(±0.02)
		UT@LW	3	0.182(±0.121)	0.94(±0.51)	0.51(±0.38)	2.05(±1.88)	0.922(±0.055)	0.12(±0.00)
		UT@FN	3	0.230(±0.140)	6.08(±5.61)	8.27(±6.73)	4.83(±3.31)	0.896(±0.065)	0.17(±0.00)
		WGMprop@LPN	20.45(±4.00)	0.169(±0.270)	0.38(±0.23)	0.11(±0.12)	1.11(±1.34)	0.969(±0.035)	1.41(±0.24)
		WGMprop@HPN	18.64(±3.33)	0.023(±0.032)	0.12(±0.09)	0.01(±0.01)	0.06(±0.08)	0.988(±0.010)	2.58(±0.43)
		WGMprop@UT	16.58(±3.03)	0.054(±0.063)	0.13(±0.10)	0.08(±0.11)	0.28(±0.33)	0.975(±0.020)	1.51(±0.24)
		WGMprop@FN	138.06(±36.60)	0.015(±0.015)	0.06(±0.05)	0.02(±0.02)	0.07(±0.17)	0.990(±0.012)	12.61(±3.32)
	I_3	LPN	3	0.027(±0.163)	3.70(±1.65)	0.49(±0.39)	7.28(±7.15)	0.810(±0.097)	0.02(±0.00)
		HPN	3	0.199(±0.099)	1.09(±0.72)	0.12(±0.11)	1.08(±0.61)	0.902(±0.033)	0.31(±0.02)
		UT@LW	3	0.245(±0.129)	1.01(±0.51)	0.29(±0.29)	1.86(±1.36)	0.881(±0.044)	0.12(±0.00)
		UT@FN	3	0.365(±0.155)	11.98(±12.96)	14.65(±14.25)	10.34(±7.19)	0.832(±0.058)	0.17(±0.00)
		WGMprop@LPN	21.55(±3.16)	0.263(±0.240)	0.24(±0.12)	0.07(±0.05)	1.30(±1.07)	0.962(±0.027)	1.48(±0.19)
		WGMprop@HPN	20.96(±2.54)	0.024(±0.023)	0.12(±0.11)	0.01(±0.01)	0.10(±0.08)	0.987(±0.008)	2.88(±0.35)
		WGMprop@UT	19.67(±2.83)	0.048(±0.048)	0.12(±0.09)	0.05(±0.11)	0.28(±0.41)	0.977(±0.017)	1.75(±0.23)
		WGMprop@FN	131.53(±30.41)	0.027(±0.022)	0.08(±0.06)	0.02(±0.02)	0.07(±0.09)	0.982(±0.014)	11.96(±2.73)
Contrast	I_1	LPN	3	0.258(±0.439)	6.91(±8.19)	0.74(±0.69)	47.64(±99.38)	0.542(±0.176)	0.03(±0.00)
		HPN	3	0.138(±0.178)	1.65(±1.58)	0.10(±0.14)	7.95(±43.90)	0.874(±0.092)	0.32(±0.02)
		UT@LW	3	0.077(±0.076)	1.36(±1.33)	0.37(±0.46)	7.02(±6.05)	0.834(±0.094)	0.13(±0.00)
		UT@FN	3	0.202(±0.136)	9.59(±12.48)	6.93(±8.53)	16.81(±13.57)	0.825(±0.092)	0.16(±0.00)
		WGMprop@LPN	14.11(±2.13)	0.447(±0.474)	0.30(±0.31)	0.12(±0.12)	3.86(±2.42)	0.876(±0.056)	1.02(±0.13)
		WGMprop@HPN	12.64(±3.29)	0.093(±0.174)	0.20(±0.21)	0.01(±0.01)	6.64(±41.50)	0.973(±0.036)	1.78(±0.42)
		WGMprop@UT	11.15(±3.29)	0.021(±0.034)	0.24(±0.25)	0.05(±0.06)	0.69(±0.85)	0.966(±0.032)	1.05(±0.26)
		WGMprop@FN	90.85(±53.80)	0.014(±0.039)	0.10(±0.12)	0.01(±0.01)	0.28(±1.90)	0.981(±0.052)	8.18(±4.79)
	I_2	LPN	3	0.162(±0.413)	23.78(±12.53)	6.16(±4.20)	30.62(±56.14)	0.635(±0.159)	0.03(±0.00)
		HPN	3	0.676(±0.237)	17.89(±8.59)	3.70(±2.47)	10.26(±3.63)	0.746(±0.079)	0.30(±0.02)
		UT@LW	3	0.663(±0.208)	13.54(±5.09)	2.44(±1.90)	9.13(±5.15)	0.760(±0.117)	0.13(±0.01)
		UT@FN	3	0.859(±0.453)	57.17(±62.62)	27.73(±51.36)	10.22(±14.92)	0.696(±0.082)	0.16(±0.01)
		WGMprop@LPN	342.99(±125.17)	0.335(±0.419)	6.72(±22.02)	2.85(±9.72)	1.31(±4.41)	0.967(±0.088)	19.92(±6.56)
		WGMprop@HPN	70.21(±18.98)	0.028(±0.030)	2.61(±3.54)	0.10(±0.05)	0.12(±0.16)	0.992(±0.009)	8.91(±2.35)
		WGMprop@UT	56.98(±14.85)	0.041(±0.028)	2.67(±3.85)	0.15(±0.09)	0.33(±0.24)	0.986(±0.014)	4.77(±1.19)
		WGMprop@FN	345.40(±99.62)	0.021(±0.022)	2.20(±3.23)	0.07(±0.04)	0.09(±0.56)	0.994(±0.013)	30.55(±8.81)
	I_3	LPN	3	0.167(±0.423)	16.85(±10.33)	4.36(±3.56)	7.95(±18.25)	0.845(±0.149)	0.02(±0.00)
		HPN	3	0.223(±0.107)	9.31(±5.72)	2.24(±1.80)	4.55(±3.80)	0.835(±0.134)	0.31(±0.02)
		UT@LW	3	0.250(±0.113)	7.03(±4.11)	1.84(±1.53)	3.16(±2.27)	0.808(±0.142)	0.12(±0.00)
		UT@FN	3	0.331(±0.148)	43.43(±37.50)	20.60(±22.83)	4.98(±7.01)	0.742(±0.073)	0.16(±0.00)
		WGMprop@LPN	287.24(±78.33)	0.349(±0.461)	2.52(±4.01)	0.04(±0.18)	0.09(±0.20)	0.988(±0.016)	17.33(±4.69)
		WGMprop@HPN	50.65(±14.57)	0.015(±0.033)	4.53(±6.30)	0.02(±0.01)	0.06(±0.11)	0.994(±0.009)	6.67(±1.89)
		WGMprop@UT	42.70(±11.48)	0.018(±0.012)	4.64(±6.38)	0.06(±0.05)	0.14(±0.12)	0.993(±0.011)	3.64(±0.94)
		WGMprop@FN	231.09(±59.52)	0.009(±0.008)	3.97(±5.49)	0.01(±0.01)	0.03(±0.16)	0.997(±0.008)	20.53(±5.32)

B.2.2. BOXPLOT RAW DATA TABLES

In this subsection we provide the boxplot for all noises and values of $K \in \{1, 2, 3\}$ and intensity I_3.

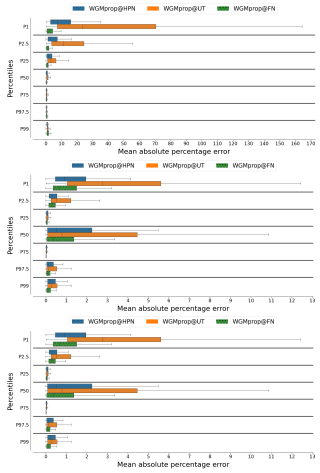


Figure 5. Mean absolute percentage error over the percentiles 1, 2.5, 25, 50, 75, 97.5 and 99 between WGMprop methods and MC reference for contrast noise of intensity I_3 and $K = 1$ (**Top**), $K = 2$ (**Middle**) and $K = 3$ (**Bottom**).

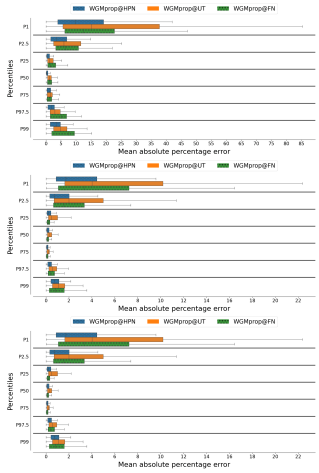


Figure 6. Mean absolute percentage error over the percentiles 1, 2.5, 25, 50, 75, 97.5 and 99 between WGMprop methods and MC reference for blur noise of intensity I_3 and $K = 1$ (**Top**), $K = 2$ (**Middle**) and $K = 3$ (**Bottom**).

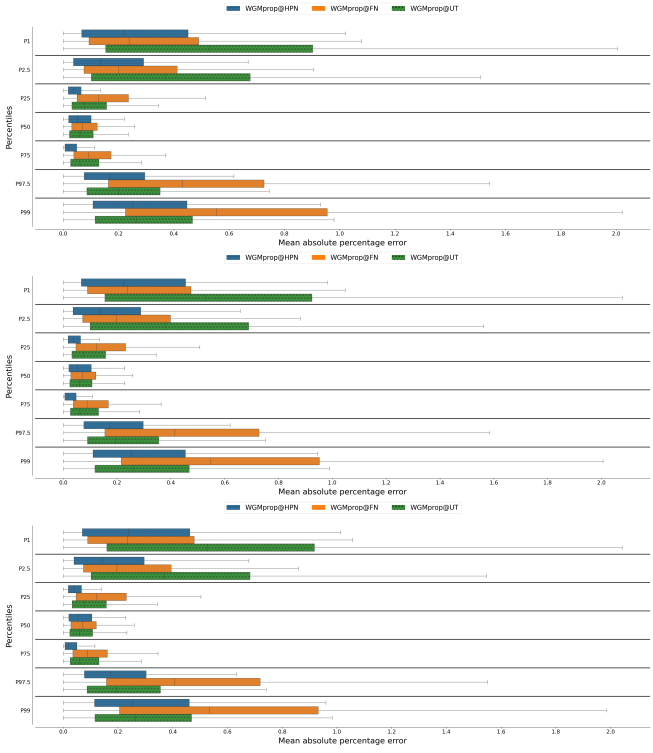


Figure 7. Mean absolute percentage error over the percentiles 1, 2.5, 25, 50, 75, 97.5 and 99 between WGMprop methods and MC reference for Gaussian noise of intensity I_3 and $K = 1$ (**Top**), $K = 2$ (**Middle**) and $K = 3$ (**Bottom**).

