TRAINING-FREE UNCERTAINTY ESTIMATION FOR DENSE REGRESSION: SENSITIVITY AS A SURROGATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Uncertainty estimation is an essential step in the evaluation of the robustness for deep learning models in computer vision, especially when applied in risk-sensitive areas. However, most state-of-the-art deep learning models either fail to obtain uncertainty estimation or need significant modification (e.g., formulating a proper Bayesian treatment) to obtain it. Most previous methods are not able to take an arbitrary model off the shelf and generate uncertainty estimation without retraining or redesigning it. To address this gap, we perform a systematic exploration into training-free uncertainty estimation for dense regression, an unrecognized yet important problem, and provide a theoretical construction justifying such estimations. We propose three simple and scalable methods to analyze the variance of outputs from a trained network under tolerable perturbations: **infer-transformation**, **infer-noise**, and **infer-dropout**. They operate solely during inference, without the need to re-train, re-design, or fine-tune the model, as typically required by state-of-the-art uncertainty estimation methods. Surprisingly, even without involving such perturbations in training, our methods produce comparable or even better uncertainty estimation when compared to training-required state-of-the-art methods.

1 INTRODUCTION

Deep neural networks have achieved remarkable or even super-human performance in many tasks (Krizhevsky et al., 2012; He et al., 2015; Silver et al., 2016). While most previous work in the field has focused on improving accuracy in various tasks, in several risk-sensitive areas such as autonomous driving (Chen et al., 2015) and healthcare (Zhang et al., 2019), reliability and robustness are arguably more important and interesting than accuracy.

Recently, several novel approaches have been proposed to take into account an estimation of uncertainty during training and inference (Huang et al., 2018). Some use probabilistic formulations for neural networks (Graves, 2011; Hernández-Lobato & Adams, 2015; Wang et al., 2016; Shekhovtsov & Flach, 2018) and model the distribution over the parameters (weights) and/or the neurons. Such formulations naturally produce distributions over the possible outputs (Ilg et al., 2018; Yang et al., 2019). Others utilize the randomness induced during training and inference (e.g., dropout and ensembling) to obtain an uncertainty estimation (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Kendall et al., 2015).

All methods above require specific designs or a special training pipeline in order to involve the uncertainty estimation during training. Unfortunately, there are many cases where such premeditated designs or pipelines cannot be implemented. For example, if one wants to study the uncertainty of trained models released online, retraining is not always an option, especially when only a black-box model is provided or the training data is not available. Moreover, most models are deterministic and do not have stochasticity. A straightforward solution is to add dropout layers into proper locations and finetune the model (Gal & Ghahramani, 2016). However, this is impractical for many state-of-the-art and published models, especially those trained on large datasets (e.g. ImageNet (Deng et al., 2009)) with a vast amount of industrial computing resources. In addition, models that have already been distilled, pruned, or binarized fall short of fitting re-training (Han et al., 2015; Hou et al., 2016).

To fill this gap, we identify the problem of *training-free uncertainty estimation*: how to obtain an uncertainty estimation of any given model without re-designing, re-training, or fine-tuning it. We focus on two scenarios: black-box uncertainty estimation (BBUE), where one has access to the

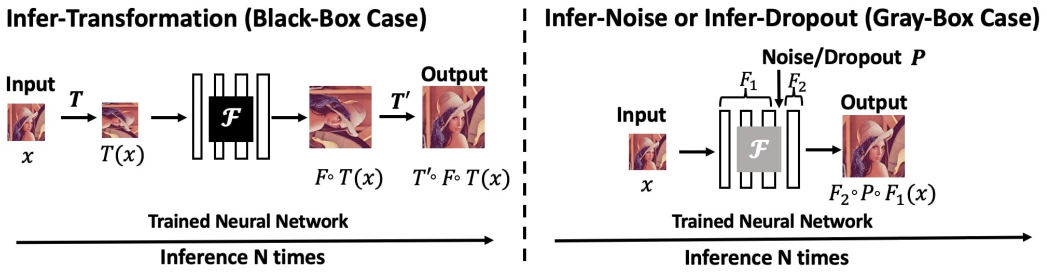

Figure 1: Method description of our training-free uncertainty estimation: apply infer-transformation T (**left**) and infer-noise or infer-dropout P (**right**) to a trained neural network F during inference.

model only as a black box, and gray-box uncertainty estimation (GBUE), where one has access to intermediate-layer neurons of the model (but not the parameters). Our work is a systematic exploration into this unrecognized yet important problem.

We propose a set of simple and scalable training-free methods to analyze the variance of the output from a trained network, shown in Fig. 1. Our main idea is to add a *tolerable* perturbation into inputs or feature maps during inference and use the variance of the output as a surrogate for uncertainty estimation. This is theoretically inspired by Chebyshev's inequality, which establishes the connection between the variance of the output and model sensitivity, as well as a simple triangle inequality that connects model sensitivity to prediction uncertainty (see Sec. 3.3 for details).

The first method, which we call *infer-transformation*, is to apply transformation that exploits the natural characteristics of a CNN – it is variant to input transformation such as rotation (Cohen & Welling, 2016). Transformations have been frequently used as data augmentation but rarely evaluated for uncertainty estimation. The second method, *infer-noise*, is to inject Gaussian noise with a zero-mean and a small standard deviation into intermediate-layer neurons. The third one, called *infer-dropout*, is to perform inference-time dropout in a chosen layer. Although at first blush infer-dropout is similar to MC-dropout, where dropout is performed during both training and inference in the same layers, they are different in several aspects: (1) Infer-dropout is involved *only during inference*. (2) Infer-dropout can be applied to arbitrary layers, even those without dropout training. Surprisingly, we find that even without involving dropout during training, infer-dropout is still comparable to, or even better than, MC-dropout for the purpose of uncertainty estimation.

In our paper, we focus on regression tasks. We note that for classification tasks, the softmax output is naturally a distribution. Methods that use entropy for uncertainty estimation qualify as a training-free method and have outperformed MC-Dropout (Bahat & Shakhnarovich, 2018; Gal & Ghahramani, 2016; Hendrycks & Gimpel, 2016; Wang et al., 2019) (See Appendix Sec. H for experiment results). Regression tasks are more challenging than classification problems since there is no output distribution. Our major contributions are:

1. We perform a systematic exploration of training-free uncertainty estimation for regression models and provide a theoretical construction justifying such estimations.
2. We propose simple and scalable methods, *infer-transformation*, *infer-noise* and *infer-dropout*, using a tolerable perturbation to effectively and efficiently estimate uncertainty.
3. Surprisingly, we find that our methods are able to generate uncertainty estimation comparable or even better than training-required baselines in real-world large-scale dense regression tasks.

2 Related Work

Probabilistic Neural Networks for Uncertainty Estimation. Probabilistic neural networks consider the input and model parameters as random variables which take effect as the source of stochasticity (Nix & Weigend, 1994; Welling & Teh, 2011; Graves, 2011; Hernández-Lobato & Adams, 2015; Wang et al., 2016). Traditional Bayesian neural networks model the distribution over the parameters (weights) (MacKay, 1992; Hinton & Van Camp, 1993; Graves, 2011; Welling & Teh, 2011) and obtain the output distribution by marginalizing out the parameters. Even with recent improvement (Balan et al., 2015; Hernández-Lobato & Adams, 2015), one major limitation is that the size of network at least doubles under this assumption, and the propagation with a distribution is usually computationally expensive. Another set of popular and efficient methods (Gal & Ghahramani, 2016; Teye et al., 2018) formulate dropout (Srivastava et al., 2014) or batch normalization (Ioffe & Szegedy, 2015) as approximations to Bayesian neural networks. For example, MC-dropout (Gal

& Ghahramani, 2016) injects dropout into some layers during both training and inference. Unlike most models that disable dropout during inference, MC-dropout feed-forwards the same example multiple times with dropout enabled, in order to form a distribution on the output. Meanwhile, other works (Wang et al., 2016; Shekhovtsov & Flach, 2018) propose sampling-free probabilistic neural networks as a lightweight Bayesian treatment for neural networks.

