Enabling Uncertainty Estimation in Iterative Neural Networks

Nikita Durasov [1] **Doruk Oner** [1] **Jonathan Donier** [2] **Hieu Le** [1] **Pascal Fua** [1]

Abstract

Turning pass-through network architectures into iterative ones, which use their own output as input, is a well-known approach for boosting performance. In this paper, we argue that such architectures offer an additional benefit: The convergence rate of their successive outputs is highly correlated with the accuracy of the value to which they converge. Thus, we can use the convergence rate as a useful proxy for uncertainty. This results in an approach to uncertainty estimation that provides state-of-the-art estimates at a much lower computational cost than techniques like Ensembles, and without requiring any modifications to the original iterative model. We demonstrate its practical value by embedding it in two application domains: road detection in aerial images and the estimation of aerodynamic properties of 2D and 3D shapes.

poster / code / web

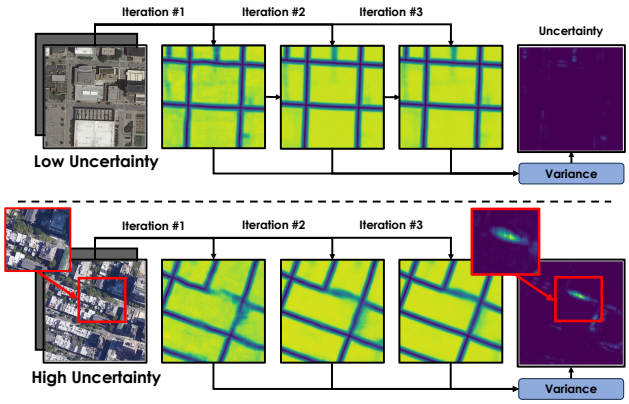


Figure 1. **Uncertainty in recursive models.** Such models use their initial predictions as inputs to produce subsequent predictions. We display the output of three consecutive iterations of a network trained to compute distance maps to road pixels. **(Top:)** All roads are clearly visible. The three maps are similar and the per pixel variance is low. **(Bottom:)** The road in the red square is tree-covered. It is eventually detected properly but the variance is high.

1. Introduction

It has long been known that using deep networks to recursively refine predictions is often beneficial. This has been demonstrated for semantic segmentation (Zhou et al., 2018; Wang et al., 2019), pose estimation (Newell et al., 2016), depth estimation (Zhang et al., 2018a), multi-task learning (Xu et al., 2018), delineation (Mosińska et al., 2018), natural language processing (A.Vaswani et al., 2017; Devlin et al., 2018), among others. Given a network f parameterized by weights Θ and takes as input a vector $\mathbf{x}$, which can represent an image or a text, and produces an output $\mathbf{y}$, the output of the recursion's i^{th} iteration can be written as $\mathbf{y}_i = f_\Theta(\mathbf{x}, \mathbf{y}_{i-1})$, where $\mathbf{y}_{i-1}$ is the output of the previous iteration. This recursion often yields improved performance over non-iterative methods using the same network architectures (Shen et al., 2017; Zhang et al., 2018b; Wang et al., 2019; Oner et al., 2021; 2022), while requiring less labeled data for training purposes (Mosińska et al., 2018). In essence, giving a previous output as input to the network sets up a virtuous circle in which the model receives relevant spatial attention signals that serve as priors and help generate improved predictions. For example, in the road delineation example of Fig. 1, the presence of road fragments with gaps in them cues the network to the possible existence of connecting segments. These ideas been explored before the advent of Deep Learning, for example using Tensor Voting (Medioni et al., 2000), but incorporating them into deep networks has given them a new lease on life.

The key insight of this paper is that how fast this refinement occurs is closely connected to the accuracy of the prediction. A hard sample typically requires more refinement iterations than an easy one. Thus, convergence speed can be used as a proxy for certainty. This yields an approach to uncertainty estimation that is on par with *Deep Ensembles* (Lakshminarayanan et al., 2017), while delivering increased accuracy at a much lower computational cost and without requiring any modifications to the original iterative model. This makes our approach practical and easy to deploy across many dif-

[1]Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland [2]Neural Concept SA, Lausanne, Switzerland. Correspondence to: Nikita Durasov <nikita.durasov@epfl.ch>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ferent applications. This is significant because Ensembles is often considered to be one of the very best uncertainty estimation methods whose only severe drawback is its high computational cost.

More specifically, we derive uncertainty measures by analyzing the variance in outputs from consecutive iterations of an iterative model, where higher variance indicates greater uncertainty. We demonstrate that this is fast, accurate, and easy to deploy in two very different application domains, road detection in aerial images and the estimation of aerodynamic properties of 2D and 3D shapes. These two applications feature unique sets of challenges, which underscore the versatility and broad applicability of our approach.

Our contributions are as follows:

- We introduce an effective method for estimating uncertainty in iterative models. It relies on consistency across consecutive predictions and does not require modifying the network architectures.
- We provide extensive experiments and analyses to demonstrate the correctness and effectiveness of the proposed method. In particular, we show that going from a toy example to our two complex real-world scenarios, there remains a consistent correlation between convergence speed and prediction accuracy.
- Our method, once embedded in a Bayesian Optimization framework, delivers state-of-the-art accuracy and predictive uncertainty quantification in both road detection from aerial images and 2D/3D shape optimization.

The code will be made publicly available.

2. Related Work

2.1. Uncertainty Estimation

Uncertainty Estimation (UE) aims at accurately evaluating the reliability of a model's predictions. Deep Ensembles (Lakshminarayanan et al., 2017), MC-Dropout (Gal & Ghahramani, 2016), and Bayesian Networks (Mackay, 1995) have emerged as the most influential approaches.

Deep Ensembles involve training multiple networks, starting from different initial conditions. They are noted for the potential diversity of their predictions, attributable to randomness from weight initialization, data augmentation, and stochastic gradient updates. This diversity is central to their effectiveness (Perrone & Cooper, 1995; Fort et al., 2019). In many situations, they deliver more reliable uncertainty estimates than other methods (Ovadia et al., 2019; Gustafsson et al., 2020; Ashukha et al., 2020; Postels et al., 2022). They therefore remain the leading technique, despite the high computational cost of training several networks instead of a single one and of performing several forward passes at inference time. One of the active research directions is reducing the training and inference time of ensembling methods, as well as their memory requirements. For example, Antorán et al. (2020) try to emulate ensemble predictions by producing several outputs based on features at different levels of the model, and Daxberger et al. (2021) perform Bayesian inference only on a subset of the model's weights chosen through a pruning-like procedure.

Among the other techniques, *MC-Dropout* involves randomly zeroing out network weights and assessing the effect and is popular due to its lower computational cost. Unfortunately, its estimates remain less reliable than those of *Deep Ensembles* (Ashukha et al., 2020), even though there has been recent attempts at improving it (Wen et al., 2020; Durasov et al., 2021). Similarly, *Bayesian Networks* rarely outperform *Deep Ensembles* (Blundell et al., 2015; Graves, 2011; Hernández-Lobato & Adams, 2015; Kingma et al., 2015).

All the above methods are sampling-based and require several forwards passes at inference time. Thus, when a fast response time is required, as in robotics control, sampling-free approaches with single-pass inference become of interest. However, deploying them often requires significant modifications to the network's architecture (Postels et al., 2019), substantial changes to the training procedures (Malinin & Gales, 2018), limiting their application to very specific tasks (Amersfoort et al., 2020; Malinin & Gales, 2018; Mukhoti et al., 2021), or reducing the quality of the uncertainty estimate (Postels et al., 2022; Ashukha et al., 2020). As a result, they have not gained as much traction as MC-Dropout and Ensembles.

2.2. Iterative Refinement Methods

Iterative refinement techniques have been for many different purposes (Mnih & Hinton, 2010; Pinheiro & Collobert, 2014; Tu & Bai, 2009; Shen et al., 2017). This incorporates surrounding context into the prediction (Seyedhosseini et al., 2013), proving particularly useful for tasks such delineation (Sironi et al., 2016), human pose estimation (Newell et al., 2016), semantic segmentation (Zhang et al., 2018b; Wang et al., 2019), depth estimation (Durasov et al., 2019; Xu et al., 2018), and multi-task learning (Durasov et al., 2022b). In the first use case of this work, we build upon the recursive networks used in (Mosińska et al., 2018) to delineate roads by computing distance maps to the road pixels, as shown in Fig. 1. The network is a UNet (Ronneberger et al., 2015) that takes as input the image and the distance map computed at the previous iteration, starting from a blank one. However, whereas the typical focus of delineation papers is to increase performance in terms of a number of delineation metrics, ours is to provide an uncertainty estimate on the detections

without sacrificing performance. In the second use case, we use an iterative model with Graph Neural Networks (Monti et al., 2017; Baqué et al., 2018) for 2D and 3D shape optimization. We show that uncertainty measures can be used to effectively select out-of-distribution data to enhance the training dataset.

3. Method

Let us consider a *recursive* network f_Θ, where Θ are the network weights. f_Θ takes as input a vector $\mathbf{x}$—an image or a 3D shape in the examples presented in the results section—and its own output $\mathbf{y}$—a segmentation image or a pressure field in our examples. At the i^{th} iteration, we have

$$\mathbf{y}_i = f_\Theta(\mathbf{x}, \mathbf{y}_{i-1}) \ , \tag{1}$$

where the initial value $\mathbf{y}_0$ can be taken to be a vector of zeros. This computation is repeated N times and $\mathbf{y}_N$ is taken to be the final output. In supervised approaches, the network is trained so that $\mathbf{y}_N$ matches the ground truth, with (Newell et al., 2016; Carreira et al., 2016) or without (Chen et al., 2018; Gupta & Chandraker, 2020) supervision on the intermediate outputs $\{\mathbf{y}_1, \dots \mathbf{y}_{N-1}\}$. Our key insight is that, at inference time, rather than treating $\mathbf{y}_N$ as the sole output, as is usually done, we should consider the whole sequence $\mathbf{Y} = \{\mathbf{y}_1, \dots \mathbf{y}_N\}$ because it provides valuable information about prediction certainty.