B.2.3. More figures

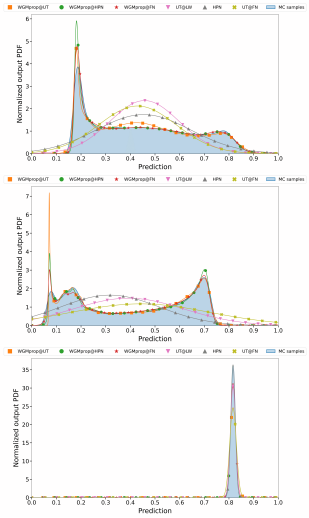


Figure 8. Estimated output PDFs (main marginal) for sample n°58 corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity and $K = 1$), MC reference filled in blue.

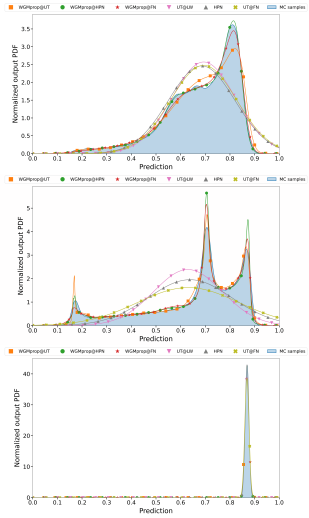


Figure 9. Estimated output PDFs (main marginal) for sample n°181 corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity and $K = 1$), MC reference filled in blue.

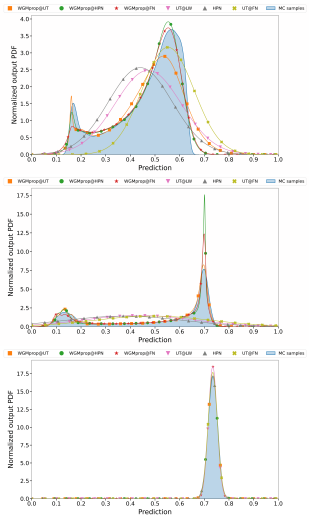


Figure 10. Estimated output PDFs (main marginal) for sample n°416 corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity and $K = 1$), MC reference filled in blue.

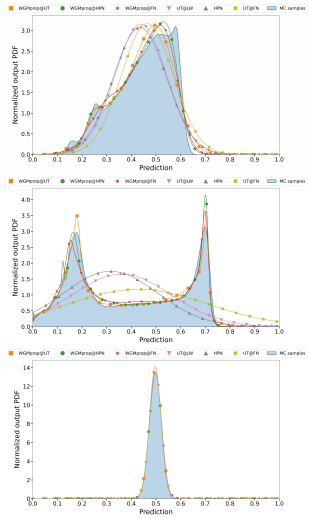


Figure 11. Estimated output PDFs (main marginal) for sample n°904 corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity and $K = 1$), MC reference filled in blue.

B.2.4. Analysis

From an overall perspective, state-of-the-art methods are way faster than WGMprop methods at the cost of accuracy. Indeed, no matter the input noise complexity, the mean prediction time of these methods stays constant ($\approx 10^{-1}$) while their prediction errors rise dramatically (the IOU_{95} drops from 0.948 for a Gaussian noise and a HPN propagation to 0.640 for a contrast noise). In fact, in the case of an input corrupted with a Gaussian noise, the output PDF is closely Gaussian (KL value of 0.005 for the HPN method) whereas, for more complex noises, the Gaussian assumption on the output PDF of each layer does not hold.

To illustrate this statement, Figure 11 displays both the estimated output PDFs using the methods listed in Table 4 and the reference PDF (MC) over sample image n°904 corrupted with the whole set of noises at the highest intensity (I_3). In the case of blur noise and contrast noise, UT@LW, UT@FN and HPN methods suffer from shape misrepresentation due to their Gaussian assumption. On the other hand, WGMprop methods capture well the shape of the output PDF and are therefore good candidates to obtain a reliable estimate of the 95% prediction interval.

Moreover, although the mean prediction time scales with the input noise complexity (more Gaussian components in the mixture are needed), WGMprop methods' predictions remain highly accurate: $\mathbf{IOU_{95}}$ superior to 0.900, $\mathbf{MAPE}_{\text{MEAN}}$ lower than 0.25%. The analytical approach WGMprop@HPN achieves the highest $\mathbf{IOU_{95}}$ value for all studied noise models. WGMprop@HPN also produces the lowest error values with respect to 2W, KL, $\mathbf{MAPE}_{\text{MEAN}}$ and $\mathbf{MAPE}_{\text{STD}}$; except for the contrast noise where WGMprop@FN performs slightly better on $\mathbf{MAPE}_{\text{MEAN}}$ and $\mathbf{MAPE}_{\text{STD}}$ at the cost of a high prediction time (5 times as many Gaussian components in the mixture compared to WGMprop@UT (97.04) and WGMprop@HPN (108.15)). In terms of mean prediction time, WGMprop@UT is the fastest for the whole experiment. This higher speed is clearly correlated with the reduced number of Gaussian components in the mixture.

In addition, the boxplot of the MAPE of several predicted percentiles (1, 2.5, 25, 25, 97.5 and 99) are shown in Figure 4. These boxplots present the MAPE distributions over the different percentiles of the output PDF for the 3 mixture-based methods: WGMprop@HPN, WGMprop@FN and WGMprop@UT. Firstly, the proposed methods exhibit comparable performances with an overlap of the MAPE distributions for all estimated percentiles. The methods show larger MAPE errors as one moves towards the tails of the distribution (P1, P99) yet behave similarly. WGMprop@HPN has both the lowest median values and the smallest dispersion of errors from P1 to P99.

However, in a layer-wise propagation paradigm such as in WGMprop@HPN and WGMprop@UT, the propagation can become infeasible for deeper and wider neural networks. In fact, a layer-wise propagation induces many extra computational costs:

- After the first layer, each Gaussian component of the mixture does not have the same covariance matrix and such forced to apply N inverse where N is the number of components of the current Gaussian mixture distribution,
- To avoid the last point, we can reduce N by applying a merging phase which can be costly in practice, needing the computation of $\frac{N(N-1)}{2}$ bounds,
- The curse of dimensionality can arise from the number of neurons in a specific layer, contrary to a full network propagation where the dimensionality bottleneck only arise on the input or output layer of the network.

For all these reasons, our full network approach, based on UT sampling and a Wasserstein based criterion is to be favored for its genericity and simplicity.

B.3. More complex dataset

B.3.1. CIFAR10 DATASET

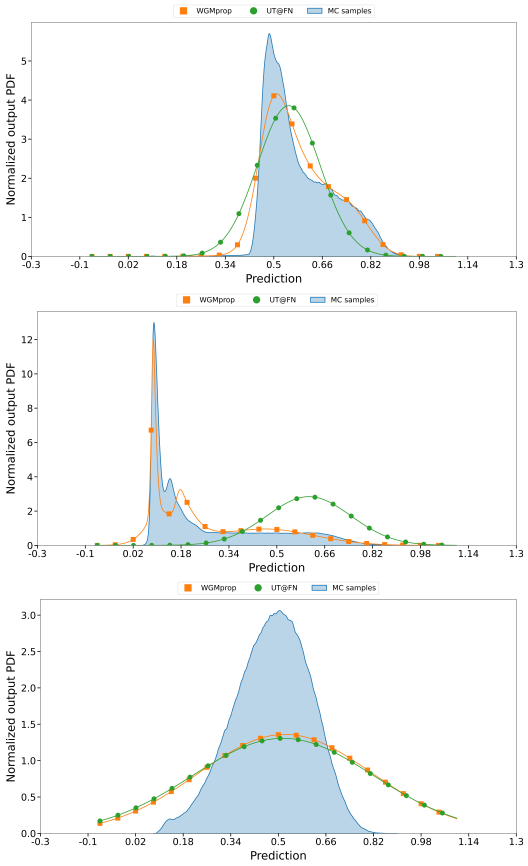


Figure 12. Estimated output PDFs (main marginal) for sample n°2 of the CIFAR10 dataset corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

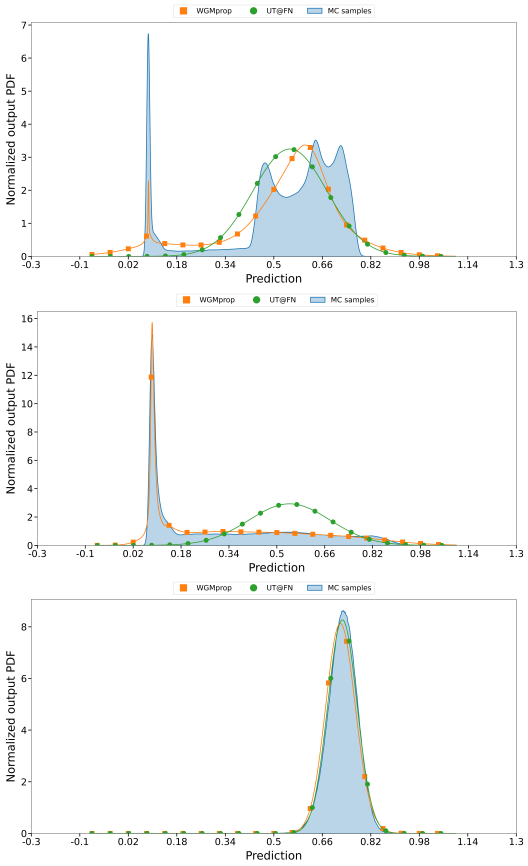


Figure 13. Estimated output PDFs (main marginal) for sample n°89 of the CIFAR10 dataset corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

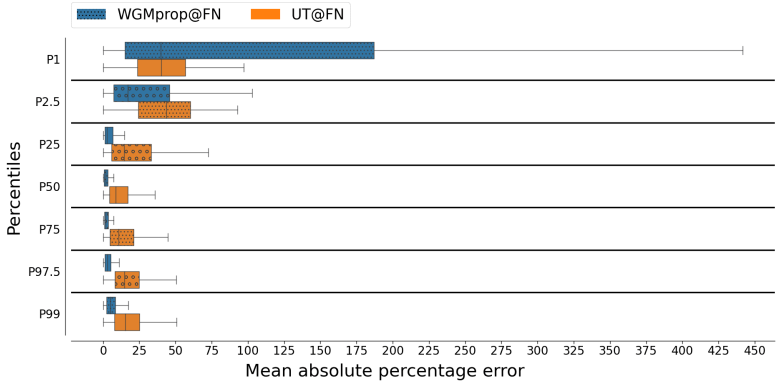


Figure 14. MAPE boxplots for CIFAR10 dataset and for percentiles 1, 2.5, 25, 50, 75, 97.5 and 99. Input images were degraded by a large contrast kernel (I_3).