Non-probabilistic Neural Networks for Uncertainty Estimation. Other strategies such as deep ensemble (Lakshminarayanan et al., 2017; Huang et al., 2017a) train an ensemble of neural networks from scratch, where some randomness is induced during the training process, i.e. the initial weight is randomly sampled from a distribution. During inference, these networks will generate a distribution of the output. Though simple and effective, training multiple networks costs even more time and memory than Bayesian neural networks. Another efficient method log likelihood maximization (LLM) is to train the network to have both original outputs and uncertainty predictions, by jointly optimizing both (Zhang et al., 2019; Poggi et al., 2020). However, such a design requires re-training, introduces heavy implementation overhead, and sometimes makes the optimization process more challenging.

3 METHODOLOGY

Three Cases on Parameter Accessibility. We distinguish among three cases based on accessibility of the original model. 1. Black-box case: the model is given as a trained black box without any access to its internal structure. 2. Gray-box case: the internal representations (feature maps) of the model is accessible (while the parameters are not) and can be modified during inference. 3. White-box case: the model is available for all modifications (e.g. its weights can be modified, which requires training). In this paper we focus on the black-box and gray-box cases, for which we offer, correspondingly, two classes of methods. For the black-box case, we propose *infer-transformation*, which exploits the model's dependence on input transformations, e.g. rotations/flips. For the grey-box case, we propose *infer-noise* and *infer-dropout*, which introduce an internal embedding/representation manipulation - injecting a noise layer or a dropout layer during inference. These three methods are illustrated in Fig. 1. The description of our methods and a theoretical construction are presented as below.

3.1 BLACK-BOX UNCERTAINTY ESTIMATION: INFER-TRANSFORMATION

Given a black-box model, we explore the behavior of the outputs for different transformed versions of the input. Specifically, we transform the input with tolerable perturbations, e.g. perturbations that do not cause significant increase in the loss (see Sec. 3.3 for details), and then use the variance of the perturbed outputs as estimated uncertainty. Here we focus on transformations that preserve pertinent characteristics of the input, such as rotations, flips, etc. Formally, given an input image X, our measured uncertainty is defined as $\mathbb{V}[Z] = \mathbb{V}_T[T' \circ F \circ T(X)]$, where $T \in \mathcal{T}$ is a transformation, T' is T's inverse operation, and F is a function representing the black-box neural network. $Z = T' \circ F \circ T(X)$ is a sample from the perturbed output distribution. Note that it is possible to sample $Z = F(X)$, where P happens to be a 360-degree rotation.

3.2 GRAY-BOX UNCERTAINTY ESTIMATION: INFER-NOISE AND INFER-DROPOUT

Given a gray-box model, we consider another class of methods for generating multiple outputs from a distribution: randomly perturbing latent codes. Compared with the black-box case, this provides finer granularity on modulating the perturbation strength to ensure tolerability. Specifically we propose *infer-noise*, which introduces Gaussian noise at an intermediate layer of the trained model, and *infer-dropout*, which uses dropout instead. For infer-noise, the noise will be added to the feature maps of a certain layer. This noise is randomly sampled multiple times during inference to form a set of diverse outputs. For infer-dropout, random dropout is performed for multiple forwards to generate output samples, the variance of which are then used as uncertainty estimation. Formally, given an input image X, our measured uncertainty is defined as $\mathbb{V}[Z] = \mathbb{V}_P[F_2 \circ P \circ F_1(X)]$, where P is sampled from a perturbation set $\mathcal{P}$ (e.g. Gaussian noise with $\sigma = 1$). F_1 is the function of network layers before the perturbation P, F_2 represents network layers after P, and $F_2 \circ F_1(X)$ is the gray-box network $F(X)$. $Z = F_2 \circ P \circ F_1(X)$ is a sample from the perturbed output distribution. Note that it is possible to sample $Z = F(X)$, where P happens to be a perturbation noise of all zeros.

3.3 SENSITIVITY AS A SURROGATE MEASURE

The idea at the core of our approach is to impose *tolerable perturbations* on the original trained model's input or intermediate representations (feature maps). Given a perturbation output sample Z and ground truth Y, the perturbation tolerability is defined using $C = |\mathbb{E}[Z] - Y| \leq \epsilon$, where ϵ denotes the perturbation error threshold; *smaller ϵ indicates better tolerability*. Such perturbations generate a sensitivity map $\mathbb{V}[Z]$ as a surrogate measure of the model's uncertainty. Note that our method involves *no sacrifice* in the predictive performance of original model. One can use our method to produce uncertainty estimation while still use the original model to make predictions.

Lemma 3.1 (Chebyshev's Inequality)**.** *Let Z be any random variable with variance $\mathbb{V}[Z] < \infty$. Then for a constant margin $t \geq 0$, $\mathbb{P}(|Z - \mathbb{E}[Z]| \geq t) \leq \mathbb{V}[Z]/t^2$.*

Variance and Sensitivity. If Z is the model prediction, the probability $\mathbb{P}(|Z - \mathbb{E}[Z]| \geq t)$ translates to 'how possible the model prediction Z deviates from $\mathbb{E}[Z]$ by a margin larger than t' and therefore measures the 'model sensitivity'. Such sensitivity is bounded by a scaled variance $\mathbb{V}[Z]/t^2$ (where t^2 is a constant). *Therefore, Lemma 3.1 connects our output variance $\mathbb{V}[Z]$ to model sensitivity.*

Theorem 3.1. *Let Y be the ground truth and Z be a random variable representing the model prediction, where randomness comes from our perturbation. We have that*

$$\mathbb{P}(|Z - Y| \geq t) \leq \mathbb{V}[Z]/(t - C)^2, \tag{1}$$

for any constant margin $t \geq C$, where $C = |\mathbb{E}[Z] - Y|$ is the prediction error for $\mathbb{E}[Z]$.

Variance and Uncertainty. Similar to Lemma 3.1, $\mathbb{P}(|Z - Y| \geq t)$ in Theorem 3.1 translates to 'how possible the model prediction Z deviates from the ground truth Y by a margin larger than t' and therefore measures the 'uncertainty'. *Theorem 3.1 establishes $\mathbb{V}[Z]/(t - C)^2$ as an upper bound for the 'uncertainty'.* In practice, we directly use $\mathbb{V}[Z]$ since C is the unknown ground-truth prediction error. In Sec. 4, we empirically show that $\mathbb{V}[Z]$ as a rough approximation for $\mathbb{V}[Z]/(t - C)^2$ can already obtain uncertainty estimation on par with or even better than state-of-the-art baselines.

Why We Need Tolerability. (1) Wider valid region: The upper bound in Eqn. 1 is valid only when $t \geq C$; therefore since $C \leq \epsilon$ by definition, better tolerability (i.e. lower ϵ) leads to smaller C, giving the bound a wider valid region w.r.t. the margin t. (2) Tighter bound: Better tolerability also guarantees smaller C and $\mathbb{V}[Z]$, making $\mathbb{V}[Z]/(t - C)^2$ smaller and consequently a tighter bound given a constant margin t (see Sec. 4.3 and Fig. 5 for details).

4 EXPERIMENTS

In this section, we evaluate our three proposed approaches in two representative real-world large-scale dense regression tasks, super resolution and depth estimation.

4.1 SINGLE IMAGE SUPER RESOLUTION

The task of Single Image Super Resolution (SR) is to reconstruct a high-resolution (HR) image from a low-resolution (LR) input. Here we focus on analyzing the state-of-the-art SRGAN model (Ledig et al., 2017), which can restore photo-realistic high-quality images. SRGAN always outputs deterministic restorations since the conditional GAN (Mirza & Osindero, 2014) used in this model involves no latent variable sampling. However, we can still evaluate its uncertainty with our proposed methods.