In the remainder of this section, we first discuss the behavior of this series of estimates and then propose a simple algorithm that exploits it to estimate uncertainty.

3.1. Motivation

Each iteration of Eq.1 takes the current prediction $\mathbf{y}_{i-1}$ and refines it into $\mathbf{y}_i$. This resembles what a denoising auto-encoder does when mapping a noisy input signal to its true value. Hence, the theoretical understanding of reconstruction errors in auto-encoders is relevant to our problem.

In fact, the reconstruction error for a sample fed into an autoencoder can indicate whether the sample lies within the training distribution of the model or not (Japkowicz et al., 1995). More formally, given a denoising or contractive autoencoder, $\mathcal{R}$, and a sample $\mathbf{x}$, the reconstruction error $|\mathcal{R}(\mathbf{x}) - \mathbf{x}|$ is closely related to the log-probability of the data distribution $p_{data}(\mathbf{x})$ (Bengio et al., 2013; Alain & Bengio, 2014). This understanding was later expanded to include a broader spectrum of autoencoders (Kamyshanska & Memisevic, 2013), and then to standard autoencoders trained under stochastic optimization (Solinas et al., 2020). This theoretical work has been effectively applied to many practical tasks requiring out-of-distribution detection (Zhou, 2022; Sabokrou et al., 2016).

Thus, as in Bengio et al. (2013); Alain & Bengio (2014), we can rewrite the update equation of Eq. 1 as

$$\mathbf{y}_{i+1} - \mathbf{y}_i = f_\Theta(\mathbf{x}, \mathbf{y}_i) - \mathbf{y}_i \propto \frac{\partial \log p(\mathbf{y}_i|\mathbf{x})}{\partial \mathbf{y}_i} \ , \tag{2}$$

where $p(\mathbf{y}_i|\mathbf{x})$ is the probability of the model yielding the prediction $\mathbf{y}_i$ given the input $\mathbf{x}$. In other words, the recursion can be understood as a gradient ascent on $\log p(\mathbf{y}_i|\mathbf{x})$, which explains why the prediction progressively improves as illustrated by Fig. 2 in a simple regression case. We can distinguish three different scenarios

- **In distribution, without aleatoric noise.** $\mathbf{x}$ is in-distribution, there is little noise in the training data, which makes the *aleatoric uncertainty* low and the probability p peaked. So, if $\mathbf{y}_i$ is already close to being correct, the derivative of the log probability will be small and it will not move much. In contrast, it is not correct but still within the main mode of the probability distribution, the derivative will be large and the convergence rapid. This is the case $x = 2.0$.
- **In distribution, with aleatoric noise.** $\mathbf{x}$ is in-distribution, the training data is noisy, making the he *aleatoric uncertainty* higher and the probability p less peaked. If $\mathbf{y}_i$ is not initially correct but still within the main mode of the probability distribution, the derivative will be smaller than in the no-noise scenario, and thus the convergence slower, as in the case of $x = 5.0$.
- **Out of distribution.** $\mathbf{x}$ is out-of-distribution, which can be understood as *epistemic uncertainty* and p cannot be expected to have a well-defined peaked shape but often tends to be flatter. The behavior is then somewhat random as in the case of $x = -2.0$ and $x = 7.0$.

In short, both aleatoric and epistemic uncertainty are likely to impact convergence speed negatively.

3.2. Estimating Uncertainty

In Section 3.1, we have argued that when the input $\mathbf{x}$ to network f_Θ is in-domain with respect to the sample distribution that was used to train the network and the aleatoric uncertainty is low, we can expect the convergence of sequence $\mathbf{Y}$ tends to be quick. In contrast, when $\mathbf{x}$ is out-of-domain or the aleatoric uncertainty is high, we can expect it to be far more erratic. To exploit this insight, for a given $\mathbf{x}$ t we take the variance of sequence $\mathbf{Y}$ as a proxy for uncertainty. We take it to be

$$U^i = Var(\{\mathbf{y}_1^i, \mathbf{y}_2^i, \cdots, \mathbf{y}_N^i\}) \ , \tag{3}$$

where i corresponds to the i^{th} pixel/node in the original input $\mathbf{x}$ and N is the number of iterations passing through the iterative network. Here the variance represents the convergence speed, where higher values imply slower convergence

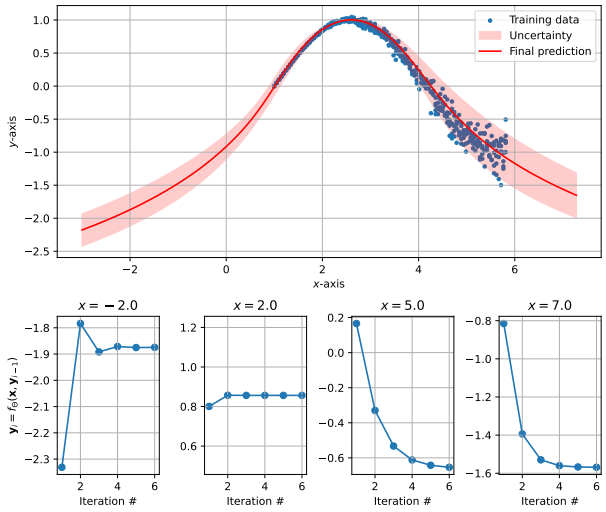


Figure 2. **Uncertainty vs Convergence.** In this example, we generated training data from a sinusoidal function for $x \in [1, 6]$ and added Gaussian noise with a variance that increases from left to right. We take f_Θ to be a simple MLP with three hidden layers that takes two inputs, x and the output of the previous iteration. We train it to predict the noisy data points at each step of the iterative process by minimizing the loss of Eq. 4. Once trained, we use f_Θ to produce predictions $\mathbf{Y}(x) = \{\mathbf{y}_1(x), \dots \mathbf{y}_N(x)\}$ for $x \in [-3, 7]$ **(Top):** The red line denotes the final prediction $\mathbf{y}_N(x)$, and the standard deviation of $\mathbf{Y}(x)$ is shown in pink. It increases away from the data and when the data is noisy, as it should. **(Bottom):** The plots depict the values in the sequence $\mathbf{Y}(x)$ for four different values. For $x = 2$, both aleatoric and systemic uncertainties are low and convergence quickly. For $x = 5$, the aleatoric uncertainty data is high because the data is noisy and the convergence is slow. For $x = -2.0$ and $x = 7.0$ the systemic uncertainty is high because the points are out of distribution and the convergence is slow or erratic.

and vice versa. This approach lets us evaluate uncertainties at the pixel or node level. To obtain a scalar uncertainty estimate for the whole output $\mathbf{y}$, we average the values across all pixels/nodes.

In the result section, we show that this variance estimate strongly correlates with the actual accuracy of the prediction on experimental data. Note that for the arguments made for two in-distribution cases discussed at the end Section 3.1 to apply, the prediction $\mathbf{y}_i$ has to fall within the mode of the probability distribution. To maximize the chances of this happening, at training time, we supervise the network so that all $\mathbf{y}_i$ in $\mathbf{Y}$ are as close possible to the ground-truth by minimizing

$$\mathcal{L}_{total} = \sum_{i=1}^{N} \mathcal{D}(\mathbf{y}_i, \mathbf{y}^{\text{gt}}), \tag{4}$$

where D is a measure of distance and gt stands for ground-truth.

4. Experiments

Our approach applies to both classification and regression. To demonstrate the first, we use it for road delineation purposes, that is, classifying pixels in aerial images as belonging to roads or not. To demonstrate the second, we use it to assess the reliability of performance numbers—-drag for cars and lift-to-drag for airfoils—predicted by networks given 2D and 3D shapes as input. We then use these reliability estimates to implement a Bayesian optimization scheme that enables us to refine the shapes for improved performance. For both classification and regression, we outperform Deep Ensembles and MC-Dropout, along with Kriging in the regression case.

4.1. Delineation

Tasks such as road detection or modeling thin biological structures from images fall within the heading of visual delineation. After more than 50 years of research, it remains an open topic even though modern networks have boosted the state-of-the-art. Their final output often is a binary map indicating where in the image pixels belonging to structures of interest are. Generating such a map can be viewed as classifying the pixels as belonging to the target structures or not.

Datasets. We experimented on two publicly available datasets.

- *RoadTracer*. It comprises high-resolution satellite images covering urban areas of forty cities in six different countries (Bastani et al., 2018). Fifteen cities are set aside for validation purposes. The ground truth was generated using OpenStreetMap.

- *Massachusetts*. The Massachusetts dataset features both urban and rural neighborhoods, with many different kinds of roads ranging from small paths to highways. We used the same splits as in (Hu et al., 2019).

Together, these datasets exhibit a very large variety of urban scapes, which makes them a comprehensive benchmark for aerial road network reconstruction.

Baselines. Architectures such as U-Net (Ronneberger et al., 2015) or SegNet (Badrinarayanan et al., 2017) are commonly employed for delineation purposes, given their effectiveness in image segmentation challenges. For a fair comparison, all the method we tested rely on the standard U-Net (Ronneberger et al., 2015) architecture, with five blocks, each with three sequences of convolution-ReLU-batch normalization. Max-pooling in 2×2 windows followed each of the blocks. The initial feature size was set to 32 and grew to 1024 in the smallest feature map in the network. The network is trained to output a distance map that can then be

thresholded to produce the binary one. We augmented the training data with vertical and horizontal flips, along with random rotations. Thus, the four methods we compare are

- *U-Net*. Standard U-Net (Ronneberger et al., 2015).
- *MC-Dropout*. Adding drop-out layers (Gal & Ghahramani, 2016) into the standard U-Net to estimate the mean and variance of the predictions.
- *Deep Ensembles*. Using five standard *U-Net*s to estimate the mean and variance of the predictions (Lakshminarayanan et al., 2017).
- *Ours*. Using a recursive version of the standard U-Net (Wang et al., 2019), A dual-gated recurrent unit has been added in the bridge part of the network. During training, we performed three iterations. After each one, the output of the network is used as an additional input channel for the next one.