B.3.2. CIFAR100 DATASET

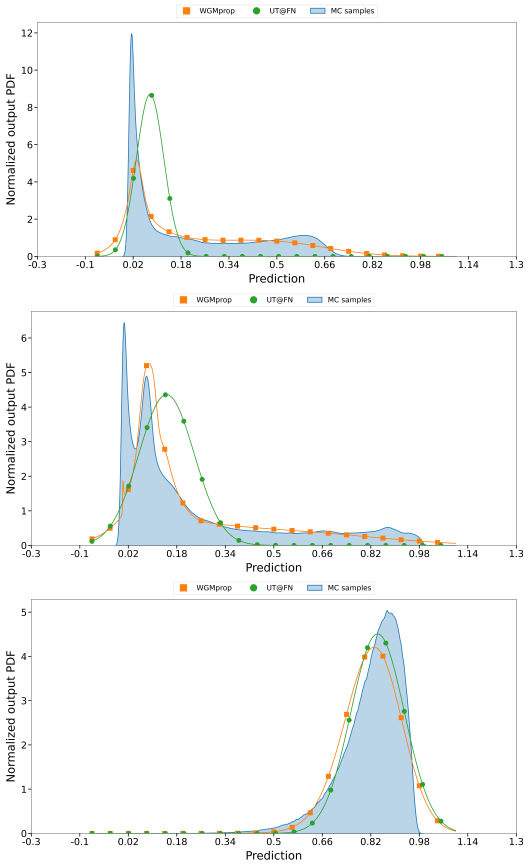


Figure 15. Estimated output PDFs (main marginal) for sample n°2 of the CIFAR100 dataset corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

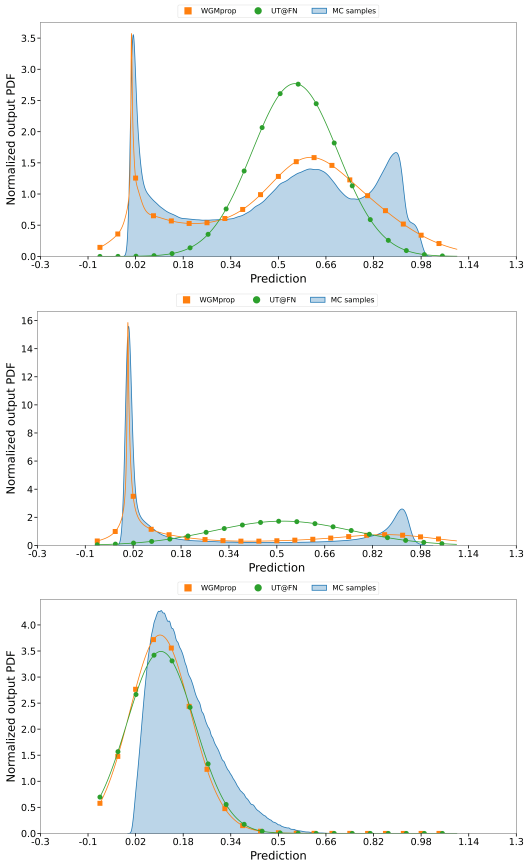


Figure 16. Estimated output PDFs (main marginal) for sample n°89 of the CIFAR100 dataset corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

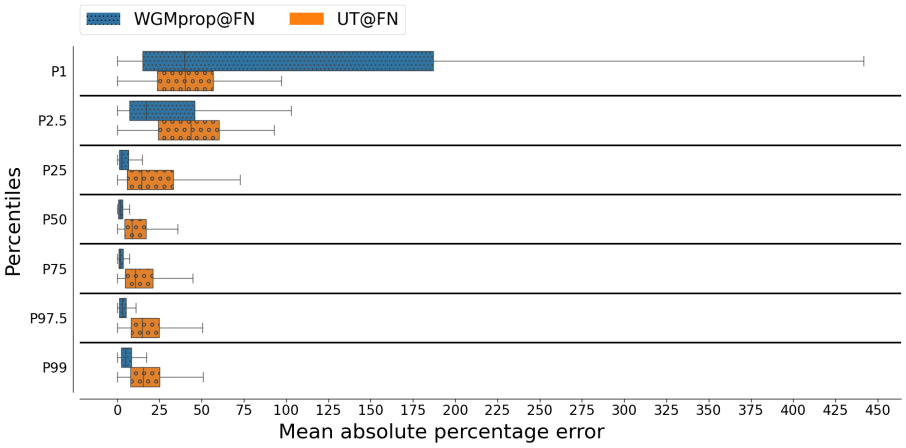


Figure 17. MAPE boxplots for CIFAR100 dataset and for percentiles 1, 2.5, 25, 50, 75, 97.5 and 99. Input images were degraded by a large contrast kernel (I_3).

B.3.3. CAMELYON DATASET

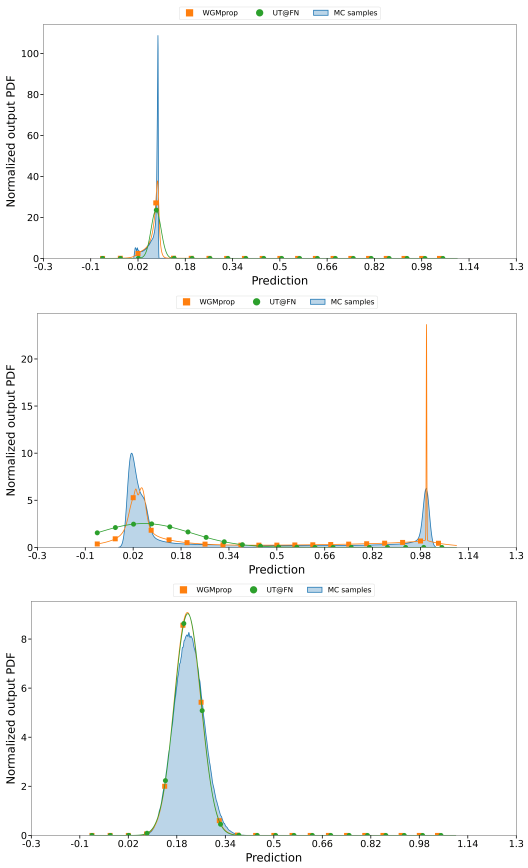


Figure 18. Estimated output PDFs (main marginal) for sample n°2 of the Camelyon dataset corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

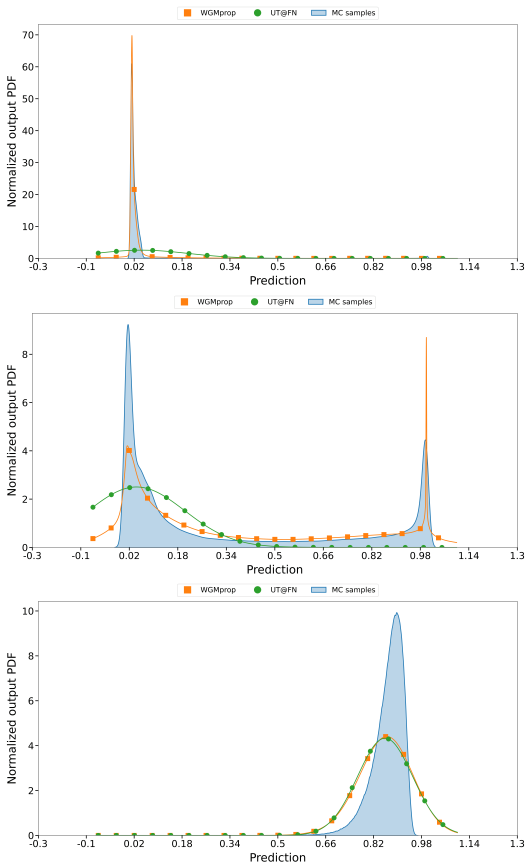


Figure 19. Estimated output PDFs (main marginal) for sample n°89 of the Camelyon dataset corrupted by blur kernel (**Top**), contrast noise (**Middle**) and Gaussian-distributed additive noise (**Bottom**) (I_3 intensity). The output PDF estimated using Monte Carlo propagation (reference) is filled in blue.

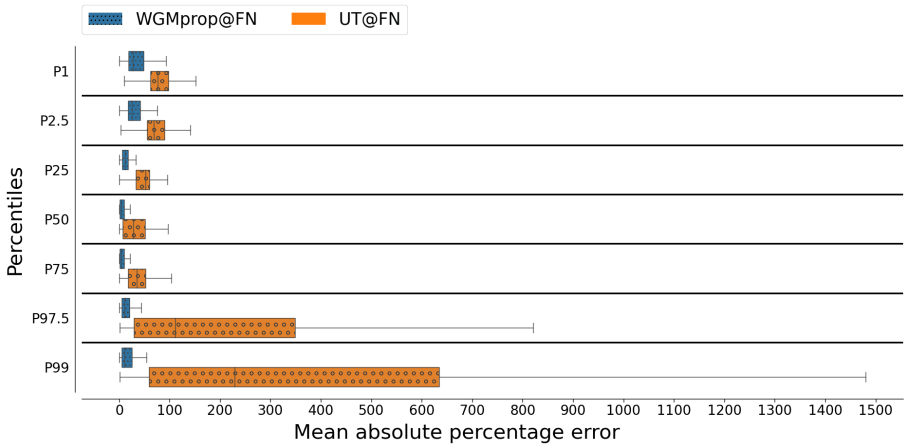


Figure 20. MAPE boxplots for Camelyon dataset and for percentiles 1, 2.5, 25, 50, 75, 97.5 and 99. Input images were degraded by a large contrast kernel (I_3).

C. Analytical moment propagation through classical activation functions.

For fully connected network and CNN, we provide the analytical formula to propagate the first two moments of a Gaussian distribution through ReLU, leaky ReLU (Maas et al., 2013) or ELU (Clevert et al., 2015) activation function.
Due to the excessive length of the proofs, they are only given in the supplementary files available at https://github.com/PaulMcht/WGMprop.