We apply our methods to estimate uncertainty in one open-source version of this work (Dong et al., 2017). The package provides two models trained with different loss functions: 1) SRresnet model with L_2 loss and 2) SRGAN model with a combination of L_2 loss and adversarial loss. We evaluate our methods on both models in the black-box/gray-box settings.

Infer-Transformation. For infer-transformation, we apply rotation of $K \times 90$ degrees ($K = 0, 1, 2, 3$) as well as horizontal flip to the LR input, feed it into the trained model during the inference, and apply the inverse transformation to its output. We could generate at most 8 samples using this strategy, and then calculate the pixel-wise variance.

Infer-Noise. In infer-noise, we take the trained model and add a Gaussian-noise layer, which has standard deviation $\sigma \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ and mean 0, at different locations (layers). We choose 4 different locations for noise injection, including the layers right after the input and some

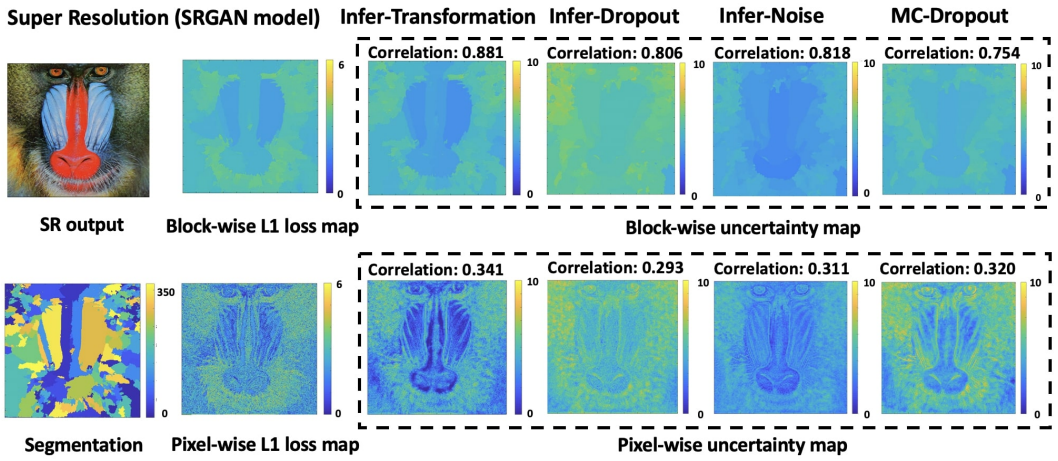

Figure 2: Visualization of block-wise and pixel-wise uncertainty (variance) maps (log scale) generated by infer-transformation, infer-dropout, MC-dropout (Gal & Ghahramani, 2016), using SRGAN (Ledig et al., 2017) for the super resolution task. L_1 loss map (log scale) is also provided for comparison. Correlation between the L_1 loss map and the uncertainty map is also presented.

intermediate layers (see details in Appendix Sec. C). For each experiment, we only add the noise into one layer with a specific σ value. Sample numbers of 8 and 32 are evaluated.

Infer-Dropout. In infer-dropout, we take the trained model and add a dropout layer with varied dropout rates. We choose the dropout rate ρ from the set $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ and use the same set of locations as the infer-noise. For each experiment, we only add the layer into one location with one specific dropout rate. Sample numbers of 8 and 32 are evaluated.

Baselines. We compare our methods with three training-required baselines. The first baseline is MC-dropout (Gal & Ghahramani, 2016) with a dropout rate $\rho \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$. For each experiment, we add dropout layer only into one location with one dropout rate during training. The same dropout rate is used for sampling during inference. We try different sample numbers of 8 and 32. The second baseline is deep ensemble (Lakshminarayanan et al., 2017). We follow Lakshminarayanan et al. (2017) to train ensembles as 4 and 8 networks, respectively. We train these networks with the same number of epochs until they converge. During inference, each of them generates a single deterministic output, with 4 or 8 samples generated in total. The third baseline is a sampling-free method log likelihood maximization (LLM) (Zhang et al., 2019; Poggi et al., 2020), where a network is trained to predict a output distribution with log likelihood maximization.

4.2 Monocular Depth Estimation

For depth estimation (Postels et al., 2019; Kendall & Gal, 2017), we use one of the commonly applied models based on fully convolutional residual network (FCRN) (Laina et al., 2016). We directly use the trained model released by the original author; this is consistent with the scenarios of black-box and gray-box cases, since the code for training is not released.

We evaluate the model on NYU Depth Dataset V2. For infer-transformation, we avoid applying 90-degree rotation to input, since the orientation is a strong prior to predict depth which can violate the tolerability, and only apply horizontal flip to generate 2 samples for uncertainty estimation. For infer-dropout, we choose two locations (intermediate layers) to add the dropout layer. For infer-noise, we choose three locations to add the noise layer (two intermediate layers and one layer before the final FC layer). Then we conduct similar experiments as described in the SR task. For the baseline MC-dropout, note that the model has a dropout layer before the final fully connected (FC) layer during training, we directly perform sampling from the existing dropout layer. Sample numbers of 2 and 8 are evaluated for both infer-dropout and infer-noise (see details in Appendix Sec. C).

4.3 Experiment Results

Evaluation Metrics. Commonly used metrics to evaluate uncertainty estimation include Brier score (BS), expected calibration error (ECE), and negative log-likelihood (NLL) (Lakshminarayanan et al., 2017; Guo et al., 2017). However, BS and ECE are for classification tasks only and hence not

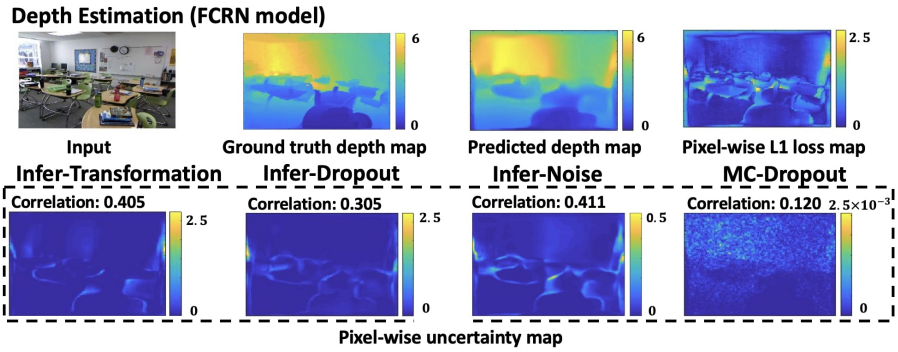

Figure 3: Visualization of pixel-wise uncertainty (variance) maps from infer-transformation, infer-dropout, MC-dropout (Gal & Ghahramani, 2016) compared with the L_1 loss map in depth estimation task. Correlation between the L_1 loss map and the uncertainty map is also presented.

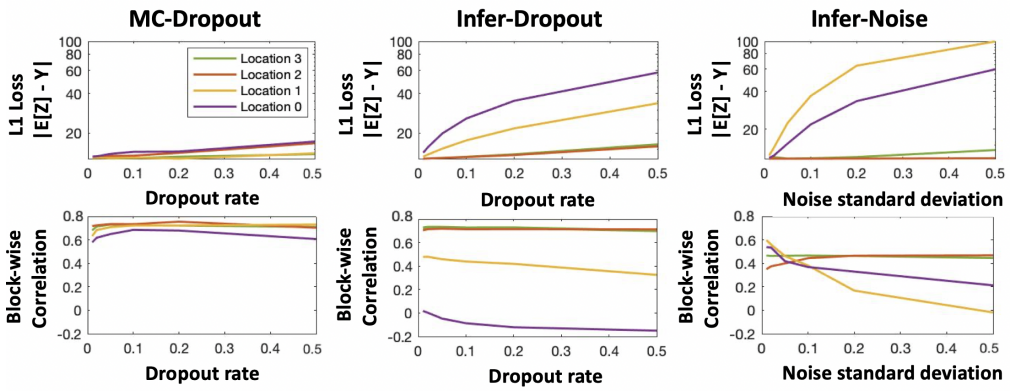

Figure 4: **Top**: L_1 loss of perturbed model for MC-dropout (Gal & Ghahramani, 2016), infer-dropout and infer-noise. Various dropout rates and noise levels have been evaluated. Location 0 is right after the input; location 1, 2, 3 are intermediate layers. **Bottom**: Correlation between error and variance with different locations and perturbation strength. For infer-dropout, note that location 2 and 3 cause minimal increase in the L_1 loss after perturbation (i.e. high tolerability), leading to high correlation.