	Corr	*Comp*	*Qual*	*F1*	*APLS*	
U-Net	85.2	59.5	54.3	21.1	65.04	RT
MC-Dropout	**87.1**	58.2	54.1	20.4	58.78	RT
Deep Ensembles	**87.4**	**66.7**	**60.8**	**22.1**	**68.81**	RT
Ours	85.2	**77.8**	**68.6**	**24.5**	**77.21**	RT
U-Net	81.5	**91.4**	77.8	13.8	65.42	MS
MC-Dropout	81.6	**92.3**	78.2	13.6	59.65	MS
Deep Ensembles	**83.6**	90.4	**78.7**	**14.1**	**67.53**	MS
Ours	**92.3**	86.7	**81.1**	**15.4**	**78.04**	MS

Table 1. **Delineation accuracy on RoadTracer (top), and Massachusetts (bottom).** The best result in each category is in **bold** and the second best is in **bold**. Most correspond to *Ours* and *DeepE*.

Metrics. For road delineation, the true measure of success is preservation of the topology of the road networks rather than the very precise location of the centerline. This is ussually expressed in terms of hte following metrics:

- *APLS* (↑). Average Path Length Similarity, defined as an aggregation of relative length difference of shortest paths between pairs of corresponding points in the reconstructed and predicted maps (van Etten, 2019).
- *CCQ* (↑). Metric that measures spatial co-occurrence of annotated and predicted road pixels. The *Correctness*, *Completeness* and *Quality* are equivalent to precision, recall and intersection-over-union, where the definition of a true positive has been relaxed from spatial coincidence of prediction and annotation to co-occurrence within a distance of 5 pixels (Wiedemann et al., 1998).
- *F1 Score* (↑). A balance between precision and recall, the F1 score is twice the product of precision and recall divided by their sum. It's widely used in binary segmentation to equally weigh false positives and false negatives (Fawcett, 2006).

To similarly evaluate the quality of the uncertainty estimates, as in (Postels et al., 2022), we compute

- *Relative Area Under the Lift Curve* (rAULC). It is derived from the *Area Under the Lift Curve* concept (Vuk & Curk, 2006) and assesses the calibration quality of uncertainty measures across various methods.
- *Pearson Correlation Coefficient* (Corr). It measures the correlation between the estimated uncertainty and the actual error.

Evaluation. We report accuracy and uncertainty results in Tabs. 1 and 2. For all uncertainty evaluations, we calculate predictions and uncertainty for each pixel in the image. We then divide these into 512×512 crops, averaging the uncertainties and errors across each crop to ensure a more stable evaluation of the metrics. In terms of uncertainty estimation, *Deep Ensembles* and *Ours* are comparable and outperform the others. In terms of accuracy, *Ours* does best. To highlight this, in Fig. 3, we show scatter plots of estimated uncertainty vs actual accuracy. Note that our results exhibit a more linear behavior, which is what the Pearson Correlation Coefficient measures.

	rAULC	*Corr*	*Train*	*Inf*	
MC-Dropout	30.18	59.72	**1x**	**5x**	RT
Deep Ensembles	**72.19**	**79.42**	5x	**5x**	RT
Ours	**69.23**	**74.73**	**2.8x**	**2.7x**	RT
MC-Dropout	19.56	32.50	**1x**	**5x**	MS
Deep Ensembles	**78.65**	**76.39**	5x	**5x**	MS
Ours	**79.27**	**87.46**	**2.8x**	**2.7x**	MS

Table 2. **Delineation uncertainty quality on RoadTracer (top), and Massachusetts (bottom).** The best result in each category is in **bold** and the second best is in **bold**. Most correspond to *Ours* and *DeepE*. The *Train* and *Inf* metrics represent the total training time and inference time for the model, respectively, relative to a single model.

	ROC-AUC	*PR-AUC*	
MC-Dropout	61.25	62.64	RT
Deep Ensembles	**67.03**	**67.85**	RT
Ours	**67.09**	**72.11**	RT

Table 3. **RoadTracer vs Massachusetts out-of-distribution detection results.** The best result in each category is in **bold** and the second best is in **bold**. Most correspond to *Ours* and *DeepE*

To further evaluate our uncertainty estimates, we use the same insight as in (Malinin & Gales, 2018; Durasov et al., 2022a): A network trained on samples drawn from a given

distribution should be more confident on samples drawn from the same distribution than on samples drawn from a different one. In this context, we conducted an out-of-distribution detection task. We utilized the model trained on the *RoadTracer* dataset, treating its test set as in-distribution data. For out-of-distribution data, we selected the test set of the *Massachusetts* dataset. These two datasets exhibit markedly different landscapes. *RoadTracer* primarily features images of urban centers, whereas *Massachusetts* encompasses aerial images of rural areas.

We rely on the uncertainty measure generated by our model to decide whether a sample is in-domain or out-of-domain. We then apply standard detection metrics, ROC and PR AUCs (Malinin & Gales, 2018), to quantify the performance of our model. As for calibration metrics, we perform this evaluation using 512×512 crops: we average per-pixel uncertainties across the entire crop and classify it as in- or out-of-distribution based on this averaged uncertainty value. We report our results in Tab. 3.

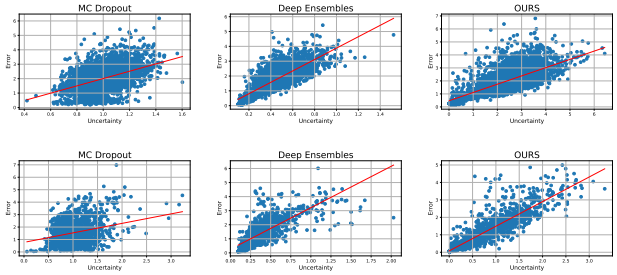


Figure 3. **Error vs Uncertainty.** These plots illustrate the error-uncertainty relationship for three methods on the *RoadTracer* **(Top)** and *Massachusetts* **(Bottom)** datasets. Our method surpasses the others on the *Massachusetts* dataset and performs comparably with Ensembles on *RoadTracer*. Correlation numbers are in Tab. 2. The red line indicates the optimal linear fit.

4.2. Aerodynamics Prediction and Optimization

We now showcase the effectiveness of our approach at estimating the uncertainly of surrogate models used to estimate and optimize aerodynamic performance.

Bayesian optimization. To refine 2D and 3D shapes and increase their expected aerodynamic performance, we rely on Bayesian Optimization (BO) (Mockus, 2012) as depicted by Fig. 4. Our implementation comprises four steps:

1. Use the training shapes and simulation results to train a surrogate model f_Θ.
2. Take each shape from the unlabelled pool and make a prediction with f_Θ.
3. Given the uncertainty of the predictions, compute the *acquisition function* (Auer, 2002; Qin et al., 2017) for the shapes in the unlabelled pool. This function balances between exploration and exploitation, and it is the key to the success of Bayesian optimization.
4. Pick the best new shapes in terms of the acquisition function, optimize their shape with gradient optimization, add them to the training set, and iterate.

These are standard BO steps, as described in Appendix A.2, except for step #4. It involves exploring the shape space without running additional simulations. It takes advantage of the fact that GNNs allow for gradient-based shape optimization. The key to implementing Bayesian shape optimization is an effective way to estimate not only the performance value associated with a shape but also the uncertainty on this estimate in step #3, which is something ordinary GNNs (Monti et al., 2017) do not provide, and which is being addressed by our approach.

Datasets. Given a set of N 3D shapes $\{\mathbf{x}_i\}_{1\leq i\leq N}$ represented by triangulated meshes, we run a physics-based simulator yielding a corresponding set $\{\mathbf{y}_i\}_{1\leq i\leq N}$ of physical values, such as pressure at each vertex. Let R be the function that takes as input the $\mathbf{y}$ values and returns an overall performance value $r = R(\mathbf{y})$, such as overall drag for a car or lift for a wing. R is task-specific. For example, in the case of drag, it is computed by integrating pressure values over the 3D shape. The simulator also generates $\{r_i\}_{1\leq i\leq N}$ in conjunction with the $\mathbf{y}_i$'s. Assuming that each mesh $\mathbf{x}_i$ is parameterized by a lower-dimensional latent vector $\mathbf{z}_i$ and that there is a differentiable mapping $\mathbf{P} : \mathbf{z} \rightarrow \mathbf{x}$, this gives us the initial training set $T = \{(\mathbf{z}_i, \mathbf{x}_i, r_i, \mathbf{y}_i)\}_i$ that we need to initialize our optimization scheme. Similarly, we expect a larger pool of unlabeled shapes, consisting of latent vectors and meshes denoted as $U = \{(\mathbf{z}_i, \mathbf{x}_i)\}_i$, but no simulation data. We train the surrogate model using samples from T (Step 1) and use it to perform predictions (Step 2), compute the acquisition function (Step 3), and select samples for simulations from the set U (Step 4).

- *Airfoils*. We generated a dataset comprising 1500 two-dimensional airfoil shapes. This was achieved by randomly selecting NACA parameters, $\mathbf{z}_i$, and producing corresponding airfoil contours, $\mathbf{x}_i$. The pressure distribution, $\mathbf{y}_i$, over each airfoil surface was computed using the XFoil simulator. Additionally, the global lift-to-drag ratio, $\mathbf{r}_i$, a measure of aerodynamic efficiency, was calculated for each shape. The dataset was divided into 1000 training samples, 300 testing samples, and 200 high-performance shapes, treated as out-of-distribution samples for uncertainty analysis.
- *Cars*. We use a cleaned-up and processed subset of the ShapeNet dataset (Chang et al., 2015) that features $N = 1500$ car meshes suitable for CFD simulation.