C.1. ReLU activation function

Proposition C.1. *Let X be a random variable with $X \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\mu} \in \mathbb{R}^d$, $\boldsymbol{\Sigma} \in \mathbb{S}_d^+(\mathbb{R})$ with rank$(\boldsymbol{\Sigma}) = r$, $r \in \mathbb{N}^*$ and f be the rectified linear function, then for the random variable $\boldsymbol{Y} = (Y_1, ..., Y_d) = f(\boldsymbol{X}) = (f(X_1), ..., f(X_d))$, $\forall k, k' \in \{1...d\}$:*

$$\begin{aligned}
\mathbb{E}[Y_k] &= c_k\phi_k + \mu_k\Phi_k && (11)\\
\mathbb{E}[Y_k^2] &= (\mu_k^2 + c_k^2)\Phi_k + \mu_k c_k \phi_k && (12)\\
\mathbb{E}[Y_k Y_{k'}] &= c_k\alpha\phi_{k'}\phi_{NS} + (\mu_k\mu_{k'} + c_k\beta)\left(\Phi_{k'} - \Phi_{kk'}\right) + \mu_k c_{k'}\phi_{k'}\Phi_S + \mu_{k'}c_k\phi_k\Phi_{NS} && (13)
\end{aligned}$$

where: $\mu_k = \mathbb{E}[X_k]$, the mean of the k^{th} component of the random variable $\boldsymbol{X}$, $Q_r = (q_k)_{k\in\{1...d\}} \in \mathcal{M}_{r,d}(\mathbb{R})$ the r first line of Q such that $\Sigma = Q^T\Lambda Q = Q_r^T\Lambda_r Q_r$, and $\Lambda_r \in \mathcal{D}_r(\mathbb{R})$. Then, we have $c_k = \left\|\sqrt{\Lambda_r}q_k\right\|$, $c_{k-k'} = \left\|\sqrt{\Lambda_r}(q_k - q_{k'})\right\|$, $\beta = \frac{c_k^2 + c_j^2 - c_{k-k'}^2}{2c_k}$, $\alpha = \sqrt{c_j^2 - \beta^2}$. Finally, we note $\phi_k = \phi_{0,1}\left(\frac{\mu_k}{c_k}\right)$, $\Phi_{k'} = \Phi_{0,1}\left(\frac{\mu_{k'}}{c_{k'}}\right)$, $\Phi_{NS} = \Phi_{0,1}\left(\frac{\mu_{k'}}{\alpha} - \frac{\beta\mu_k}{\alpha c_k}\right)$, $\Phi_S = \Phi_{0,1}\left(\frac{c_{k'}\mu_k}{c_k\alpha} - \frac{\beta\mu_{k'}}{\alpha c_{k'}}\right)$, $\phi_{NS} = \phi_{0,1}\left(\frac{\beta\mu_{k'}}{\alpha c_{k'}} - \frac{\mu_k c_{k'}}{c_k\alpha}\right)$ and $\Phi_{kk'} = \Phi_{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & -\frac{\beta}{c_{k'}}\\ -\frac{\beta}{c_{k'}} & 1\end{bmatrix}}\left(\left[\frac{\mu_{k'}}{c_{k'}}, -\frac{\mu_k}{c_k}\right]^T\right)$ and ϕ and Φ are respectively the PDF and CDF of the standard normal distribution.

C.2. Leaky ReLU activation function

Proposition C.2. *Let X be a random variable with $X \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\mu} \in \mathbb{R}^d$, $\boldsymbol{\Sigma} \in \mathbb{S}_d^+(\mathbb{R})$ with rank$(\boldsymbol{\Sigma}) = r$, $r \in \mathbb{N}^*$ and f be the leaky rectified linear function of parameters $\lambda \in [0, 1[$, then for the random variable $\boldsymbol{Y} = (Y_1, ..., Y_d) = f(\boldsymbol{X}) = (f(X_1), ..., f(X_d))$, $\forall k, k' \in \{1...d\}$:*

$$\begin{aligned}
\mathbb{E}[Y_k] &= (1-\lambda)c_k\phi_k + (1-\lambda)\mu_k\Phi_k + \lambda\mu_k && (14)\\
\mathbb{E}[Y_k^2] &= (1-\lambda^2)(\mu_k^2 + c_k^2)\Phi_k + (1-\lambda^2)\mu_k c_k\phi_k + \lambda^2(c_k^2 + \mu_k^2) && (15)\\
\mathbb{E}[Y_k Y_{k'}] &= (1-\lambda)^2 c_k\alpha\phi_{k'}\phi_{NS}\\
&+ (\mu_k\mu_{k'} + c_k\beta)\left((1-\lambda)\Phi_{k'} - (1-\lambda)^2\Phi_{kk'} + \lambda(1-\lambda)\Phi_k + \lambda^2\right)\\
&+ (1-\lambda)^2\mu_k c_{k'}\phi_{k'}\Phi_S\\
&+ (1-\lambda)^2\mu_{k'}c_k\phi_k\Phi_{NS}\\
&+ \lambda(1-\lambda)(\mu_{k'}c_k\phi_k + \mu_k c_{k'}\phi_{k'}) && (16)
\end{aligned}$$

where: $\mu_k = \mathbb{E}[X_k]$, the mean of the k^{th} component of the random variable $\boldsymbol{X}$, $Q_r = (q_k)_{k\in\{1...d\}} \in \mathcal{M}_{r,d}(\mathbb{R})$ the r first line of Q such that $\Sigma = Q^T\Lambda Q = Q_r^T\Lambda_r Q_r$, and $\Lambda_r \in \mathcal{D}_r(\mathbb{R})$. Then, we have $c_k = \left\|\sqrt{\Lambda_r}q_k\right\|$, $c_{k-k'} = \left\|\sqrt{\Lambda_r}(q_k - q_{k'})\right\|$, $\beta = \frac{c_k^2 + c_j^2 - c_{k-k'}^2}{2c_k}$, $\alpha = \sqrt{c_j^2 - \beta^2}$. Finally, we note $\phi_k = \phi_{0,1}\left(\frac{\mu_k}{c_k}\right)$, $\Phi_{k'} = \Phi_{0,1}\left(\frac{\mu_{k'}}{c_{k'}}\right)$, $\Phi_{NS} = \Phi_{0,1}\left(\frac{\mu_{k'}}{\alpha} - \frac{\beta\mu_k}{\alpha c_k}\right)$, $\Phi_S = \Phi_{0,1}\left(\frac{c_{k'}\mu_k}{c_k\alpha} - \frac{\beta\mu_{k'}}{\alpha c_{k'}}\right)$, $\phi_{NS} = \phi_{0,1}\left(\frac{\beta\mu_{k'}}{\alpha c_{k'}} - \frac{\mu_k c_{k'}}{c_k\alpha}\right)$ and $\Phi_{kk'} = \Phi_{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & -\frac{\beta}{c_{k'}}\\ -\frac{\beta}{c_{k'}} & 1\end{bmatrix}}\left(\left[\frac{\mu_{k'}}{c_{k'}}, -\frac{\mu_k}{c_k}\right]^T\right)$ and ϕ and Φ are respectively the PDF and CDF of the standard normal distribution.

C.2.1. ELU ACTIVATION FUNCTION

Proposition C.3. *Let X be a random variable with $X \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\mu} \in \mathbb{R}^d$, $\boldsymbol{\Sigma} \in \mathbb{S}_d^+(\mathbb{R})$ with rank$(\boldsymbol{\Sigma}) = r$, $r \in \mathbb{N}^*$ and f be the exponential linear function, then for the random variable $\boldsymbol{Y} = (Y_1, ..., Y_d) = f(\boldsymbol{X}) = (f(X_1), ..., f(X_d))$, $\forall k, k' \in \{1...d\}$:*

$$\begin{aligned}
\mathbb{E}[Y_k] &= c_k\phi_k + (\mu_k + \lambda)\Phi_k + \lambda e^{\mu_k + \frac{c_k^2}{2}}(1 - \Phi_{kc_k}) - \lambda && (17)\\
\mathbb{E}[Y_k^2] &= (\mu_i^2 + c_i^2 - \lambda^2)\Phi_i + \mu_i c_i \phi_i + \lambda^2 e^{2(c_i^2+\mu_i)}(1 - \Phi_{2c_i}) - 2\lambda^2 e^{\mu_i + \frac{c_i^2}{2}}(1 - \Phi_{c_i})\\
&+ \lambda^2 && (18)\\
\mathbb{E}[Y_k Y_{k'}] &= c_i\alpha\phi_j\phi_{NS}\\
&+ \left(\mu_i\mu_j + c_i\beta + \lambda\mu_i\right)\Phi_j\\
&- \left(\mu_i\mu_j + c_i\beta + \lambda(\mu_i + \mu_j) + \lambda^2\right)\Phi_{ij}\\
&+ (\mu_i c_j\phi_j + \lambda c_j\phi_j + \frac{\lambda\beta c_i}{c_j}((\phi_j - e^{\mu_j + \frac{c_j^2}{2}}\phi_{c_j}))\Phi_S\\
&+ (\mu_j c_i\phi_i + \lambda c_i\phi_i + \lambda\beta(\phi_i - e^{\mu_i + \frac{c_i^2}{2}}\phi_{c_i}))\Phi_{NS}\\
&+ \lambda c_i e^{\mu_j + \frac{c_j^2}{2}}\phi_{\beta_i}\Phi_\alpha\\
&+ \lambda c_j e^{\mu_i + \frac{c_i^2}{2}}\phi_a\Phi_e\\
&+ \lambda^2 e^{\mu_i + \frac{c_i^2}{2}}\Phi_{c_i}\\
&- \lambda e^{\mu_j + \frac{c_j^2}{2}}(\mu_i + \beta c_i)\Phi_{c_j}\\
&+ \lambda e^{\mu_i + \frac{c_i^2}{2}}(\lambda + \beta c_i + \mu_j)\Phi_A\\
&+ \lambda e^{\mu_j + \frac{c_j^2}{2}}(\lambda + \beta c_i + \mu_i)(\Phi_{\beta_i} + \Phi_B)\\
&+ \lambda^2 e^{\mu_i + \mu_j + \frac{c_i^2 + c_j^2}{2} + c_i\beta}(1 - \Phi_{\beta + c_i} - \Phi_C)\\
&- \lambda(\lambda e^{\mu_i + \frac{c_i^2}{2}} + \lambda e^{\mu_j + \frac{c_j^2}{2}} + c_i\phi_i + c_j\phi_j - \lambda) && (19)
\end{aligned}$$