applicable in our setting. We therefore use the following metrics for evaluations: (1) **NLL**, which is defined in regression tasks by assuming a Gaussian distribution. However, note that NLL depends on not only the quality of uncertainty estimation but also the prediction accuracy itself. Therefore contrary to previous belief, we argue that it is not an ideal metric for evaluating uncertainty estimation. (2) **Area Under the Sparsification Error (AUSE)**, which quantifies how much uncertainty estimation coincides with the true errors (Ilg et al., 2018). (3) **Correlation** between the estimated uncertainty and the error. Here we define four variants of correlation (see details in Appendix Sec. E): *pixel-wise*, *mean*, *block-wise*, and *patch-wise* correlations to evaluate performance at the pixel, image, block, and patch levels, repsectively. The intuition is that in many situations it is more instructive and meaningful when uncertainty is visualized in each region (e.g. a region with a possible tumor for a medical imaging application). Note that block-wise correlation depends on specific segmentation algorithms, while patch-wise correlation defines regions in an algorithm-independent way. Similarly we also define four evaluation forms for AUSE.

For our training-free methods, these metrics are computed between uncertainty and the error from the *original* model (without perturbation), because we will still use the original model for prediction. For training-required methods such as MC-dropout (i.e. MC-drop^2 in Table 1) (Gal & Ghahramani, 2016), deep ensemble (Lakshminarayanan et al., 2017) and log likelihood maximization (LLM) (Zhang et al., 2019; Poggi et al., 2020), the mean of output samples are used as prediction. Meanwhile, we also evaluate another MC-dropout variant, denoted as MC-drop^1, where the output of the original model is used as prediction, to be consistent with training-free methods. The evaluation results using metrics described above are shown in Table 1 and Table 2. The evaluations between uncertainty and L_2 loss are shown in Appendix Sec. F.

Qualitative Results. Fig. 2 shows some qualitative results for an example image in the SR task. We can see that the variance maps generated in our task are consistent to the level of ambiguity.

SRGAN model: Super Resolution														
Condition		Training Free (Ours)						Training Required						
Prediction		Original Output								Outputs Mean				
Method		Infer-trans		Infer-drop		Infer-noise		MC-drop^1		MC-drop^2		Ensemble		LLM
Samples		4	8	8	32	8	32	8	32	8	32	4	8	0
AUSE	mean L_1	**0.006**	**0.006**	0.013	0.013	0.016	0.016	0.008	0.009	0.007	0.007	0.018	0.020	0.035
	patch L_1	**0.021**	0.022	0.029	0.030	0.040	0.039	0.029	0.029	0.025	0.025	0.033	0.033	0.044
	block L_1	**0.023**	**0.023**	0.032	0.031	0.044	0.043	0.030	0.029	0.028	0.028	0.033	0.033	0.042
	pixel L_1	0.128	**0.121**	0.154	0.141	0.173	0.162	0.141	0.131	0.137	0.129	0.152	0.144	0.137
Corr	mean L_1	0.931	0.930	0.884	0.882	0.774	0.780	0.942	**0.943**	0.938	0.936	0.692	0.694	0.484
	patch L_1	0.765	**0.770**	0.722	0.731	0.590	0.598	0.748	0.755	0.734	0.741	0.674	0.677	0.565
	block L_1	0.757	**0.767**	0.717	0.730	0.579	0.592	0.735	0.747	0.698	0.710	0.651	0.664	0.588
	pixel L_1	0.367	**0.394**	0.323	0.376	0.245	0.288	0.339	0.390	0.330	0.379	0.290	0.326	0.393
NLL		17.910	9.332	4.889	4.804	4.899	4.791	6.804	6.013	6.365	5.541	11.520	5.994	**1.320**
SRresnet model: Super Resolution														
Condition		Training Free (Ours)						Training Required						
Prediction		Original Output								Outputs Mean				
Method		Infer-trans		Infer-drop		Infer-noise		MC-drop^1		MC-drop^2		Ensemble		LLM
Samples		4	8	8	32	8	32	8	32	8	32	4	8	0
AUSE	mean L_1	0.044	0.044	0.042	0.043	0.067	0.066	0.041	0.047	**0.036**	0.039	0.073	0.066	**0.036**
	patch L_1	0.045	0.044	0.048	0.047	0.055	0.055	0.046	0.045	0.040	0.041	0.066	0.066	**0.037**
	block L_1	0.047	0.045	0.049	0.048	0.062	0.061	0.048	0.046	0.041	0.042	0.073	0.070	**0.031**
	pixel L_1	0.164	0.153	0.165	0.155	0.190	0.181	0.163	0.152	0.150	0.144	0.194	0.185	**0.133**
Corr	mean L_1	0.340	0.359	0.401	0.404	0.056	0.055	0.408	0.379	0.527	0.512	0.016	0.048	**0.622**
	patch L_1	0.501	0.520	0.508	0.518	0.371	0.385	0.535	0.545	0.547	0.542	0.323	0.361	**0.648**
	block L_1	0.462	0.486	0.498	0.509	0.358	0.370	0.505	0.521	0.531	0.529	0.274	0.286	**0.673**
	pixel L_1	0.237	0.269	0.258	0.303	0.172	0.216	0.264	0.309	0.288	0.322	0.184	0.206	**0.393**
NLL		107.243	43.071	5.155	4.955	4.941	4.788	8.430	7.221	8.018	6.688	13.559	7.906	**1.422**

Table 1: Mean/patch-wise/block-wise/pixel-wise AUSE and correlation between L_1 loss and uncertainty, and NLL on SR benchmark dataset Set 14. Our infer-transformation, infer-dropout and infer-noise are compared with MC-dropout (Gal & Ghahramani, 2016), deep ensemble (Lakshminarayanan et al., 2017) and log likelihood maximization (LLM) (Zhang et al., 2019). MC-drop^1 uses the output of the original model as prediction while MC-drop^2 uses the mean of output samples from the re-trained model (with added dropout) as prediction. Models evaluated: SRGAN and SRresnet.

FCRN model: Depth Estimation										
Condition		Training Free (Ours)					Training Required			
Prediction		Original Output							Outputs Mean	
Method		Infer-trans		Infer-drop		Infer-noise	MC-drop^1		MC-drop^2	
Samples		2	2	8	2	8	2	8	2	8
AUSE	mean L_1	0.051	0.044	**0.041**	0.046	**0.041**	0.062	0.062	0.060	0.062
	patch L_1	0.106	0.109	0.092	0.108	**0.091**	0.127	0.126	0.122	0.125
	block L_1	0.056	0.057	0.047	0.053	**0.045**	0.065	0.065	0.063	0.065
	pixel L_1	0.165	0.168	0.135	0.167	**0.134**	0.208	0.193	0.207	0.193
Corr	mean L_1	0.596	0.630	0.677	0.651	**0.708**	0.473	0.471	0.469	0.469
	patch L_1	0.324	0.306	0.409	0.312	**0.411**	0.258	0.268	0.266	0.269
	block L_1	0.354	0.354	**0.449**	0.364	0.447	0.215	0.220	0.220	0.221
	pixel L_1	0.208	0.182	0.284	0.188	**0.288**	0.075	0.134	0.076	0.134
NLL		8.634	8.443	4.889	3.526	**1.006**	12.365	9.842	12.503	9.866

Table 2: Mean/patch-wise/block-wise/pixel-wise AUSE and correlation between L_1 loss and uncertainty, and NLL on NYU Depth Dataset V2. Our infer-transformation, infer-dropout and infer-noise are compared with MC-dropout (Gal & Ghahramani, 2016). MC-drop^1 uses the output of the original model as prediction while MC-drop^2 uses the mean of output samples from the re-trained model (with added dropout) as prediction. Models evaluated: FCRN model.