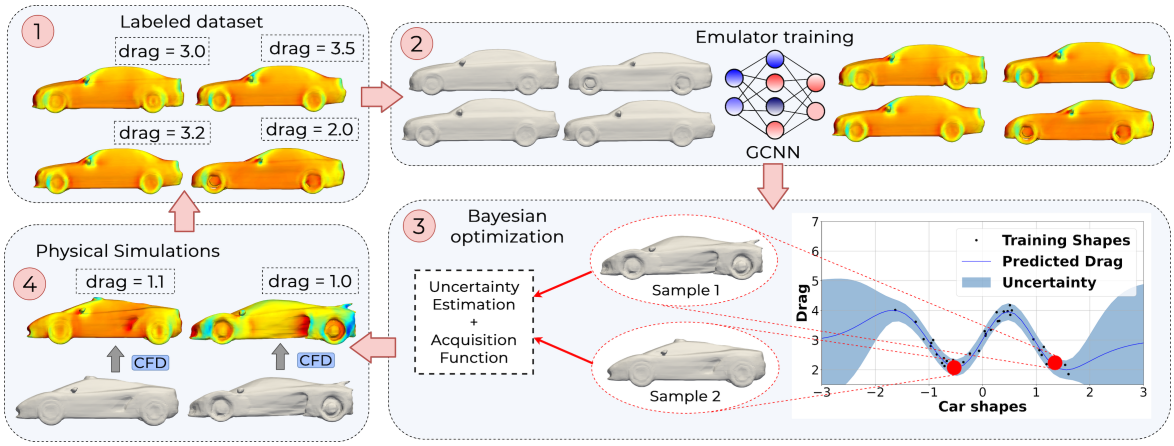


Figure 4. **Bayesian optimization pipeline. (1)** Run physical simulations. **(2)** Train the GNN. **(3)** Evaluate the acquisition function on samples without an associated physical simulation. **(4)** Select promising samples according the acquisition function, optimize their shape, add them to the training set, and go back to step 1.

For each such mesh $\mathbf{x}_i$, we run OpenFOAM (Jasak et al., 2007) to estimate the pressure field $\mathbf{y}_i$ and drag $\mathbf{r}_i$ created by air traveling at 15 meters per second towards the car. We also use MeshSDF (Remelli et al., 2020) in conjunction with an auto-decoding approach (Park et al., 2019) to learn a function $P : \mathbb{R}^{256} \rightarrow \mathbb{R}$ and a set of latent vectors $\{\mathbf{z}_i\}$ such that $\forall i \ \mathbf{x}_i = P(\mathbf{z}_i)$. We use the same protocols as for *Airfoils* for splitting.

Baselines. We compare our method against widely recognized and universally adopted baselines, which are considered the gold standard in the field:

- *KNN*: Given a set of simulated shapes, we use a standard K-Nearest Neighbors regressor to estimate the performance of additional shapes and add the best one to the training set. No uncertainty is computed. For this approach, we use $K = 8$ and employ distance-based neighbor weighting, as this has been shown to be the optimal choice for this task (Baqué et al., 2018).
- *Kriging*: Using a Gaussian Processes (GPs) to estimate performance values and corresponding uncertainty (Laurenceau et al., 2010) directly from parameters $\mathbf{z}$. As discussed above, it can be directly used to perform Bayesian Optimization. For GPs, we use the squared exponential kernel, which has been shown to be particularly effective for aerodynamic prediction (Toal & Keane, 2011; Rosenbaum & Schulz, 2013).
- *GNN*: GNNs (Baqué et al., 2018; Hines & Bekemeyer, 2023) are a valid alternative to GPs for the purpose of estimating performance numbers. Since they do not compute uncertainties, we simply add the ones that receive the best score from the GNN to the training set and optimize their shape as in (Baqué et al., 2018).
- *Deep Ensembles*: We use sets of GNNs to predict mean and variances of performance values, which is known as an Ensemble-based technique. These are then exploited by the procedure introduced previously in this section. For all of the experiments, we use 5 GNNs in ensemble.
- *MC-Dropout*: Instead of using Ensembles to estimate the performance numbers and their uncertainty, we use MC-Dropout in the Bayesian optimization procedure.
- *Ours*: Using the iterative GNN to simultaneously estimate the performance values and their uncertainty for the Bayesian optimization procedure.

Metrics. We use the following metrics to evaluate the quality of our baselinse in terms of predictions accuracy and uncertainty estimation:

- *Mean absolute error* (↓) (MAE). It is the average of the absolute differences between the predicted and actual values. It is a common metric for regression tasks.
- *Opimized performance*. We use our baselines to perform Bayesian optimization as it was described above. We then report the best performance value, $\mathbf{r}_i$ lift-to-drag value for airfoils and $\mathbf{r}_i$ drag value for cars, obtained by each method and the dynamics of optimization process.

Evaluation. As all six methods being compared rely on an emulator, the left *Airfoil* and *Cars* plots in Fig. 5 depict the accuracy of each on the test set as a function of the number of samples from the training set used to train it. *Ours* outperforms the others consistently, especially when there are only a few training examples. For this accuracy

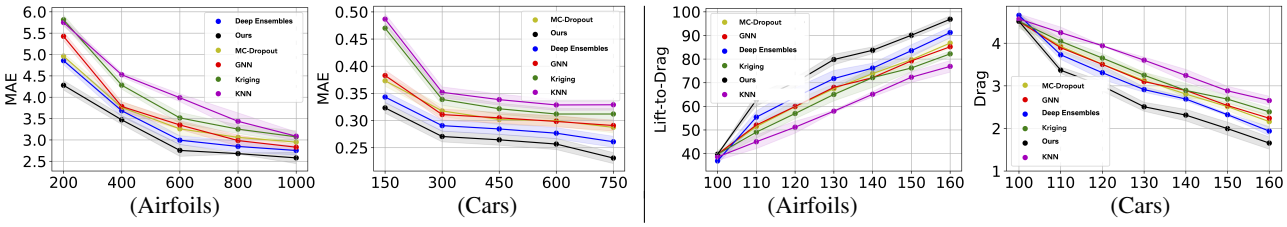


Figure 5. **Left.** Accuracy of the lift-to-drag estimate as a function of the number of exemplars used to train the emulators. **Right.** Lift-to-drag ratio of the shapes during optimization, as a function of number of iterations.

	Krig	*MC-DP*	*DeepE*	*Ours*	
ROC-AUC	0.79	0.84	**0.88**	0.87	AIR
PR-AUC	0.78	0.82	0.86	**0.88**	AIR
ROC-AUC	0.62	0.73	**0.90**	0.86	CAR
PR-AUC	0.52	0.62	0.78	**0.79**	CAR

Table 4. **Evaluation the uncertainty measure for 2D airfoils (AIR) and 3D cars (CAR).** The best result in each category is in **bold** and the second best is in bold. They all correspond to *Ours* and *Deep Ensembles*. The two approaches are comparable in terms of evaluating uncertainty but the reentrant GNNs deliver better accuracy, as shown in Fig. 5.

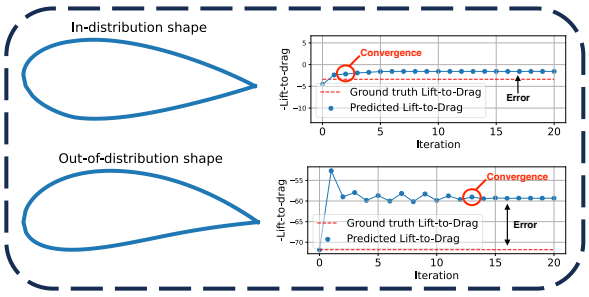


Figure 6. **Convergence rate vs error.** For the in-distribution airfoil at the top, the consecutive values of predicted lift-to-drag values converge quickly and the limit is very close to the correct answer. By contrast, for an out-of-distribution airfoil below, the convergence is much slower and the limit is wrong. This is a behavior that we have consistently observed in our experiments.

evaluation, at each iteration, we add 100 new samples for *Airfoils* and 150 for *Cars*.

As in the delineation experiments, we also evaluate the quality of uncertainty estimates through the lens of Out-of-distribution Detection (Fort et al., 2021) task. To this end, given all the 3D shapes we have, we took the 200 top-performing ones in terms of their lift-to-drag ratio to be the out-of-distribution samples. For both datasets, the remaining shapes were then considered as the in-distribution ones. One thousand of these were used to train the emulators, and the others were used for testing purposes. After training, we generated uncertainty values for each shape in the in-distribution and out-of-distribution test sets. Finally, we computed standard ROC-AUC, PR-AUC (Malinin & Gales, 2018) metrics for in- or out-of-distribution classification based on the uncertainty estimate. As can be seen in the top rows of Tab. 4, our approach generates uncertainty of a quality similar to that of ensembles. Furthermore, as shown in Fig. 7, our method often only require 3 iterations to converge. This makes them a little faster than an ensemble of 5 ordinary GNNs and, importantly, requires far less memory and training time. We provide more details in Appendix A.3.

We now turn to shape optimization using each one of the 6 methods. In each case, we used 100 randomly chosen samples from the training set, along with the corresponding simulations, to train the initial emulator. The rest of the training set, plus the OOD set, were treated as a set of unlabelled shapes. After the initial training, we ran the inference for each shape in it. For non-uncertainty approaches (*KNN* and *GNN*), this yielded predicted performance values, and for the other values of the acquisition function (UCB (Auer, 2002) with $\lambda = 3$). We sorted the unlabelled shapes according to these values and picked the 10 best. For GNN-based methods, for each one of these 10 shapes, we also performed 10 steps of gradient-based optimization (Kingma & Ba, 2015). This relatively small number of iterations was chosen to allow us to reap the benefits of GNN-based shape optimization (Baqué et al., 2018), without moving too far away from the starting points and producing shapes whose acquisition value is too different from that of the starting point. We discuss the influence of the number of iterations we perform in Appendix A.5. Finally, we ran simulations for these chosen shapes, added them to the training set, and iterated. For each method, we ran this whole process three times and plot the resulting lift-to-drag ratios as a function of the number of BO iterations performed in the *airfoil* plot in Fig. 5. The shaded areas depict the corresponding variances. Again, our method outperforms the other approaches by a statistically significant margin.