where: $\mu_k = \mathbb{E}[X_k]$, the mean of the k^{th} component of the random variable $\boldsymbol{X}$, $Q_r = (q_k)_{k\in\{1...d\}} \in \mathcal{M}_{r,d}(\mathbb{R})$ the r first line of Q such that $\Sigma = Q^T\Lambda Q = Q_r^T\Lambda_r Q_r$, and $\Lambda_r \in \mathcal{D}_r(\mathbb{R})$. Then, we have $c_k = \left\|\sqrt{\Lambda_r}q_k\right\|$, $c_{k-k'} = \left\|\sqrt{\Lambda_r}(q_k - q_{k'})\right\|$, $\beta = \frac{c_k^2 + c_j^2 - c_{k-k'}^2}{2c_k}$, $\alpha = \sqrt{c_j^2 - \beta^2}$. Finally, we note $\phi_k = \phi_{0,1}\left(\frac{\mu_k}{c_k}\right)$, $\Phi_{k'} = \Phi_{0,1}\left(\frac{\mu_{k'}}{c_{k'}}\right)$, $\Phi_{NS} = \Phi_{0,1}\left(\frac{\mu_{k'}}{\alpha} - \frac{\beta\mu_k}{\alpha c_k}\right)$, $\Phi_S = \Phi_{0,1}\left(\frac{c_{k'}\mu_k}{c_k\alpha} - \frac{\beta\mu_{k'}}{\alpha c_{k'}}\right)$, $\phi_{NS} = \phi_{0,1}\left(\frac{\beta\mu_{k'}}{\alpha c_{k'}} - \frac{\mu_k c_{k'}}{c_k\alpha}\right)$, $\Phi_{kk'} = \Phi_{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & -\frac{\beta}{c_{k'}}\\ -\frac{\beta}{c_{k'}} & 1\end{bmatrix}}\left(\left[\frac{\mu_{k'}}{c_{k'}}, -\frac{\mu_k}{c_k}\right]^T\right)$, $\phi_a = \phi_{0,1}\left(\frac{\mu_j + \beta c_i}{c_j}\right)$, $\Phi_{\beta_i} = \Phi_{0,1}\left(\frac{\mu_i}{c_i} + \beta\right)$, $\Phi_e = \Phi_{0,1}\left(\frac{\beta}{\alpha}\frac{\mu_j + \beta c_i}{c_j} - \frac{c_j\mu_i}{\alpha c_i} - \frac{c_j c_i}{\alpha}\right)$, $\Phi_\alpha = \Phi_{0,1}\left(\frac{\beta\mu_i}{\alpha c_i} - \frac{\mu_j}{\alpha} - \alpha\right)$, $\Phi_{\beta+c_i} = \Phi_{0,1}(\frac{\mu_i}{c_i} + c_i + \beta)$, $\Phi_{c_i} = \Phi_{0,1}\left(\frac{\mu_i}{c_i} + c_i\right)$, $\Phi_{2c_i} = \Phi_{0,1}\left(\frac{\mu_i}{c_i} + 2c_i\right)$, $\Phi_A = \Phi_{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & -\frac{\beta}{c_j}\\ -\frac{\beta}{c_j} & 1\end{bmatrix}}\left(\left[\frac{\mu_j + \beta c_i}{c_j}, -\frac{\mu_i}{c_i} - c_i\right]^T\right)$,

$\Phi_B = \Phi_{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & -\frac{\beta}{c_j}\\ -\frac{\beta}{c_j} & 1\end{bmatrix}}\left(\left[\frac{\mu_j}{c_j} + c_j, -\frac{\mu_i}{c_i} - \beta\right]^T\right)$,

and $\Phi_C = \Phi_{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & -\frac{\beta}{c_j}\\ -\frac{\beta}{c_j} & 1\end{bmatrix}}\left(\left[\frac{\mu_j + c_j^2 + \beta c_i}{c_j}, -(\frac{\mu_i}{c_i} + c_i + \beta)\right]^T\right)$,

Finally, ϕ and Φ are respectively the PDF and CDF of the standard normal distribution.

D. Mathematical proofs

D.1. The Wasserstein metric

Here we provide the proofs of proposition 3.1 and 3.2

Proof of Proposition 3.1. Let P_0 and $\tilde{P}_0$ be the centered versions of P and $\tilde{P}$. Using the identity $\mathbb{E}\left[\|A\|^2\right] = \|\mu_A\|^2 + \mathrm{Tr}\left(\mathrm{Cov}\left(A\right)\right)$ for a random vector A, we have

$$\begin{aligned} W_2^2(P,\,\tilde{P}) &= \inf_{\gamma\in\Gamma}\mathbb{E}_{X,Y\sim\gamma}\left[\|X-Y\|^2\right] \\ &= \|\mu_P-\mu_{\tilde{P}}\|^2 + \inf_{\gamma\in\Gamma\left(P,\tilde{P}\right)}\mathrm{Tr}\left(\mathrm{Cov}\left(X-Y\right)\right) \\ &= \|\mu_P-\mu_{\tilde{P}}\|^2 + W_2^2(P_0,\,\tilde{P}_0). \end{aligned}$$

The latter gives the first assertion. Furthermore, let $X_0 \sim P_0$ and $Y_0 \sim \tilde{P}_0$. Using

$$\phi_i\colon\ (x_1,\ldots,x_n)\in\mathbb{R}^d \to x_i$$

and (Villani, 2009, Proposition 7.29) shows that

$$\begin{aligned} W_2(P,\,\tilde{P}) &\geq W_2(P_0,\,\tilde{P}_0) \\ &\geq \left|\sqrt{\int x_i^2 dP_0} - \sqrt{\int x_i^2 dP_0}\right| \\ &= \left|\sigma_{P,i}-\sigma_{\tilde{P},i}\right|. \end{aligned}$$

□

Proof of Proposition 3.2. Let Y and $\tilde{Y}$ two random variables in $\mathbb{R}^d$ with respective distributions P and $\tilde{P}$.

$$W_2(P,\tilde{P}) \geq W_1(P,\tilde{P}) = \inf_{\gamma\in\Gamma(P,\tilde{P})}\mathbb{E}\left[\|Y-\tilde{Y}\|_2^1\right] \tag{20}$$

But, $\forall x\in\mathbb{R}^d : \|x\|_2 \geq \frac{1}{\sqrt{d}}\|x\|_1$, thus:

$$W_2(P,\tilde{P}) \geq \frac{1}{\sqrt{d}}\inf_{\gamma\in\Gamma(P,\tilde{P})}\mathbb{E}\left[\|Y-\tilde{Y}\|_1^1\right] \tag{21}$$

And:

$$\begin{aligned} W_1(P,\tilde{P}) &= \inf_{\gamma\in\Gamma(P,\tilde{P})}\mathbb{E}\left[\|Y-\tilde{Y}\|_1\right] \\ &= \inf_{\gamma\in\Gamma(P,\tilde{P})}\mathbb{E}\left[\sum_{i=1}^d |Y_i-\tilde{Y}_i|\right] \\ &\geq \sum_{i=1}^d \inf_{\gamma_i\in\Gamma(P_i,\tilde{P}_i)}\mathbb{E}\left[|Y_i-\tilde{Y}_i|\right] \\ &\geq \sum_{i=1}^d W_1(P_i,\tilde{P}_i) \end{aligned} \tag{22}$$

The first preceding inequality is obtained by reversing the sum and the infimum and noting that if $\gamma \in \Gamma(P, \tilde{P})$ then by definition, we have $\gamma_i \in \Gamma(P_i, \tilde{P}_i)$. Then, when $d = 1$:

$$W_p(X, Y) = \left\| F_X^{-1} - F_Y^{-1} \right\|_p = \left(\int_0^1 \left| F_X^{-1}(\alpha) - F_Y^{-1}(\alpha) \right|^p \, \mathrm{d}\alpha \right)^{1/p} \tag{23}$$

with F_X and $F_X^{-1}(q) = \inf \{x : F_X(x) \geq q\}, q \in (0, 1)$, the distribution and quantile functions of X.

Thus, by noting Q (respectively $\tilde{Q}$) the quantile function of P (respectively $\tilde{P}$):

$$\frac{1}{\sqrt{d}} \sum_{i=1}^{d} \int_0^1 |Q(q) - \tilde{Q}(q)| dq \leq W_2(P, \tilde{P}) \tag{24}$$

□

D.2. Wasserstein criterion

First, we recall the general notations which will be used in the propositions and their proofs and introduce some new ones.

D.2.1. NOTATIONS

Write $\boldsymbol{U} \in \mathbb{R}^d$ the multidimensional random variable of probability distribution $P_{\boldsymbol{U}}$ and with mean $\mu_{\boldsymbol{U}}$ and covariance matrix $\Sigma_{\boldsymbol{U}}$.

Let $\mathbb{M}$ be the space of mixtures of Gaussian distributions and, for $\boldsymbol{U}$ such that $P_{\boldsymbol{U}} \in \mathbb{M}$, write $P_{\boldsymbol{U}} = \sum_{i=1}^{M} w_i P_{\boldsymbol{U},i}$, with non-negative w_is summing to one and $P_{\boldsymbol{U},i} = \mathcal{N}(\mu_{\boldsymbol{U},i}, \Sigma_{\boldsymbol{U},i})$. We will keep the symbols M and w_i generic for any mixture, unless explicitly stated otherwise.

Furthermore, let $S^{[1]}(\cdot; n)$ be a splitting operator defined on the space of Gaussian distributions and extended to Gaussian mixtures by linearity. $S^{[1]}(\cdot; n)$ approximates $P_{\boldsymbol{U}}$ by a Gaussian mixture distribution of n components:

$$P_{\hat{\boldsymbol{U}}^{[1]}} = S^{[1]}(P_{\boldsymbol{U}}; n) = \sum_{i=1}^{n} w_i P_{\hat{\boldsymbol{U}}^{[1]},i}$$

Given some $\boldsymbol{X} \in \mathbb{R}^d$, the goal is to propagate $P_{\boldsymbol{X}}$ through a $L_{2,2}$-Lipschitz continuous neural network $\boldsymbol{f} \colon \mathbb{R}^d \to \mathbb{R}^{d'}$. Leveraging the approximation power of mixtures of Gaussian distributions (see, e.g., Scott, 2015), we suppose from now on that $\boldsymbol{X} \in \mathbb{R}^d$ is a multivariate normal random variable.

Using the previous notations—omitting the dependence of M on n—, the s^{th} iteration

$$P_{\hat{\boldsymbol{X}}^{[s]}} = S^{[s]}(P_{\boldsymbol{X}}; n) = \sum_{i=1}^{M} w_i P_{\hat{\boldsymbol{X}}^{[s]},i}$$

of $S^{[1]}$ on $P_{\boldsymbol{X}}$ will play a major role in this work.

The split operator introduces an approximation at each iteration, related to the standard error $\epsilon_{0,1} = W_2(\mathcal{N}(0,1), S^{[1]}(\mathcal{N}(0,1); n)) > 0$.