Specifically, in our methods, high variance occurs in areas with high randomness and high frequency. For the depth estimation task shown in Fig. 3, high variance usually occurs in the area with high spatial resolution and large depth. As expected, these high-variance areas usually correspond to large prediction error.

The Role of Tolerable Perturbations. Tolerable perturbations play a crucial role in obtaining effective uncertainty estimation. Better tolerability means smaller decrease in accuracy after perturbation (i.e. smaller ϵ in Sec. 3.3). Fig. 4 shows the correlation for different amount of perturbations (noise or dropout) in different locations, and the corresponding predictive performance $C = |\mathbb{E}[Z] - Y|$ (evaluated as L_1 loss) *after perturbations*. As we can see, the optimal cases to generate uncertainty maps with high correlation require that C should remain small after perturbation (high tolerability). Interestingly, different methods have different ways of achieving high tolerability: (1) For MC-dropout, involving dropout during training increases the robustness of model against perturbations, keeping the loss relatively small after adding dropout layer in most locations during inference; (2) for infer-dropout, adding dropout layer in intermediate locations (i.e. location 2 and location 3) where

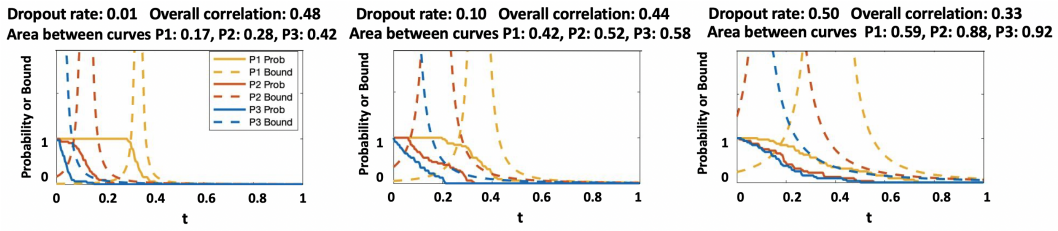

Figure 5: Visualization of probability $\mathbb{P}(|Z-Y| \geq t)$ and the upper bound $\mathbb{V}[Z]/(t-C)^2$ (only valid when $t \geq C$). The increasing area between curves (looser bound) is caused by the larger perturbation strength (dropout rate), which violates the condition of tolerability (i.e. small C).

Method		MC-drop[1]	MC-drop[2]	Ensemble	LLM	Infer-trans		Infer-drop		Infer-noise	
		Baselines				Proposed	PB	Proposed	PB	Proposed	PB
pixel L_1	AUSE	0.131	0.129	0.144	0.137	<u>0.121</u>	0.097	0.141	**0.085**	0.162	0.106
	Corr	0.390	0.379	0.326	0.393	<u>0.394</u>	0.424	0.376	**0.590**	0.288	0.583

Table 3: Comparison of our methods and baselines as well as performance bound (PB). We mark the best performance bound in **bold face** and the best method by <u>underlining</u>. Model evaluated: SRGAN.

the information is the most redundant (He et al., 2014), can effectively alleviate disturbance; (3) for infer-noise, adding noise with small standard deviation effectively limits the perturbation level. More interestingly, we further find that for both MC-dropout and infer-dropout, adding perturbation in intermediate layers are usually the optimal choices for uncertainty estimation. Applying infer-dropout in these intermediate layers, we could achieve comparable or even better correlation compared to training-required baselines. For infer-noise, locations do not have similar effect; one can therefore further tune the noise strength σ to achieve higher correlation. The conclusion above is also consistent with the evaluation of other models in Appendix Sec. D.

Comparable Performance with Training-required Baselines. We report correlation, AUSE and NLL using the optimal hyper-parameters in different methods in Table 1 and Table 2. As depicted in both tasks, based on these metrics, our methods infer-transformation, infer-dropout and infer-noise are able to provide comparable or even better results than the training-required state-of-the-art baselines. Even a small number of samples are able to guarantee sufficient quality. For the super-resolution task, we find infer-transformation achieves the highest performance and even outperforms training-required baselines. Note that SRGAN has a higher correlation than the SRresnet model. For depth estimation, we find that using infer-noise in the intermediate layers outperforms other methods. For baseline MC-dropout, we perturb right before the last convolutional layer – the only dropout layer during the original model training, and therefore produce a highly localized variance map with poor correlation, shown in Fig. 3. If we are allowed to perform MC-dropout in intermediate layers and re-train the model, a correlation value comparable to that of infer-dropout should be expected.

Theorem 3.1 and Performance Bound. Theorem 3.1 establishes $\mathbb{V}[Z]/(t-C)^2$ as an upper bound for the 'uncertainty' $\mathbb{P}(|Z-Y| \geq t)$. Correspondingly, Fig. 5 shows $\mathbb{V}[Z]/(t-C)^2$ (dashed lines) and $\mathbb{P}(|Z-Y| \geq t)$ (solid lines) versus t for three representative pixels, empirically verifying the validity of $\mathbb{V}[Z]/(t-C)^2$ as the uncertainty's upper bound. Note that as the perturbation strength (i.e. dropout rate) gets larger, both $\mathbb{V}[Z]$ and C increase and consequently loosen the bound (as mentioned in Sec. 3.3); reflected in Fig. 5, we can see the area between the probability curve and the bound curve of each pixel also gets larger from Fig. 5 (left) to Fig. 5 (right), which leads to the worse performance (lower average pixel-wise correlation). This highlights the need for *tolerable* perturbations (i.e. low ϵ and C in Sec. 3.3). Table 3 shows correlation and AUSE for different methods compared to the oracle (performance bound) $\mathbb{V}[Z]/(t-C)^2$. Our methods, especially infer-transformation, are reasonably close to the oracle and compare favorably with baselines. More details are in Appendix Sec. G.

5 Conclusion

In this work, we perform a systematic exploration into training-free uncertainty estimation for dense regression, an unrecognized yet important problem, and provide a theoretical construction justifying such estimations. We propose three simple, scalable, and effective methods, i.e. *infer-transformation*, *infer-noise*, and *infer-dropout*, for uncertainty estimation in both black-box and gray-box cases. Surprisingly, our training-free methods achieve comparable or even better results compared to training-required state-of-the-art methods. Furthermore, we demonstrate adding tolerable perturbations is the key to generating high quality uncertainty maps for all methods we studied.

REFERENCES

Iman Aganj, Mukesh G Harisinghani, Ralph Weissleder, and Bruce Fischl. Unsupervised medical image segmentation based on the local center of mass. *Scientific reports*, 8(1):13012, 2018.

Yuval Bahat and Gregory Shakhnarovich. Confidence from invariance to image transformations. *arXiv preprint arXiv:1804.00657*, 2018.

Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *NIPS*, 2015.

Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, pp. 2722–2730, 2015.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, pp. 2990–2999, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, and Yike Guo. TensorLayer: A Versatile Library for Efficient Deep Learning Development. *ACM Multimedia*, 2017. URL `http://tensorlayer.org`.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pp. 1050–1059, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *ICML*, pp. 1183–1192, 2017.

Alex Graves. Practical variational inference for neural networks. In *NIPS*, 2011.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, pp. 1321–1330, 2017.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu. Reshaping deep neural network for fast decoding by node-pruning. In *ICASSP*, pp. 245–249, 2014.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *COLT*, 1993.

Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. 2016.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017a.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017b.

Po-Yu Huang, Wan-Ting Hsu, Chun-Yueh Chiu, Ting-Fan Wu, and Min Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 520–535, 2018.

Eddy Ilg, Özgün **c**Ci**c**cek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*, pp. 677–693, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, pp. 5574–5584, 2017.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1097–1105, 2012.