Recall from Section 3 that our approach is predicated on the fact that convergence of the iterative GNNs can be expected to be slower for out-of-distribution samples than for in-distribution ones. The plot on the left side of Fig. 7 validates this hypothesis on the in-distribution and out-of-distribution splits.

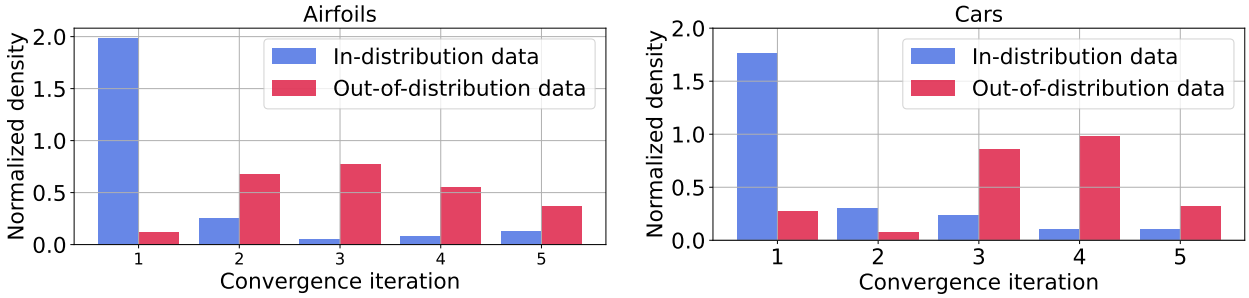


Figure 7. **Convergence rates for in- and out-of-distribution samples.** We plot the distribution of the number of iterations to convergence of our iterative GNNs for in-distribution vs. out-of-distribution samples from the test sets of airfoils and cars. In general, convergence takes significantly fewer steps for in-distribution samples than for out-of-distribution ones.

5. Ablation study

5.1. Calibration Evaluation

Though rAULC and correlation metrics from Sec. 4 provide comprehensive information about calibration quality, we also provide the results for Expected Calibration Error (ECE) (Guo et al., 2017) evaluation. Tab. 5 provides ECE values for different methods, including MC-Dropout, Ensembles, and our proposed method. The results show that our method achieves better calibration than both Ensembles and MC-Dropout for all datasets.

	MC-Dropout	*Ensembles*	*Ours*
ECE (RT)	0.997	1.138	**0.475**
ECE (MS)	0.558	0.794	**0.419**
ECE (Airfoils)	1.758	1.162	**1.142**
ECE (Cars)	0.267	0.232	**0.227**

Table 5. **Expected Calibration Error (ECE) evaluation.** As it was previously demonstrated for the rAULC metric, our method outperforms other approaches in terms of ECE calibration.

5.2. Image Classification

We expanded our experimental evaluation to include the widely-used task of image classification on the MNIST dataset, a popular benchmark for out-of-distribution (OOD) detection and uncertainty estimation. For the OOD task, we used FashionMNIST as the OOD samples, which is a standard choice in this context. To add more variability, we ran these experiments with two model architectures:

- **CNN Architecture:** A convolutional neural network (CNN) with several 2D convolutional layers followed by a fully-connected classification head.
- **MLP Architecture:** A multilayer perceptron (MLP) with 5 fully-connected layers, treating images as 784-dimensional vectors.

We evaluated these experiments using the same metrics as in our previous evaluations, ensuring consistency and comparability. The results are summarized in Table 6. As for our previous results, our method outperforms other approaches both in terms of model's accuracy and uncertainty quality.

	MC-DP	*Ensembles*	*Ours*	
Acc	97.8	98.8	**99.0**	CNN
ECE	0.021	0.018	**0.013**	CNN
ROC-AUC	94.4	**98.5**	**98.5**	CNN
PR-AUC	93.8	98.1	**98.5**	CNN
Train	1x	5x	3x	CNN
Inf	5x	5x	3x	CNN
Acc	96.1	**97.7**	97.4	MLP
ECE	0.026	0.022	**0.020**	MLP
ROC-AUC	61.5	89.5	**89.6**	MLP
PR-AUC	69.2	**89.9**	88.7	MLP
Train	1x	5x	3x	MLP
Inf	5x	5x	**3x**	MLP

Table 6. **MNIST classification results for CNN (top), and MLP (bottom) architectures.** The best result in each category is in **bold** and the second best is in **bold**. Most correspond to Ours and Deep Ensembles.

6. Conclusion

We have presented an approach to assessing the quality of predictions by iterative networks at a much lower cost than Deep Ensembles, currently the most reliable approach to such an assessment, and more reliably than other state-of-the-art methods. Our method relies on measuring how fast successive estimates converge and does require neither any change in network architecture nor training more than one. In the shape optimization part of this work, we have focused on aerodynamics but the principle applies to many other devices, ranging from the cooling plates of an electric vehicle battery to the optics of an image acquisition device. In future work, we will therefore explore a broader set of potential applications.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alain, G. and Bengio, Y. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

Allaire, G. A Review of Adjoint Methods for Sensitivity Analysis, Uncertainty Quantification and Optimization in Numerical Codes. *Ingénieurs de l'Automobile*, 836: 33–36, July 2015.

Amersfoort, V., Smith, J. A., Teh, L. A., Whye, Y., and Gal, Y. Uncertainty Estimation Using a Single Deep Deterministic Neural Network. In *International Conference on Machine Learning*, pp. 9690–9700, 2020.

Antorán, J., Allingham, J., and Hernández-Lobato, J. M. Depth uncertainty in neural networks. *Advances in neural information processing systems*, 33:10620–10634, 2020.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. In *International Conference on Learning Representations*, 2020.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

A.Vaswani, Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Baqué, P., Remelli, E., Fleuret, F., and Fua, P. Geodesic Convolutional Shape Optimization. In *International Conference on Machine Learning*, 2018.

Bastani, F., He, S., Alizadeh, M., Balakrishnan, H., Madden, S., Chawla, S., Abbar, S., and Dewitt, D. Roadtracer: Automatic Extraction of Road Networks from Aerial Images. In *Conference on Computer Vision and Pattern Recognition*, 2018.

Bengio, Y., Yao, L., Alain, G., and Vincent, P. Generalized denoising auto-encoders as generative models. *Advances in neural information processing systems*, 26, 2013.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. Human Pose Estimation with Iterative Error Feedback. In *Conference on Computer Vision and Pattern Recognition*, 2016.

Chang, A., Funkhouser, T., G., L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. Shapenet: An Information-Rich 3D Model Repository. In *arXiv Preprint*, 2015.

Chen, R., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, 2018.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *arXiv Preprint*, 2015.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, pp. 2510–2521. PMLR, 2021.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *arXiv Preprint*, 2018.

Durasov, N., Romanov, M., Bubnova, V., Bogomolov, P., and Konushin, A. Double Refinement Network for Efficient Monocular Depth Estimation. In *International Conference on Intelligent Robots and Systems*, pp. 5889–5894, 2019.

Durasov, N., Bagautdinov, T., Baque, P., and Fua, P. Masksembles for Uncertainty Estimation. In *Conference on Computer Vision and Pattern Recognition*, 2021.

Durasov, N., Dorndorf, N., and Fua, P. ZigZag: Universal Sampling-free Uncertainty Estimation Through Two-Step Inference. *arXiv Preprint*, 2022a.

Durasov, N., Dorndorf, N., and Fua, P. Partal: Efficient partial active learning in multi-task visual settings. *arXiv preprint arXiv:2211.11546*, 2022b.

Fawcett, T. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Fort, S., Ren, J., and Lakshminarayanan, B. Exploring the limits of out-of-distribution detection. *Advances in Neural Information Processing Systems*, 34:7068–7081, 2021.

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, 2016.

Graves, A. Practical variational inference for neural networks. *Advances in Neural Information Processing Systems*, 24, 2011.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On Calibration of Modern Neural Networks. In *International Conference on Learning Representations*, 2017.

Gupta, K. and Chandraker, M. Neural Mesh Flow: 3D Manifold Mesh Generation via Diffeomorphic Flows. In *Advances in Neural Information Processing Systems*, 2020.

Gustafsson, F. K., Danelljan, M., and Schön, T. B. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. In *Conference on Computer Vision and Pattern Recognition*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hernández-Lobato, J. M. and Adams, R. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hines, D. and Bekemeyer, P. Graph Neural Networks for the Prediction of Aircraft Surface Pressure Distributions. *Aerospace Science and Technology*, 137:108268, 2023.

Hu, X., Li, F., Samaras, D., and Chen, C. Topology-Preserving Deep Image Segmentation. In *Advances in Neural Information Processing Systems*, pp. 5658–5669, 2019.

Japkowicz, N., Myers, C., Gluck, M., et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pp. 518–523. Citeseer, 1995.

Jasak, H., Jemcov, A., Tukovic, Z., et al. OpenFOAM: A C++ Library for Complex Physics Simulations. In *International workshop on coupled methods in numerical dynamics*, 2007.

Kamyshanska, H. and Memisevic, R. On autoencoder scoring. In *International Conference on Machine Learning*, pp. 720–728. PMLR, 2013.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimisation. In *International Conference on Learning Representations*, 2015.

Kingma, D. P., Salimans, T., and Welling, M. Variational Dropout and the Local Parameterization Trick. *Advances in Neural Information Processing Systems*, 28, 2015.

Kumar, S. K. On Weight Initialization in Deep Neural Networks. *arXiv Preprint*, 2017.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Laurenceau, J., Meaux, M., Montagnac, M., and Sagaut, P. Comparison of Gradient-Based and Gradient-Enhanced Response-Surface-Based Optimizers. *American Institute of Aeronautics and Astronautics Journal*, 48(5):981–994, 2010.

Mackay, D. J. Bayesian Neural Networks and Density Networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80, 1995.

Malinin, A. and Gales, M. Predictive Uncertainty Estimation via Prior Networks. In *Advances in Neural Information Processing Systems*, 2018.