Let $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}^{[s]}$ the random variables such that $\boldsymbol{Y} = f(\boldsymbol{X})$, $\hat{\boldsymbol{Y}}^{[s]} = f(\hat{\boldsymbol{X}}^{[s]})$ and let $\tilde{\boldsymbol{Y}}^{[s]}$ the random variable obtained by propagating each component of $P_{\hat{\boldsymbol{X}}^{[s]}}$ through $\boldsymbol{f}$ by moment matching.

Finally, $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ is the first order Taylor linearization of $\boldsymbol{f}$ at location $\mu_{\hat{\boldsymbol{X}}^{[s]},i}$ and $\lambda_{\infty}^{[k]}$ is the highest eigenvalue of $\Sigma_{\hat{\boldsymbol{X}}^{[k]},\boldsymbol{i}}$ after applying $S^{[k]}(\cdot, n)$ to $P_{\boldsymbol{X}}$.

D.2.2. UPPER BOUND

Here, we present a proof of the proposition 3.3.

Lemma D.1. *Write Q the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and variance covariance matrix $\Sigma \in \mathcal{S}_d^+(\mathbb{R})$. Write $\hat{Q}$ the mixture of multivariate normal distribution such that $\hat{Q} = S(Q; n) = \sum_{i=1}^n w_i \mathcal{N}(\mu_i, \Sigma_i)$. Write λ_∞ the highest eigenvalue of Σ. Then:*

$$W_2(Q, \hat{Q}) \leq \epsilon_{0,1} \sqrt{\lambda_\infty} \tag{25}$$

Proof of Lemma D.1. Write Q the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and variance covariance matrix $\Sigma \in \mathcal{S}_d^+(\mathbb{R})$. Write $\hat{Q}$ the mixture of multivariate normal distribution such that $\hat{Q} = S(Q; n) = \sum_{i=1}^n w_i \mathcal{N}(\mu_i, \Sigma_i)$.

$\Sigma \in \mathcal{S}_d^+(\mathbb{R}) \implies \exists U \in \mathcal{O}_d^+(\mathbb{R}) \quad \exists \Lambda \in \mathcal{D}_d(\mathbb{R}) : \quad \Sigma = U \Lambda U^T$.

We have $\Lambda = diag(\lambda_1, .., \lambda_d)$ and by definition of the splitting operator, we have $\Sigma_i = U \Lambda_i U^T$, with $\Lambda_i = diag(\lambda_1, .., \tilde{\sigma}^2 \lambda_j, .., \lambda_d)$, where $j = \text{argmax}_k \, \lambda_k$.

$$\begin{aligned}
W_2(Q, \hat{Q}) &= \inf_{\gamma \in \Gamma(Q, \hat{Q})} \mathbb{E}\left[\|X - \hat{X}\|_2^2\right]^{\frac{1}{2}} \\
&= \inf_{\gamma \in \Gamma(Q, \hat{Q})} \mathbb{E}\left[\|UX - U\hat{X}\|_2^2\right]^{\frac{1}{2}} \\
&= \inf_{\gamma \in \Gamma(P, \hat{P})} \mathbb{E}\left[\|X - \hat{X}\|_2^2\right]^{\frac{1}{2}}.
\end{aligned} \tag{26}$$

Since U is orthogonal with P is the multivariate normal distribution $\mathcal{N}(U\mu, \Lambda)$ and $\hat{P} = \sum_{i=1}^n w_i \mathcal{N}(U\mu_i, \Lambda_i)$, the split operator modifies only the marginal of P having the highest eigenvalue (λ_∞), noted P_∞, with mean μ_∞ and variance σ_∞^2.

$$\begin{aligned}
W_2^2(P, \hat{P}) &= \inf_{\gamma \in \Gamma(P, \hat{P})} \mathbb{E}\left[\|X - \hat{X}\|_2^2\right] \\
&= \inf_{\gamma \in \Gamma(P, \hat{P})} \mathbb{E}\left[\sum_{i=1}^d (X_i - \hat{X}_i)^2\right] \\
&= \inf_{\gamma \in \Gamma(P, \hat{P})} \mathbb{E}\left[(X_\infty - \hat{X}_\infty)^2\right] + \mathbb{E}\left[\sum_{i \neq \infty} (X_i - \hat{X}_i)^2\right] \\
&\leq \inf_{\gamma \in \Gamma(P_\infty, \hat{P}_\infty)} \mathbb{E}\left[\|X_\infty - \hat{X}_\infty\|_2^2\right] \\
&= W_2^2(P_\infty, \hat{P}_\infty)
\end{aligned} \tag{27}$$

Because $\Gamma(P_\infty, \hat{P}_\infty) \subset \Gamma(P, \hat{P})$. $\Gamma(P_\infty, \hat{P}_\infty)$ is chosen as the subset of $\Gamma(P, \hat{P})$ restricting to the identity for all dimensions except the one of highest eigenvalue.

Thus:

$$\begin{aligned}
W_2(P, \hat{P}) &\leq W_2(P_\infty, \hat{P}_\infty) \\
&= \inf_{\gamma \in \Gamma(P_\infty, \hat{P}_\infty)} \mathbb{E}\left[\|X_\infty - \hat{X}_\infty\|_2^2\right]^{\frac{1}{2}} \\
&= \sigma_\infty \inf_{\gamma \in \Gamma(P_\infty, \hat{P}_\infty)} \mathbb{E}\left[\|\frac{X_\infty - \mu_\infty}{\sigma_\infty} - \frac{\hat{X}_\infty - \mu_\infty}{\sigma_\infty}\|_2^2\right]^{\frac{1}{2}} \\
&= \sigma_\infty \epsilon_{0,1} \\
&= \sqrt{\lambda_\infty} \epsilon_{0,1}
\end{aligned} \tag{28}$$

□

Lemma D.2. *Write Q the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and variance covariance matrix $\Sigma \in \mathcal{S}_d^+(\mathbb{R})$. Write $\hat{Q}^{[s]}$ the mixture of multivariate normal distribution such that $\hat{Q}^{[s]} = S^{[s]}(Q; n) = \sum_{i=1}^M w_i \mathcal{N}(\mu_i^{[s]}, \Sigma_i^{[s]})$, $s \in \mathbb{N}$. Write $\lambda_\infty^{[k]}$ the highest eigenvalue of $\Sigma^{[k]}$ where $\Sigma^{[k]}$ is the covariance matrix of each component of $\hat{Q}^{[k]}$ obtained after applying k splits. Then:*

$$W_2(Q, \hat{Q}^{[s]}) \leq \epsilon_{0,1} \sum_{k=0}^{s-1} \sqrt{\lambda_\infty^{[k]}} \tag{29}$$

Proof of Lemma D.2. Write Q the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and variance covariance matrix $\Sigma \in \mathcal{S}_d^+(\mathbb{R})$. Write $\hat{Q}^{[s]}$ the mixture of multivariate normal distribution such that $\hat{Q}^{[s]} = S^{[s]}(Q; n) = \sum_{i=1}^M w_i \hat{Q}_i^{[s]} = \sum_{i=1}^M w_i \mathcal{N}(\mu_i^{[s]}, \Sigma_i^{[s]})$, $s \in \mathbb{N}$. Write $\lambda_\infty^{[k]}$ the highest eigenvalue of $\Sigma^{[k]}$ where $\Sigma^{[k]}$ is the covariance matrix of each component of $\hat{Q}^{[k]}$ obtained after applying k splits.

$$\begin{aligned} W_2(Q, \hat{Q}^{[s]}) &= W_2\left(Q, S^{[s]}(Q; n)\right) \\ &\leq \sum_{k=0}^{s-1} W_2\left(S^{[k]}(Q; n), S^{[k+1]}(Q; n)\right) \end{aligned} \tag{30}$$

With $S^{[0]}(Q; n) = Q$

And:

$$\begin{aligned} W_2\left(S^{[k]}(Q; n), S^{[k+1]}(Q; n)\right) &\leq \sum_{i=1}^M w_i W_2\left(\hat{Q}_i^{[k]}, S(\hat{Q}_i^{[k]}; n)\right) \\ &\leq \sum_{i=1}^M w_i W_2\left(\mathcal{N}(\mu_i^{[k]}, \Sigma_i^{[k]}), S(\mathcal{N}(\mu_i^{[k]}, \Sigma_i^{[k]}); n)\right) \end{aligned} \tag{31}$$

But for a same level of split k, all components of the mixture have the same covariance matrix, i.e. $\forall i \in \{1...M\} : \quad \Sigma_i^{[k]} = \Sigma_0^{[k]}$. Thus, with $\sum_i w_i = 1$, by noting $\lambda_\infty^{[k]}$ the highest eigenvalue of $\Sigma_0^{[k]}$, and by applying Lemma D.1, we obtain:

$$W_2(Q, \hat{Q}^{[s]}) \leq \epsilon_{0,1} \sum_{k=0}^{s-1} \sqrt{\lambda_\infty^{[k]}} \tag{32}$$

□

Proof of Proposition 3.3. Let $W_2(\cdot, \cdot)$ the 2-Wasserstein distance with respect to the usual Euclidean norm on $\mathbb{R}^d$. With the general notations, we have:

$$W_2(P_{\boldsymbol{Y}}, P_{\tilde{\boldsymbol{Y}}^{[s]}}) \leq W_2(P_{\boldsymbol{Y}}, P_{\hat{\boldsymbol{Y}}^{[s]}}) + W_2(P_{\hat{\boldsymbol{Y}}^{[s]}}, P_{\tilde{\boldsymbol{Y}}^{[s]}}) \tag{33}$$

With:

$$\begin{aligned} W_2(P_{\boldsymbol{Y}}, P_{\hat{\boldsymbol{Y}}^{[s]}}) &= \inf_{\gamma \in \Gamma(P_{\boldsymbol{Y}}, P_{\hat{\boldsymbol{Y}}^{[s]}})} \mathbb{E}\left[\|Y - \hat{Y}\|_2^2\right]^{\frac{1}{2}} \\ &\leq \inf_{\gamma \in \Gamma(P_{\boldsymbol{X}}, P_{\hat{\boldsymbol{X}}^{[s]}})} \mathbb{E}\left[\|\boldsymbol{f}(X) - \boldsymbol{f}(\hat{X})\|_2^2\right]^{\frac{1}{2}} \\ &\leq L_{2,2} \inf_{\gamma \in \Gamma(P_{\boldsymbol{X}}, P_{\hat{\boldsymbol{X}}^{[s]}})} \mathbb{E}\left[\|X - \hat{X}\|_2^2\right]^{\frac{1}{2}} \\ &\leq L_{2,2} W_2(P_{\boldsymbol{X}}, P_{\hat{\boldsymbol{X}}^{[s]}}) \\ &\leq L_{2,2}\epsilon_{0,1} \sum_{k=0}^{s-1} \sqrt{\lambda_\infty^{[k]}} \end{aligned} \tag{34}$$

By applying Lemma D.2.