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, pp. 239–248, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, pp. 6402–6413, 2017.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pp. 4681–4690, 2017.

David MacKay. A practical Bayesian framework for backprop networks. *Neural computation*, 1992.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pp. 55–60. IEEE, 1994.

Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3227–3237, 2020.

Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. *arXiv preprint arXiv:1908.00598*, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pp. 234–241, 2015.

Alexander Shekhovtsov and Boris Flach. Feed-forward propagation in probabilistic neural networks with categorical and max layers. In *ICLR*, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. *arXiv preprint arXiv:1802.06455*, 2018.

Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019.

Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Natural-parameter networks: A class of probabilistic neural networks. In *NIPS*, pp. 118–126, 2016.

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

Gengshan Yang, Peiyun Hu, and Deva Ramanan. Inferring distributions over depth from a single image. *arXiv preprint arXiv:1912.06268*, 2019.

Zizhao Zhang, Adriana Romero, Matthew J Muckley, Pascal Vincent, Lin Yang, and Michal Drozdzal. Reducing uncertainty in undersampled mri reconstruction with active acquisition. In *CVPR*, pp. 2049–2058, 2019.

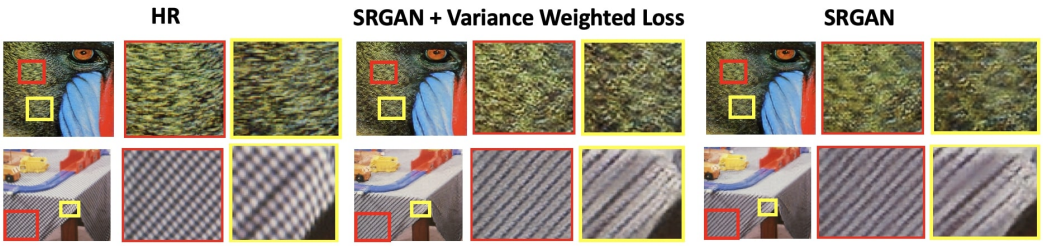

Figure 6: The first application using uncertainty map estimated in our methods to improve the quality of SR results. We compare SR results that use L_2 loss re-weighted by variance map **(middle)** and that do not **(right)**. HR **(left)** represents high resolution image. Results evaluated in Set 14.

Method	SSIM	PSNR	L_1	Method	SSIM	PSNR	L_1	Method	SSIM	PSNR	L_1
original	0.772	23.841	11.623	random	0.773	23.616	12.344	high uncertainty	**0.790**	**23.942**	**11.497**

Table 4: The second application of active learning on our generated uncertainty maps to improve the performance with more efficiency. Here we select samples with high uncertainty yields better results than select randomly. Results evaluated in Set 14 on SSIM, PSNR and L_1 loss.

A Proof of Theorem 3.1.

Proof. Replacing t in Lemma 3.1 with $(t - C)$ where $t \geq C$, we have $\mathbb{P}(|Z - \mathbb{E}[Z]| + C \geq t) \leq \mathbb{V}[Z]/(t - C)^2$, meaning that $\mathbb{P}(|Z - \mathbb{E}[Z]| + C < t) > \mathbb{V}[Z]/(t - C)^2$. Notice the triangle inequality:

$$|Z - Y| \leq |Z - \mathbb{E}[Z]| + |\mathbb{E}[Z] - Y| = |Z - \mathbb{E}[Z]| + C.$$

We therefore have $\mathbb{P}(|Z - Y| < t) > \mathbb{V}[Z]/(t - C)^2$, which is equivalent to $\mathbb{P}(|Z - Y| \geq t) \leq \mathbb{V}[Z]/(t - C)^2$, completing the proof. □

B Applications Benefit from Uncertainty Estimation

The first application is to improve the quality of SR results. We propose a novel and efficient method which takes the pixel-wise uncertainty map as a weight term for the regression loss, while keeps the original adversarial loss, which could provide a more photo-realistic SR output with finer structures and sharper edges, shown in Fig. 6. Another application is active learning (Gal et al., 2017), which aims to use uncertainty to guide annotations, then only a small subset of data are required to improve training. We find active learning based on our generated uncertainty maps can improve the performance with more efficiency, shown in Table 4.

C Details on Network Noise/Dropout Injection Locations

To perform uncertainty estimation using infer-noise and infer-dropout and baseline MC-dropout on both SRGAN model and SRresnet model, we choose 4 different locations for noise injection, including the layers right after the input, as well as some intermediate layers, as shown in Fig. 7.

To perform uncertainty estimation using infer-noise and infer-dropout on FCRN model, we choose 3 different locations for noise injection, including the layers right before the output, as well as some intermediate layers, as shown in Fig. 8. For baseline MC-dropout, we choose location 3 where the dropout layer added during training for the original model.

D Tolerability on SRResnet Model and FCRN Model

Fig. 9 shows the perturbation on the SRresnet model, and Fig. 10 shows the perturbation on the FCRN model for depth estimation. The correlation for different amount of perturbations (noise or dropout) in different locations, and the corresponding predictive performance (evaluated as L_1 loss) *after perturbations*. For SRresnet mode, We find that for both MC-dropout and infer-dropout, adding

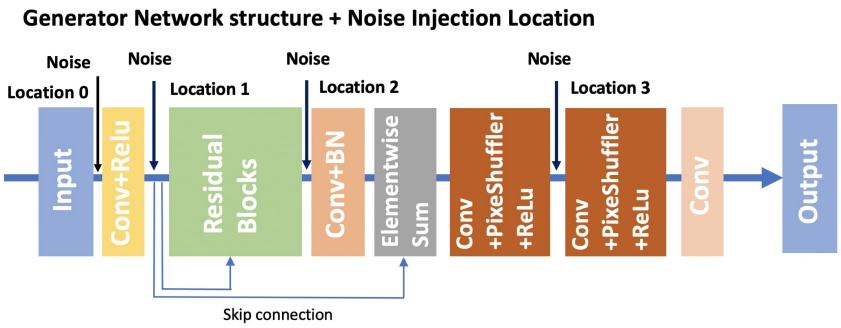

Figure 7: Different locations for infer-noise and infer-dropout in SRGAN and SRresnet for super resolution. For each experiment, the noise or dropout is injected at a single location with one perturbation level.

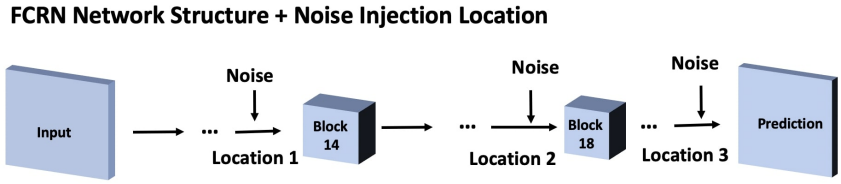

Figure 8: Different locations for noise or dropout injection in the FCRN model for depth estimation.

perturbation in intermediate layers are usually the optimal choices for uncertainty estimation. For infer-noise, locations do not have similar effect; one can therefore further tune the noise strength σ to achieve higher correlation. For FCRN model, using infer-dropout or infer-noise, intermediate layers are also usually the optimal choices for uncertainty estimation. The results are consistent with the results of SRGAN model.