Medioni, G., Tang, C., and Lee, M. Tensor Voting: Theory and Applications. In *Reconnaissance des Formes et Intelligence Artificielle*, 2000.

Mnih, V. and Hinton, G. Learning to Detect Roads in High-Resolution Aerial Images. In *European Conference on Computer Vision*, pp. 210–223, 2010.

Mockus, J. *Bayesian Approach to Global Optimization: Theory and Applications*, volume 37. Springer Science & Business Media, 2012.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *Conference on Computer Vision and Pattern Recognition*, pp. 5425–5434, 2017.

Mosińska, A., Marquez-Neila, P., Kozinski, M., and Fua, P. Beyond the Pixel-Wise Loss for Topology-Aware Delineation. In *Conference on Computer Vision and Pattern Recognition*, pp. 3136–3145, 2018.

Mukhoti, J., van Amersfoort, Torr, J. A., HS, P., and Gal, Y. Deep Deterministic Uncertainty for Semantic Segmentation. In *arXiv Preprint*, 2021.

Newell, A., Yang, K., and Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *European Conference on Computer Vision*, 2016.

Oner, D., Koziński, M., Citraro, L., Dadap, N. C., Konings, A. G., and Fua, P. Promoting Connectivity of Network-Like Structures by Enforcing Region Separation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5401–5413, 2021.

Oner, D., Koziński, M., Citraro, L., and Fua, P. Adjusting the Ground Truth Annotations for Connectivity-Based Learning to Delineate. *IEEE Transactions on Medical Imaging*, 41(12):3675–3685, 2022.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Dhift. In *Advances in Neural Information Processing Systems*, pp. 13991–14002, 2019.

Park, J. J., Florence, P., Straub, J., Newcombe, R. A., and Lovegrove, S. Deepsdf: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Devito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic Differentiation in Pytorch. In *Advances in Neural Information Processing Systems*, 2017.

Perrone, M. P. and Cooper, L. N. When networks disagree: Ensemble methods for hybrid neural networks. In *How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper*, pp. 342–358. World Scientific, 1995.

Pinheiro, P. and Collobert, R. Recurrent Neural Networks for Scene Labelling. In *International Conference on Machine Learning*, 2014.

Postels, J., Ferroni, F., Coskun, H., Navab, N., and Tombari, F. Sampling-Free Epistemic Uncertainty Estimation Using Approximated Variance Propagation. In *Conference on Computer Vision and Pattern Recognition*, pp. 2931–2940, 2019.

Postels, J., Segu, M., Sun, T., Gool, L. V., Yu, F., and Tombari, F. On the Practicality of Deterministic Epistemic Uncertainty. In *International Conference on Machine Learning*, pp. 17870–17909, 2022.

Qin, C., Klabjan, D., and Russo, D. Improving the expected improvement algorithm. *Advances in Neural Information Processing Systems*, 30, 2017.

Remelli, E., Lukoianov, A., Richter, S., Guillard, B., Bagautdinov, T., Baque, P., and Fua, P. MeshSdf: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*, 2020.

Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pp. 234–241, 2015.

Rosenbaum, B. and Schulz, V. Response Surface Methods for Efficient Aerodynamic Surrogate Models. In *Computational Flight Testing*, pp. 113–129. Springer, 2013.

Sabokrou, M., Fathy, M., and Hoseini, M. Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder. *Electronics Letters*, 52(13):1122–1124, 2016.

Seyedhosseini, M., Sajjadi, M., and Tasdizen, T. Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *International Conference on Computer Vision*, 2013.

Shen, W., Wang, B., Jiang, Y., Wang, Y., and Yuille, A. Multi-Stage Multi-Recursive-Input Fully Convolutional Networks for Neuronal Boundary Detection. In *International Conference on Computer Vision*, 2017.

Sironi, A., Turetken, E., Lepetit, V., and Fua, P. Multiscale Centerline Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1327–1341, 2016.

Solinas, M., Galiez, C., Cohendet, R., Rousset, S., Reyboz, M., and Mermillod, M. Generalization of iterative sampling in autoencoders. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 877–882. IEEE, 2020.

Toal, D. and Keane, A. Efficient Multipoint Aerodynamic Design Optimization via Cokriging. *Journal of Aircraft*, 48(5):1685–1695, 2011.

Tu, Z. and Bai, X. Auto-Context and Its Applications to High-Level Vision Tasks and 3D Brain Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

van Etten, A. Spacenet road detection and routing challenge part ii—apls implementation, 2019.

Vuk, M. and Curk, T. ROC curve, lift chart and calibration plot. *Advances in methodology and Statistics*, 3(1):89–108, 2006.

Wang, W., Yu, K., Hugonot, J., Fua, P., and Salzmann, M. Recurrent U-Net for Resource-Constrained Segmentation. In *International Conference on Computer Vision*, 2019.

Wen, Y., Tran, D., and Ba, J. Batchensemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations*, 2020.

Wiedemann, C., Heipke, C., Mayer, H., and Jamet, O. Empirical Evaluation of Automatically Extracted Road Axes. In *Empirical Evaluation Techniques in Computer Vision*, pp. 172–187, 1998.

Xu, D., Ouyang, W., Wang, X., and Sebe, N. Pad-Net: Multi-Task Guided Prediction-And-Distillation Network for Simultaneous Depth Estimation and Scene Parsing. In *Conference on Computer Vision and Pattern Recognition*, pp. 675–684, 2018.

Zhang, P., Wang, F., and Zheng, Y. Deep Reinforcement Learning for Vessel Centerline Tracing in Multi-Modality 3D Volumes. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pp. 755–763, 2018a.

Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., and Yang, J. Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 235–251, 2018b.

Zhou, Y. Rethinking reconstruction autoencoder-based out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7379–7387, 2022.

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., and Liang, J. Unet++: A Nested U-Net Architecture for Medical Image Segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, 2018.

A. Appendix

In this section, we first examine the behavior of iterative networks in a very simple case. We then provide details about the training procedure and additional supporting evidence for some of the claims made in the paper.

A.1. Analysis of a Simple Case

To model the behavior of our iterative networks in a simpler and easier-to-analyze context, we replace CNNs and GNNs with a perceptron f_W that takes two scalar inputs x and y and outputs a scalar. Given a training set $\{(x_i, r_i)\ , 1 \leq i \leq N\}$, we make it iterative by computing

$$\begin{aligned} y_i^1 &= f_W(x, 0) \\ y_i^2 &= f_W(x, y_i^1) \\ &\dots \\ y_i^t &= f_W(x, y_i^{t-1}) \end{aligned} \tag{5}$$

for each i, where t_i is a different random integer between 1 and T for each sample. In these examples, we use $T = 5$. We then minimize the total loss $\sum_i (r_i - y_i^{(t_i)})^2$. Fig. 8 depicts the results of this process when the x_i are uniformly sampled between 0 and 1 and the r_i are taken to be $sin(a * x_i) * cos(b * x_i)$ for different values of a and b. For values of x between 0 and 1, that is, for values that are within the training domain, we have $y_i^0 \approx y_i^1 .... \approx y_i^T \approx r_i$. In contrast, out of domain, that is, outside the range [0,1], this is not true anymore, and we can see strong oscillations of the successive y_i^t values for $1 \leq t \leq T$. This makes sense because deep networks are known not to extrapolate well. Thus, even though the network is trained to produce similar predictions for all values of t in-domain, the out-of-domain predictions are essentially random, and there is no reason for them to be equal. In the results section, we showed that, for both airfoils and car shapes, out-of-domain values of x tend to produce oscillations and slow convergence. Interestingly, we observe exactly the same behavior on this very simple example, as evidenced by the fact that the curves of Fig. 8 are *not* superposed for $x < 0$ and $x > 1$.

The exact values obtained for these out-of-domain samples are very hard to predict. As can be seen by comparing the two rows of Fig. 8, they depend critically on the chosen activation function, $\tanh$ or ReLu in this case. They also depend heavily on how the networks have been initialized, as can be seen in Fig. 9. In one case, we initialized the weights of our perceptrons using normally distributed weights. In the other, we used the slightly more sophisticated Xavier Initialization (Kumar, 2017).

Crucially, in all cases, seeing large variations in the values of the successive $y_k(x)$ for a given x is *always* a warning sign that the estimated value is likely to be incorrect. This is what we exploit in this work.

A.2. Bayesian Optimization

Given a performance estimator of uknown reliability, exploration-and-exploitation techniques seek to find global optimum of that estimator while at the same time accounting for potential inaccuracies in its predictions.

Bayesian Optimization (BO) (Mockus, 2012) is one of the best-known approaches to finding global minima of a black-box function $g : \mathbf{A} \rightarrow \mathbb{R}$, where $\mathbf{A}$ represents the space of possible shapes, without assuming any specific functional form for g. It is often preferred to more direct approaches, such as the adjoint method (Allaire, 2015), when g is expensive to evaluate, which often is the case when g is implemented by a physics-based simulator.

BO typically starts with a surrogate model $f_\Theta : \mathbf{A} \rightarrow \mathbb{R}$ whose output depends on a set of parameters Θ. f_Θ is assumed to approximate g, to be fast to compute, and to be able to evaluate the reliability of its own predictions in terms of a uncertainty. It is used to explore $\mathbf{A}$ quickly in search of a solution of $\mathbf{x}^* = \text{argmin}_{\mathbf{x} \in \mathbf{A}}\, g(\mathbf{x})$. Given an initial training set $\{(\mathbf{x}_i, r_i)\}_i$ of input shapes $\mathbf{x}_i$ and outputs $r_i = g(\mathbf{x}_i)$, it iterates the following steps:

Step 1: Find Θ that yields the best possible prediction by f_Θ.

Step 2: Generate new samples not present in the training set.

Step 3: Evaluate an *acquisition function* on these samples.