And:

$$\begin{aligned} W_2\left(P_{\hat{\boldsymbol{Y}}^{[s]}}, P_{\tilde{\boldsymbol{Y}}^{[s]}}\right) &\leq \sum_{i=1}^{M} w_i W_2\left(P_{\hat{\boldsymbol{Y}}^{[s]},i}, P_{\tilde{\boldsymbol{Y}}^{[s]},i}\right) \\ &\leq \sum_{i=1}^{M} w_i \left(W_2(P_{\hat{\boldsymbol{Y}}^{[s]},i}, P_{\bar{\boldsymbol{Y}}^{[s]},i}) + W_2(P_{\bar{\boldsymbol{Y}}^{[s]},i}, P_{\tilde{\boldsymbol{Y}}^{[s]},i})\right) \end{aligned} \tag{35}$$

With:

$$W_2(P_{\bar{\boldsymbol{Y}}^{[s]},i}, P_{\tilde{\boldsymbol{Y}}^{[s]},i}) = \left(\|\mu_{\bar{\boldsymbol{Y}}^{[s]},i} - \mu_{\tilde{\boldsymbol{Y}}^{[s]},i}\|_2^2 + Tr(\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} + \Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i} - 2\left(\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}^{\frac{1}{2}} \Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} \Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}^{\frac{1}{2}}\right)^{\frac{1}{2}}\right)\right)^{\frac{1}{2}} \tag{36}$$

And:

$$\begin{aligned} W_2^2(P_{\hat{\boldsymbol{Y}}^{[s]},i}, P_{\bar{\boldsymbol{Y}}^{[s]},i}) &= \inf_{\gamma \in \Gamma(P_{\hat{\boldsymbol{Y}}^{[s]},i}, P_{\bar{\boldsymbol{Y}}^{[s]},i})} \mathbb{E}\left[\|\hat{\boldsymbol{Y}}^{[s]} - \bar{\boldsymbol{Y}}^{[s]}\|_2^2\right] \\ &\leq \mathbb{E}_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}}\left[\|\boldsymbol{f}(\boldsymbol{X}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X})\|_2^2\right] \end{aligned} \tag{37}$$

By choosing γ entirely defined by $\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}$ and such that $(\hat{\boldsymbol{Y}}^{[s]}, \bar{\boldsymbol{Y}}^{[s]}) = (\boldsymbol{f}(\boldsymbol{X}), \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X}))$.

This gives the results.

□

D.3. Upper bound convergence

We present here the proof of the proposition 3.4.

Lemma D.3. *Let $x = [x_1, ..., x_d]$ denotes a vector in $\mathbb{R}_+^d$. Let g denotes the function such that: $g(x) = [x_1, ..., \tilde{\sigma}^2 x_j, ..., x_d]$, with $j = \text{argmax}_i\, x_i$ and $0 \leq \tilde{\sigma} \leq 1$, and $g^{[k]}$ the concatenation of k times g. Without loss of generality, we suppose that $\forall i \in \{1...d-1\} \quad x_i > x_d$. Then:*

$$\sum_{k=1}^{s} \sqrt{\|g^{[k]}(x)\|_\infty} \underset{s \to \infty}{\longrightarrow} \left(C + \frac{d\sqrt{x_d}}{1 - \tilde{\sigma}}\right) \tag{38}$$

where $C = \sum_{k=0}^{r_d} \sqrt{\|g^{[k]}\|_\infty}$, and $r_d = \sum_{i=1}^{d-1} \left\lceil \frac{\ln x_i - \ln x_d}{\ln \tilde{\sigma}^2} \right\rceil$

Proof of Lemma D.3. We note $x^{[k]} = g^{[k]}(x)$. For all $i \in \{1...d-1\}$, let N_i denotes the minimum integer such that $(\tilde{\sigma}^2)^{N_i} x_i < x_d$.

Trivially, $N_i = \left\lceil \frac{\ln x_i - \ln x_d}{\ln \tilde{\sigma}^2} \right\rceil$ Thus after applying $r_d = \sum_{i=1}^{d-1} N_i = \sum_{i=1}^{d-1} \left\lceil \frac{\ln x_i - \ln x_d}{\ln \tilde{\sigma}^2} \right\rceil$ times g over x, we have $\forall i \in \{1...d-1\} \quad \tilde{\sigma}^2 x_d \leq x_i^{[r_d]} \leq x_d$ (the left inequality obtained by reasoning by absurd).

More generally: $\forall m \in \mathbb{N} \quad \forall i \in \{1...d-1\} \quad (\tilde{\sigma}^2)^{m+1} x_d \leq x_i^{[r_d+md]} \leq (\tilde{\sigma}^2)^m x_d$ (by recurrence).

And: $\forall m \in \mathbb{N} \quad \forall i \in \{1...d-1\} \quad \forall p \in \{0...d-1\} \quad x_i^{[r_d+md+p]} \leq x_i^{[r_d+md]}$

$$\begin{aligned}
\sum_{k=0}^{\infty} \sqrt{\|g^{[k]}(x)\|_\infty} &= \sum_{k=0}^{r_d} \sqrt{\|g^{[k]}(x)\|_\infty} + \sum_{k=0}^{\infty} \sum_{p=0}^{d-1} \sqrt{\|g^{[r_d+kd+p]}(x)\|_\infty} \\
&\leq \sum_{k=0}^{r_d} \sqrt{\|g^{[k]}(x)\|_\infty} + d \sum_{k=0}^{\infty} \sqrt{\|g^{[r_d+kd]}(x)\|_\infty} \\
&\leq \sum_{k=0}^{r_d} \sqrt{\|g^{[k]}(x)\|_\infty} + d \sum_{k=0}^{\infty} \sqrt{(\tilde{\sigma}^2)^k x_d} \\
&\leq \sum_{k=0}^{r_d} \sqrt{\|g^{[k]}(x)\|_\infty} + \frac{d\sqrt{X_d}}{1-\tilde{\sigma}} \qquad (39)
\end{aligned}$$

finishing the proof. □

Proof of Proposition 3.4. Let us note $V_s = \sum_i w_i E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(\boldsymbol{X}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X})\|_2^2 \right]^{\frac{1}{2}} + \left(\|\mu_{\tilde{\boldsymbol{Y}}^{[s]},i} - \mu_{\bar{\boldsymbol{Y}}^{[s]},i}\|_2^2 + \right.$
$\left. \mathrm{Tr}\left(\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} + \Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i} - 2\left(\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}^{\frac{1}{2}} \Sigma_{\bar{\boldsymbol{Y}}^{[s]},i} \Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}^{\frac{1}{2}} \right)^{\frac{1}{2}} \right) \right)^{\frac{1}{2}}$

In the following, we show that $V_s \underset{s \to \infty}{\longrightarrow} 0$.

$$\begin{aligned}
\|\mu_{\tilde{\boldsymbol{Y}}^{[s]},i} - \mu_{\bar{\boldsymbol{Y}}^{[s]},i}\|_2^2 &= \|E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\boldsymbol{f}(\boldsymbol{X}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X}) \right]\|_2^2 \\
&\leq E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(\boldsymbol{X}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X})\|_2^2 \right] \qquad (40)
\end{aligned}$$

Where $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ is the first order Taylor linearization of $\boldsymbol{f}$ at location $\mu_{\hat{\boldsymbol{X}}^{[s]},i}$. With $J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})$ the Jacobian of $\boldsymbol{f}$ at location $\mu_{\hat{\boldsymbol{X}}^{[s]},i}$, we have $\bar{\boldsymbol{f}}_{\boldsymbol{i}}(X) = \boldsymbol{f}(\mu_{\hat{\boldsymbol{X}}^{[s]},i}) + J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})(\boldsymbol{X} - \mu_{\hat{\boldsymbol{X}}^{[s]},i})$. Please refer to Remark D.4 for a discussion about $J_{\boldsymbol{f}}(\cdot)$. Thus:

$$\begin{aligned}
E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(\boldsymbol{X}) - \bar{\boldsymbol{f}}_{\boldsymbol{i}}(\boldsymbol{X})\|_2^2 \right] &= E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{f}(\mu_{\hat{\boldsymbol{X}}^{[s]},i}) - J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})(\boldsymbol{X} - \mu_{\hat{\boldsymbol{X}}^{[s]},i})\|_2^2 \right] \\
&\leq 2 \left(E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{f}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})\|_2^2 \right] \right. \\
&+ \left. E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})(\boldsymbol{X} - \mu_{\hat{\boldsymbol{X}}^{[s]},i})\|_2^2 \right] \right) \\
&\leq 2(L_{2,2}^2 + |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2) E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\|\boldsymbol{X} - \mu_{\hat{\boldsymbol{X}}^{[s]},i}\|_2^2 \right] \\
&\leq 2(L_{2,2}^2 + |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2) E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[\sum_{k=1}^{d} (\boldsymbol{X}_k - \mu_{\hat{\boldsymbol{X}}^{[s]},i,k})^2 \right] \\
&\leq 2(L_{2,2}^2 + |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2) \sum_{k=1}^{d} E_{\boldsymbol{X} \sim P_{\hat{\boldsymbol{X}}^{[s]},i}} \left[(\boldsymbol{X}_k - \mu_{\hat{\boldsymbol{X}}^{[s]},i,k})^2 \right] \\
&\leq 2(L_{2,2}^2 + |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2) \mathrm{Tr}(\Sigma_{\hat{\boldsymbol{X}}^{[s]},i}) \qquad (41)
\end{aligned}$$

where $|||\cdot|||$ is the matrix norm, and Tr is the matrix trace.

$$\mathrm{Tr}\left(\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i}+\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}-2\left(\Sigma^{\frac{1}{2}}_{\tilde{\boldsymbol{Y}}^{[s]},i}\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i}\Sigma^{\frac{1}{2}}_{\tilde{\boldsymbol{Y}}^{[s]},i}\right)^{\frac{1}{2}}\right) \quad \leq \quad \mathrm{Tr}\left(\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i}+\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}\right) \tag{42}$$