E Details on Evaluation Metrics

In this section, we provide the details on our proposed correlation-based evaluation metrics. Assuming we have N outputs given the same input x from our infer-transformation, infer-drop and infer-noise, each output is represented by Y_w. Given the output image with the size of $H \times W$, the error we define for regression task is pixel-wise L_1 loss and L_2 loss, represented by $L_{1,ij}$ and $L_{2,ij}$, where i, j is the corresponding coordinates of the pixel P_{ij} in the output image. The uncertainty (variance) estimated in these methods is also a pixel-wise value, represented by $V_{ij} = \frac{\sum_{w=1}^{N}(Y_{w,ij} - \overline{Y}_{ij})^2}{N}$. The pixel-wise L_1 correlation is defined as $corr(\{V_{ij}\}, \{L_{1,ij}\})$. The second metric is mean correlation, the mean L_1 error $\overline{L}_{1,z} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} L_{1,ij}}{W \times H}$ is defined as the average error of a single image z, correspondingly, the mean variance is defined as $\overline{V}_z = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} V_{ij}}{W \times H}$, the mean L_1 correlation is defined as $corr(\{\overline{V}_z\}, \{\overline{L}_{1,z}\})$. This metric has been used in (Zhang et al., 2019). The third metric for evaluation is the block-wise correlation – a new metric we propose in this work. To compute block-wise correlation, we need to firstly apply a local segmentation algorithm to the output of the trained model to cluster pixels with similar low-level context. Here we use the local-center-of-mass approach (Aganj et al., 2018) to perform segmentation. We denote each cluster as C_i. The variance of K_{C_i} pixels inside each cluster (block) C_i is then averaged and replaced with the mean value $\widetilde{V}_i = \frac{\sum_{P_{ij} \in C_i}^{K_{C_i}} V_{ij}}{K_{C_i}}$. The block-wise L_1 loss $\widetilde{L}_{1,i}$ can be calculated similarly. After that, we calculate the pixel-wise correlation of each pixels with the updated value as the L_1 block-wise correlation $corr(\{\widetilde{V}_i\}, \{\widetilde{L}_{1,i}\})$. For the fourth metric, patch-wise correlation, where the segmentation clusters in block-wise correlation are replaced by patches. In our analysis, each image is divided into 10×10 patches. And then the patch-wise correlation is calculated with following the same rule as block-wise

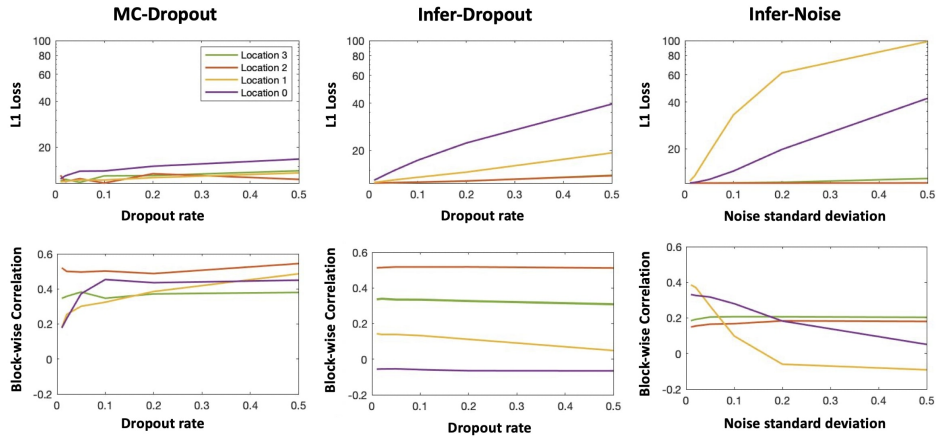

Figure 9: Evaluation on SRresnet model for super resolution. **Top**: The L_1 loss of perturbed model for infer-dropout and infer-noise and baseline MC-dropout (Gal & Ghahramani, 2016). Various dropout rates and noise levels have been evaluated. Location 0 is right after the input; location 1, 2, 3 are intermediate layers. **Bottom**: Correlation between error and variance with different locations and perturbation strength. Note that location 2 cause minimal increase in the L_1 loss after perturbation (i.e., high tolerability), leading to high correlation for MC-dropout and infer-dropout.

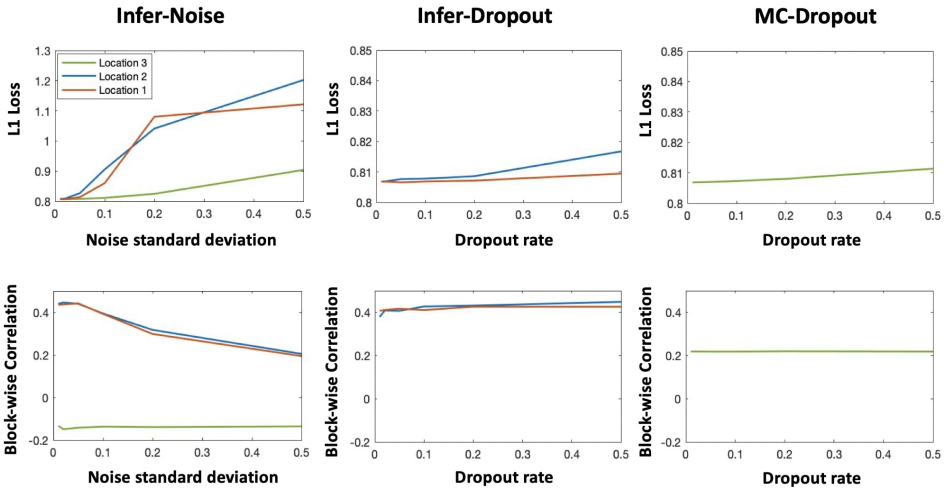

Figure 10: Evaluation on FCRN model for depth estimation. **Top**: The L_1 loss of perturbed model for infer-dropout and infer-noise and baseline MC-dropout (Gal & Ghahramani, 2016). Various dropout rates and noise levels have been evaluated. Location 1, 2 are intermediate layers, location 3 is right before the last convolutional layer. **Bottom**: Correlation between error and variance with different locations and perturbation strength. Note that location 1 and 2 cause minimal increase in the L_1 loss after perturbation (i.e., high tolerability), leading to high correlation for infer-dropout.

correlation. Besides correlation, we also define four similar metrics in terms of AUSE. More details related with the definition of AUSE are in (Ilg et al., 2018).

Meanwhile, as illustrated in Fig. 11, we find that sparsification error has a strong association with correlation, when the oracle sparsifications of different methods are the same. As a result, when AUSE of infer-dropout and MC-dropout (defined as MC-drop^1 here) is nearly identical, so is correlation.

F Performance Evaluation with L_2 Loss

We report AUSE, correlation with L_2 loss using the optimal hyper-parameters in different methods for SRGAN model, SRresnet model in Table 5, and FCRN model in Table 6. Our methods infer-

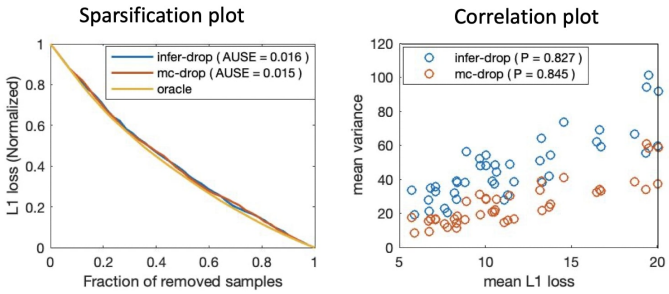

Figure 11: **Left:** The sparsification error plot using the mean of uncertainty and L_1 loss of each image sample for infer-dropout and MC-dropout, the values of AUSE are also presented. **Right:** The scatter plot using the mean of uncertainty and L_1 loss of each image sample, the values of mean L_1 correlation are also presented.