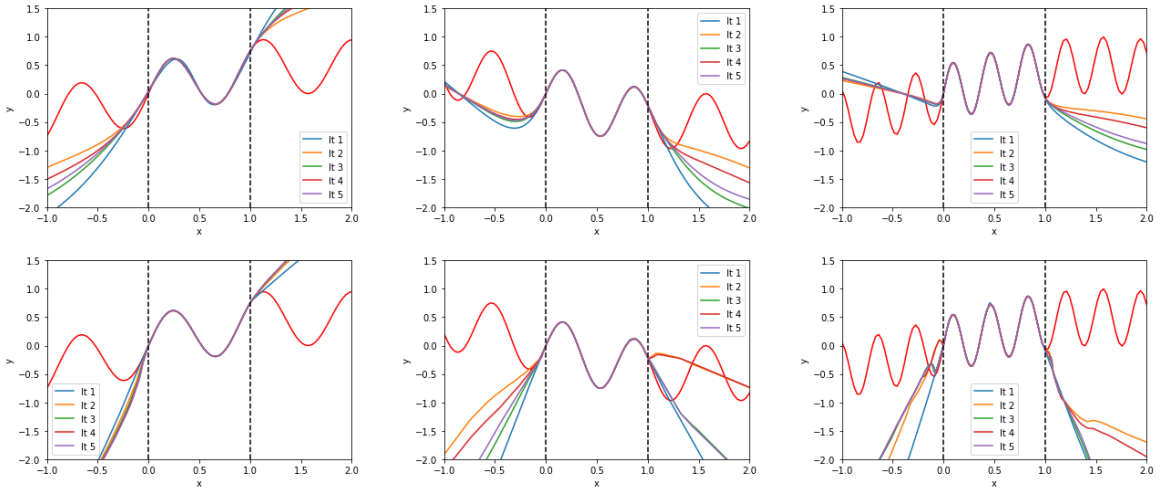
Step 4: Add the best ones to the training set and go back to Step 1.

As shown in the example of Fig. 13, the role of the acquisition function is to gauge how desirable it is to evaluate a point, based on the current state of the model. It is often taken to be the *Expected Improvement* (EI) (Qin et al., 2017) or *Upper Confidence Bound* (UCB) (Auer, 2002) that favor samples with the greatest potential for improvement over the current optimum. It is computed as a function of the values predicted by the surrogate and their associated uncertainty.

A.3. Training setups

For our experiments, we used single Tesla V100 GPU with 32Gb of memory. The training process was implemented using the Pytorch (Paszke et al., 2017) and Pytorch Geometrics (Fey & Lenssen, 2019) frameworks.

Airfoils. For airfoils, we have generated 1500 shapes from NACA parameters, and simulated pressure and lift-to-drag values with XFOIL simulator. As an emulator, we use architecture that consists of 35 GMM layers (Monti et al., 2017)



(a = 3, b = 4) (a = 5, b = 4) (a = 8, b = 9)

Figure 8. **Learning to interpolate a 1D function**. Using a two-layer perceptron to interpolate $f(x) = \sin(a * x)\cos(b * y)$ given training pairs $(x, f(x))$ for which $0 < x < 1$. Each curve represents the value of y_i^t from Eq. 5 for values of x ranging from -1.0 to 2.0, that is, both inside and outside the training domain. There is one curve per iteration i in Eq. 5, ranging from 1 to 5. **Top row.** Taking $\tanh$ to be the activation function. **Bottom row.** Using ReLu.

	GNN	*Deep Ensembles*	*MC-Dropout*	*Ours*	
Memory	1x	5x	1x	1x	AIR
Inf. Time	1x	5x	5x	3x	
Train. Time	1x	5x	1x	2x	
Memory	1x	5x	1x	1x	CAR
Inf. Time	1x	5x	5x	3x	
Train. Time	1x	5x	1x	2x	

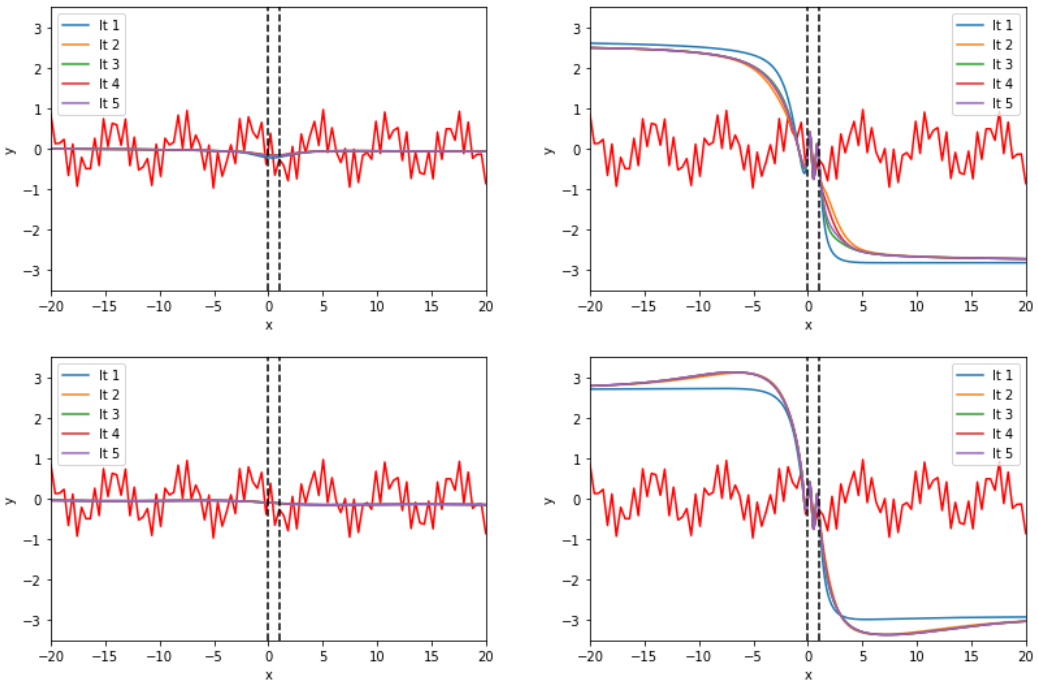
Table 7. **Computational costs.** The Memory, Inference Time, and Training Time metrics measure the amount of time and memory required to train the network(s) and to perform inference, in comparison to a single model.

with ReLU activations. First, we extract node features with these GMM layers and pass them to pressure branch, that consists out of 3 GMM layers, and lift-to-drag branch, that uses global pooling and 3 fully-connected layers to predict final scalar. For training, we use Adam optimizer (Kingma & Ba, 2015) and perform 200 epochs with 128 batch size and 0.001 learning rate. Both for lift-to-drag and pressure, we use mean squared error (MSE) loss and combine them into final loss with weights 1 for scalar and 100 for pressure.

Cars. For cars dataset, we have generated 1500 shapes from MeshSDF vectors, and simulated pressure and drag values with OpenFOAM simulator. As an emulator, we use architecture that consists of 50 GMM layers with ELU activations (Clevert et al., 2015) and skip-connections (He et al., 2016). Similar to airfoils, we extract node features with these GMM layers and pass them to pressure branch, that consists out of 5 GMM layers, and drag branch, that uses global pooling and 5 fully-connected layers to predict final scalar. For training, we use Adam optimizer and perform 6 epochs with 8 batch size and 0.001 learning rate. Both for lift-to-drag and pressure, we use mean squared error (MSE) loss and combine them into final loss with weights 1 for scalar and $1/200$ for pressure.

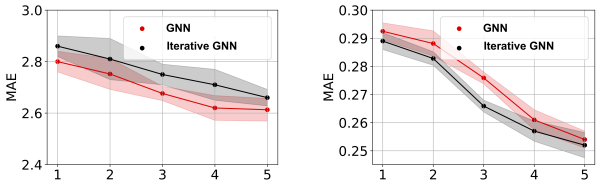
A.4. Propagating Information

In a standard GNN information is propagated across the shape with each successive convolution. Hence, it is comparatively slow and our reentrant GNNs address this. To support, this claim we ran an experiment to test the influence of the receptive fields of the GNNs, which control the speed at which information percolates across the network. We trained 5 airfoils and car emulator models of increasing depth while keeping total weights number fixed. Starting from the original architecture, we plot the prediction mean error for both lift-to-drag and drag in Fig. 10 in red. As expected, the error decreases as depth increases and more information is propagated across the shape. The exact same behavior can be observed when using our GNN run iteratively, as shown by the black curves. This supports our claim that each iteration helps propagate the information



(Before training) (After training)

Figure 9. **Influence of initialization.** We plot the same curves as in Fig. 8 when learning to interpolate the function $f(x) = \sin(5 * x)\cos(4 * y)$, but over a more extended range of x and starting from a different initialization of the perceptron weights in each row. **Before training.** As before, each curve represents the values y_i^t as a function of x. Here we plot those returned by our perceptrons after initialization of their activation weights, but before actual training. The two plots correspond to the two different initializations. **After training.** Values after training. There are similar for $0 < x < 1$ but different out of this domain. Not that they are also very different across the two rows because of the slightly different initializations.



Lift-to-Drag Drag

Figure 10. **Propagating information across an shape**. A comparable behavior is observed when increasing the depth of a standard GNN (red curves) and when running several iterations of a shallower iterative GNN (black curves).

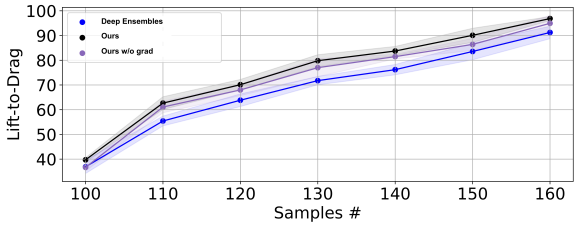


Figure 11. **Impact of refining the shapes.** Turning on gradient optimization of new samples delivers a small performance increase, but smaller than the one used by replacing ensembles with a version of our approach without the refinement.

across the shape just as effectively as when using the deeper network.