And:

$$\begin{aligned}
\mathrm{Tr}\left(\Sigma_{\tilde{\boldsymbol{Y}}^{[s]},i}\right) &= \mathrm{Tr}\left(\mathrm{Cov}(\tilde{Y}^{[s],i})\right)\\
&= \mathrm{Tr}\left(\mathrm{Cov}\left(\boldsymbol{f}(\hat{X}^{[s],i})\right)\right)\\
&= \mathrm{Tr}\left(\mathrm{Cov}\left(\boldsymbol{f}(\hat{X}^{[s],i})-E_{X'}[\boldsymbol{f}(X')]\right)\right)\\
&\leq E_{\hat{X}^{[s],i}}[\|\boldsymbol{f}(\hat{X}^{[s],i})-E_{X'}[\boldsymbol{f}(X')]\|_2^2]\\
&= E_{\hat{X}^{[s],i}}[\|E_{X'}[\boldsymbol{f}(\hat{X}^{[s],i})-\boldsymbol{f}(X')]\|_2^2]\\
&= L_{2,2}^2 E_{\hat{X}^{[s],i}}[\|E_{X'}[\hat{X}^{[s],i}-X']\|_2^2]\\
&\leq L_{2,2}^2 E_{\hat{X}^{[s],i}}[E_{X'}[\|\hat{X}^{[s],i}-X'\|_2^2]]\\
&= L_{2,2}^2 E_{\hat{X}^{[s],i}}[E_{X'}[\|(\hat{X}^{[s],i}-E_{\hat{X}^{[s],i}}[\hat{X}^{[s],i}])-(X'-E_{X'}[X'])\|_2^2]]\\
&\leq 2L_{2,2}^2(E_{\hat{X}^{[s],i}}[\|\hat{X}^{[s],i}-E_{\hat{X}^{[s],i}}[\hat{X}^{[s],i}]\|_2^2]+E_{X'}[\|X'-E_{X'}[X']\|_2^2])\\
&= 4L_{2,2}^2 E_{\hat{X}^{[s],i}}[\|\hat{X}^{[s],i}-E_{\hat{X}^{[s],i}}[\hat{X}^{[s],i}]\|_2^2]\\
&= 4L_{2,2}^2\mathrm{Tr}\left(\mathrm{Cov}\left(\hat{X}^{[s],i}-E_{\hat{X}^{[s],i}}[\hat{X}^{[s],i}]\right)\right)\\
&= 4L_{2,2}^2\mathrm{Tr}\left(\Sigma_{\hat{\boldsymbol{X}}^{[s]},i}\right)
\end{aligned} \tag{43}$$

Where X' is an i.i.d copy of $\hat{X}^{[s],i}$.

Finally:

$$\begin{aligned}
\mathrm{Tr}\left(\Sigma_{\bar{\boldsymbol{Y}}^{[s]},i}\right) &= \mathrm{Tr}\left(\mathrm{Cov}\left(\bar{Y}^{[s],i}\right)\right)\\
&= \mathrm{Tr}\left(\mathrm{Cov}\left(\bar{\boldsymbol{f}}(\hat{X}^{[s],i})\right)\right)\\
&= \mathrm{Tr}\left(\mathrm{Cov}\left(\boldsymbol{f}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})+J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})(\hat{X}^{[s],i}-\mu_{\hat{\boldsymbol{X}}^{[s]},i})\right)\right)\\
&= \mathrm{Tr}\left(\mathrm{Cov}\left(J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})(\hat{X}^{[s],i}-\mu_{\hat{\boldsymbol{X}}^{[s]},i})\right)\right)\\
&\leq E_{\hat{X}^{[s],i}}[|||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})(\hat{X}^{[s],i}-\mu_{\hat{\boldsymbol{X}}^{[s]},i})\|_2^2]\\
&\leq |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2 E_{\hat{X}^{[s],i}}[\|\hat{X}^{[s],i}-\mu_{\hat{\boldsymbol{X}}^{[s]},i}\|_2^2]\\
&= |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2\mathrm{Tr}\left(\mathrm{Cov}\left(\hat{X}^{[s],i}-E_{\hat{X}^{[s],i}}[\hat{X}^{[s],i}]\right)\right)\\
&= |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||^2\mathrm{Tr}\left(\mathrm{Cov}\left(\Sigma_{\hat{\boldsymbol{X}}^{[s]},i}\right)\right)
\end{aligned} \tag{44}$$

And $\forall i \in \{1...M\}:\quad \Sigma_{\hat{\boldsymbol{X}}^{[s]},i}=\Sigma_{\hat{\boldsymbol{X}}^{[s]},0}$, noting $J_{[s]}=\max_i |||J_{\boldsymbol{f}}(\mu_{\hat{\boldsymbol{X}}^{[s]},i})|||$ and having $\sum_i \omega_i = 1$ conducts:

$$V_s \leq \sqrt{2\left(L_{2,2}^2+J_{[s]}^2\right)\mathrm{Tr}\left(\Sigma_{\hat{\boldsymbol{X}}^{[s]},0}\right)}+\sqrt{\left(6L_{2,2}^2+3J_{[s]}^2\right)\mathrm{Tr}\left(\Sigma_{\hat{\boldsymbol{X}}^{[s]},0}\right)} \tag{45}$$

$$V_s \underset{s\to\infty}{\longrightarrow} 0 \tag{46}$$

Because $\Sigma_{\hat{\boldsymbol{X}}^{[s]},i} \underset{s\to\infty}{\longrightarrow} 0$ and $J_{[s]} \leq L_{2,2}$ by the Lipschitz assumption on $\boldsymbol{f}$.

Applying Lemma D.3 gives us the requested result. □

Remark D.4. The proof of Proposition **??** introduces the Jacobian matrix of $\boldsymbol{f}$. It assumes that f is C^1 so that $J_{\boldsymbol{f}}(x)$ is the Jacobian matrix of f at $x \in \mathbb{R}^d$. The proof only requires $\bar{\boldsymbol{f}}_{\boldsymbol{i}}$ to be a linear mapping of the form $\boldsymbol{x} \mapsto \boldsymbol{f}(\mu_{\hat{\boldsymbol{X}}^{[s]},i}) + A(\boldsymbol{x} - \mu_{\hat{\boldsymbol{X}}^{[s]},i})$, with a bound on $|||A|||$. This is the case for gradients computed by standard backpropagation algorithms with Lipschitz activation functions such as ReLU or Leaky ReLU.

E. Splitting library

We recall that the splitting of a multivariate Gaussian distribution consists in splitting the distribution in a specific direction, following the formula:

$$w_i = w\tilde{w}_i, \quad \boldsymbol{\mu_i} = \boldsymbol{\mu} + \tilde{\mu}_i\sqrt{\lambda_j}\boldsymbol{\nu_j}, \quad \tilde{\boldsymbol{\Lambda}} = diag(\lambda_1, \ldots, \tilde{\sigma}^2\lambda_j, \ldots, \lambda_k) \tag{47}$$

where j is the splitting direction, i.e. the axis along which the Gaussian mixture PDF terms are split, $\tilde{w}_i$, $\tilde{\mu}_i$ and $\tilde{\sigma}$ are specified by Table 8.

Table 8. Splitting library for the standard univariate Gaussian distribution.

# components	$\tilde{w}_i$	$\tilde{\beta}_i$	$\tilde{\sigma}$	$\epsilon_{0,1}(\times 10^{-5})$
3	0.6364	0.0	0.005	5.57
	0.1818	+1.0579		
	0.1818	-1.0579		
5	0.4444	0.0	0.008	4.42
	0.2455	+0.9332		
	0.2455	-0.9332		
	0.0323	+1.9776		
	0.0323	-1.9776		
7	0.3048	0.0	0.019	4.18
	0.2410	+0.7056		
	0.2410	-0.7056		
	0.0948	+1.4992		
	0.0948	-1.4992		
	0.0118	+2.4601		
	0.0118	-2.4601		

This library is reused from (Zhang & Shin, 2021) (apart from $\epsilon_{0,1}$ values) but originally appears in (DeMars et al., 2013). These parameters can be obtained by resolving the minimization problem presented in Section 3.4.1 using methods such as fmincon from Matlab (Mat, 2017).

F. Merge criterion

For a layer-wise approach, we provide here a Wasserstein based criterion. Let's note $p(\boldsymbol{Y}) = \sum_{k=1}^{M} w_k p_k(\boldsymbol{Y})$, with $p_k(\boldsymbol{Y}) = \mathcal{N}(\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$, a Gaussian mixture distribution and $p^{(i,j)}(\boldsymbol{Y}) = \sum_{k=1}^{M-1} w_k p_k^{(i,j)}(\boldsymbol{Y})$ with $p_k^{(i,j)}(\boldsymbol{Y}) = \mathcal{N}(\boldsymbol{\mu_k^{(i,j)}}, \boldsymbol{\Sigma_k^{(i,j)}})$ the resulting mixture after merging the Gaussian i and j of $p(\boldsymbol{Y})$. Without loss of generality, we can assume that $(i,j) = (M-1, M)$:

$$\begin{aligned} C_{M-1,M}^{merge} &= W_2^2(p, p^{(M-1,M)}) \\ &\leq w_{M-1}W_2^2(p_{M-1}, p_{M-1}^{(M-1,M)}) + w_M W_2^2(p_M, p_{M-1}^{(M-1,M)}) \end{aligned} \tag{48}$$

G. Societal impacts

More generally, uncertainty quantification consists in quantifying a doubt over a prediction so that one can assess its reliability. Our work proposes to do so for deep learning algorithms where classic uncertainty quantification techniques can not be applied.

As such, our work and research have no direct negative societal impacts. On the contrary, quantifying uncertainty for complex systems based on deep learning-based technological solutions is critical. Indeed, it allows control systems to increase the security of such solutions in the case of a decision with an elevated associated doubt.

However, quantifying uncertainty in deep learning algorithms tends to facilitate the deployment of these solutions, which present negative social impacts inherent to this kind of technology. Among these impacts, we mainly find issues concerning privacy, security (Bae et al., 2018), bias and discrimination (Mehrabi et al., 2021).

H. Libraries

Table 9 presents the library used in this project as well as their version and license.

Table 9. Details of library used in this project

Library	Version	Build	License
matplotlib (Hunter, 2007)	3.5.1	$py37haa95532\_1$	BSD
numpy (Harris et al., 2020)	1.211.5	$py37h7a0a035\_2$	BSD 3
pot (Flamary et al., 2021)	0.8.2	$py37h3182a2c\_0$	MIT
scikit-image (Van der Walt et al., 2014)	0.19.2	$py37hf11a4ad\_0$	BSD 3
scikit-learn (Pedregosa et al., 2011)	1.0.2	$py37hf11a4ad\_1$	BSD 3
scipy (Virtanen et al., 2020)	1.7.3	$py37h0a974cb\_0$	BSD
seaborn (Waskom, 2021)	0.11.2	$pyhd3eb1b0\_0$	BSD
tensorflow-gpu (Agrawal et al., 2019)	2.1.0	$h0d30ee6\_0$	Apache 2.0
tensorflow-probability (Agrawal et al., 2019)	0.8.0	py_0	Apache 2.0