A.5. Gradient Optimization

Given the shapes selected according to the acquisition function during Bayesian Optimization, our method performs several gradient steps in order to refine these shapes and makes them more performant. In this subsection, we examine the impact of performing this optimization.

In the results shown in the main paper, given the current state of the emulator, we performed 10 steps of an Adam-based optimizer with a $1e-4$ learning rate to refine each selected shape. In Fig. 11, we plot the results obtained for the airfoils by doing this refinement (*Ours*), not doing it (*Ours* w/o grad), or using deep ensembles (*Deep Ensembles*) baseline. *Ours* without refinement already delivers an improvement overs ensembles, with a further but smaller improvement when performing the refinement. We tried increasing the number of refinement steps but that brought no further improvement.

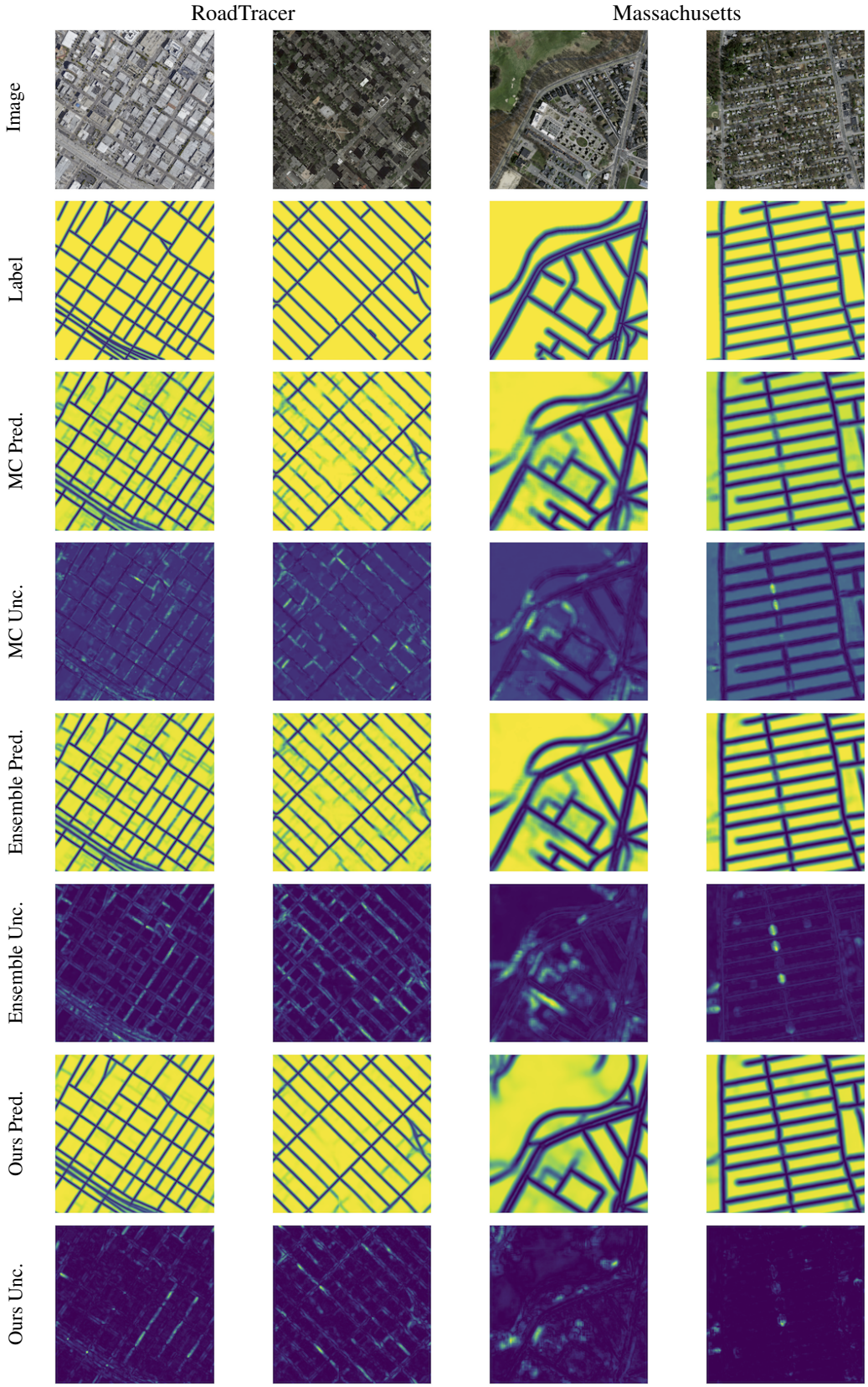


Figure 12. Test predictions and uncertainties produced by different methods on two datasets.

	MC-DP	*Ens.*	*Ens. Iter.*	*Ours*
Corr	87.1	87.4	**88.1**	85.2
Comp	58.2	66.7	76.1	**77.8**
Qual	54.1	60.8	**69.1**	68.6
F1	20.4	22.1	24.4	**24.5**
APLS	58.78	68.81	**78.40**	77.21
rAULC	30.18	**72.19**	69.03	69.23
Corr (unc.)	59.72	**79.42**	79.17	74.73
ECE	0.997	1.138	0.850	**0.475**
Train	1x	5x	14x	2.8x
Inf	5x	5x	13.5x	2.7x

Table 8. **Uncertainty and performance metrics on RT Dataset.** The best result in each category is in **bold** and the second best is in bold. Most correspond to Ours and Iterative Ensembles.

A.6. Ensembles on Iterative Architecture

Another baseline for comparison could involve applying Deep Ensembles (DE) to our iterative architecture. While this approach would significantly increase computational complexity, it could provide better metrics for uncertainty estimation. In this subsection, we compare the proposed DE baseline against our model.

In the Tables 8 and 9 below, we present the results of this comparison. While the DE baseline delivers good accuracy and uncertainty quality, it is approximately 14 times slower than the single model baseline and about 5 times slower than our approach. These results confirm that while DE applied to the iterative architecture provides high-quality uncertainty estimates and accuracy, the computational cost is significantly higher compared to our approach. Our method strikes a balance between computational efficiency and performance, making it a more practical choice for real-world applications where computational resources and time are critical factors.

	MC-DP	*Ens.*	*Ens. Iter.*	*Ours*
Corr	81.6	83.6	**93.9**	92.3
Comp	**92.3**	90.4	85.9	86.7
Qual	78.2	78.7	**81.3**	81.1
F1	13.6	14.1	14.8	**15.4**
APLS	59.65	67.53	**78.81**	78.04
rAULC	19.56	78.65	69.03	**79.27**
Corr (unc.)	32.50	76.39	76.51	**87.46**
ECE	0.558	0.794	0.746	**0.419**
ROC-AUC	61.25	67.03	**68.42**	67.09
PR-AUC	62.64	67.85	71.25	**72.11**
Train	**1x**	5x	14x	2.8x
Inf	5x	5x	13.5x	**2.7x**

Table 9. **Uncertainty and performance metrics on MS Dataset.** The best result in each category is in **bold** and the second best is in bold. Most correspond to Ours and Iterative Ensembles.

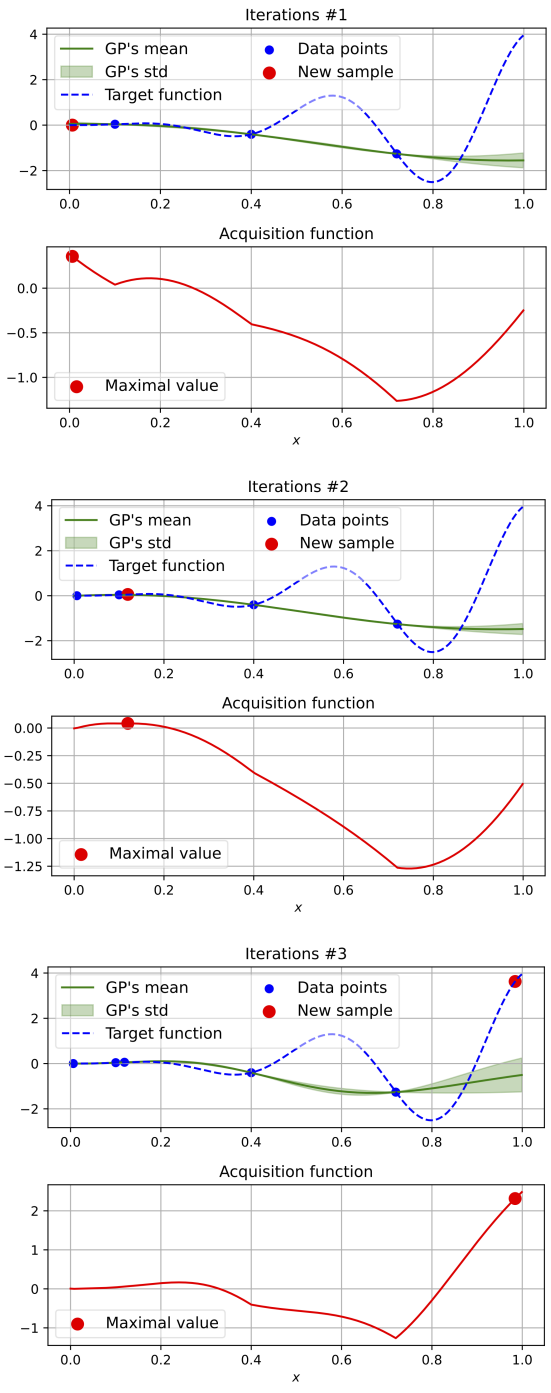


Figure 13. **Bayesian Optimization.** Given three initial data points for the function (dashed blue), we want to optimize, we train a GP surrogate model (Step 1) and compute the UCB acquisition function (Auer, 2002) over the $[0, 1]$ range (Steps 2-3). We then select the points that maximize, evaluate the target function at those points, and include the results in our training dataset (Step 4). The process is then iterated and, eventually, we find the true maximum of the function at $x \approx 1$, whereas a simple gradient based method would probably have remained trapped at the local maximum $x \approx 0.58$.