SRGAN model: trained with adversarial loss and L_2 loss													
Condition		Training Free (Ours)						Training Required					
Prediction		Original Output								Outputs Mean			
Method		Infer-trans		Infer-drop		Infer-noise		MC-drop^1		MC-drop^2		Ensemble	
Samples		4	8	8	32	8	32	8	32	8	32	4	8
AUSE	mean L_2	0.011	0.011	0.011	0.011	**0.009**	**0.009**	0.012	0.012	**0.009**	**0.009**	0.030	0.034
	patch L_2	**0.025**	0.026	0.032	0.035	0.037	0.036	0.035	0.035	0.035	0.035	0.049	0.050
	block L_2	**0.030**	**0.030**	0.038	0.038	0.042	0.041	0.037	0.036	0.043	0.042	0.053	0.054
	pixel L_2	0.147	**0.140**	0.176	0.163	0.192	0.182	0.162	0.151	0.178	0.159	0.197	0.183
Corr	mean L_2	0.885	0.885	0.871	0.862	0.844	0.846	0.896	0.896	**0.947**	0.943	0.660	0.669
	patch L_2	0.684	0.691	0.652	0.662	0.625	0.633	0.686	0.692	0.734	**0.740**	0.653	0.647
	block L_2	0.673	0.684	0.650	0.661	0.601	0.615	0.668	0.678	0.720	**0.730**	0.644	0.649
	pixel L_2	0.268	0.296	0.268	0.311	0.219	0.257	0.278	0.320	0.305	**0.351**	0.249	0.274
SRresnet model: trained with L_2 loss													
Condition		Training Free (Ours)						Training Required					
Prediction		Original Output								Outputs Mean			
Method		Infer-trans		Infer-drop		Infer-noise		MC-drop^1		MC-drop^2		Ensemble	
Samples		4	8	8	32	8	32	8	32	8	32	4	8
AUSE	mean L_2	0.051	0.051	0.050	0.050	0.056	0.056	**0.048**	0.052	0.061	0.062	0.133	0.128
	patch L_2	0.048	**0.047**	0.053	0.052	0.055	0.055	0.050	0.049	0.063	0.064	0.115	0.117
	block L_2	0.053	**0.051**	0.057	0.056	0.062	0.062	0.054	0.052	0.069	0.069	0.131	0.128
	pixel L_2	0.184	**0.174**	0.188	0.179	0.209	0.202	0.185	**0.174**	0.214	0.202	0.285	0.275
Corr	mean L_2	0.238	0.251	0.266	0.270	0.266	0.266	0.292	0.261	**0.525**	0.517	0.061	-0.032
	patch L_2	0.429	0.447	0.419	0.429	0.357	0.369	0.461	0.470	**0.521**	0.520	0.299	0.333
	block L_2	0.373	0.395	0.407	0.412	0.338	0.348	0.417	0.432	**0.529**	0.525	0.263	0.266
	pixel L_2	0.185	0.210	0.200	0.234	0.144	0.180	0.206	0.241	0.238	**0.267**	0.157	0.174

Table 5: Mean/patch-wise/block-wise/pixel-wise AUSE and correlation between L_2 loss and uncertainty on SR benchmark dataset Set 14. Our infer-transformation, infer-dropout and infer-noise are compared with MC-dropout (Gal & Ghahramani, 2016) and deep ensemble (Lakshminarayanan et al., 2017). MC-drop^1 uses the output of the original model as prediction while MC-drop^2 uses the mean of output samples from the re-trained model (with added dropout) as prediction. Models evaluated: SRGAN trained with L_2 loss and adversarial loss and SRresnet trained with L_2 loss.

transformation, infer-dropout and infer-noise are able to provide results that are comparable to or even better than those from training-required state-of-the-art methods such as MC-dropout, and they consistently outperform deep ensemble.

G THEOREM 3.1 AND PERFORMANCE BOUND

Table 7 shows correlation and AUSE with four variants (pixel-wise, mean, block-wise, patch-wise) for different methods compared to the oracle (performance bound) $\mathbb{V}[Z]/(t-C)^2$. Our methods, especially infer-transformation, are reasonably close to the oracle and compare favorably with baselines. The oracle (performance bound) is calculated with a empirically chosen constant margin $t = 5\sigma$, where σ^2 is average of $\mathbb{V}[Z]$ for all the pixels across the entire image.

FCRN model: Depth Estimation										
Condition		Training Free (Ours)					Training Required			
Prediction		Original Output							Outputs Mean	
Method		Infer-trans	Infer-drop		Infer-noise		MC-drop^1		MC-drop^2	
Samples		2	2	8	2	8	2	8	2	8
AUSE	mean L_2	0.083	0.070	**0.064**	0.078	0.067	0.110	0.110	0.108	0.110
	patch L_2	0.156	0.160	0.132	0.160	**0.131**	0.205	0.204	0.199	0.203
	block L_2	0.085	0.085	**0.067**	0.080	**0.067**	0.113	0.113	0.111	0.113
	pixel L_2	0.237	0.243	**0.184**	0.244	0.187	0.325	0.308	0.324	0.307
Corr	mean L_2	0.600	0.590	0.638	0.631	**0.675**	0.407	0.404	0.401	0.403
	patch L_2	0.346	0.328	**0.438**	0.325	0.431	0.225	0.233	0.233	0.234
	block L_2	0.375	0.370	**0.471**	0.378	0.461	0.187	0.190	0.193	0.191
	pixel L_2	0.205	0.175	**0.272**	0.178	0.271	0.054	0.096	0.056	0.096

Table 6: Mean/patch-wise/block-wise/pixel-wise AUSE and correlation between L_2 loss and uncertainty on NYU Depth Dataset V2. Our infer-transformation, infer-dropout and infer-noise are compared with MC-dropout (Gal & Ghahramani, 2016). MC-drop^1 uses the output of the original model as prediction while MC-drop^2 uses the mean of output samples from the re-trained model (with added dropout) as prediction. Models evaluated: FCRN model for depth estimation.

Method		MC-drop^1	MC-drop^2	Ensemble	LLM	Infer-trans		Infer-drop		Infer-noise	
		Baselines				Proposed	PB	Proposed	PB	Proposed	PB
AUSE	mean L_1	0.009	0.007	0.020	0.035	0.006	0.004	0.013	**0.002**	0.016	**0.002**
	patch L_1	0.029	0.025	0.033	0.044	0.022	**0.021**	0.031	0.024	0.043	0.027
	block L_1	0.029	0.028	0.033	0.042	0.023	**0.020**	0.030	0.024	0.039	0.022
	pixel L_1	0.131	0.129	0.144	0.137	0.121	0.097	0.141	**0.085**	0.162	0.106
Corr	mean L_1	0.943	0.936	0.694	0.484	0.930	0.938	0.882	0.956	0.780	**0.966**
	patch L_1	0.755	0.741	0.677	0.565	0.770	0.766	0.731	**0.794**	0.598	0.746
	block L_1	0.747	0.710	0.664	0.588	0.767	0.765	0.730	**0.789**	0.592	0.745
	pixel L_1	0.390	0.379	0.326	0.393	0.394	0.424	0.376	**0.590**	0.288	0.583

Table 7: Comparison of our methods and baselines as well as performance bound (PB). We mark the best performance bound in **bold face** and the best method by underlining. Model evaluated: SRGAN.

H Evaluations on Classification Tasks

We compare the simple training-free method using entropy and the training-required method MC-dropout on classification tasks. For classification tasks, the most straightforward and commonly used method is to calculate the entropy of output probability as uncertainty, which already qualifies as a training-free method. We then compare it with a sampling-based and training-required method – MC-dropout, tuned on different locations and using 8 samples. Here we conduct three experiments: the first one is multi-class segmentation using Densenet (Huang et al., 2017b) on CamVid dataset, the second one is binary segmentation using UNET (Ronneberger et al., 2015) on a biomedical public benchmark dataset from the SNEMI3D challenge, and the third one is classification on CIFAR100 using ResNet (He et al., 2016). We calculate the correlation between entropy of softmax output and the cross-entropy loss. We find that using entropy outperforms MC-dropout based on the correlation metric, as shown in Table 8.

	Densenet		UNET		Resnet	
	Entropy	MC-drop	Entropy	MC-drop	Entropy	MC-drop
Mean Correlation	**0.928**	0.718	**0.964**	0.881	**0.669**	0.537
Pixel-wise Correlation	**0.502**	0.209	**0.789**	0.317	–	–

Table 8: Correlation of uncertainty and cross-entropy loss, comparing using entropy with the baseline MC-dropout, models evaluated are Densenet for the segmentation on CamVid dataset, UNET for segmentation on SNEMI3D dataset and Resnet for classification on CIFAR100 dataset.

I Visualization of Uncertainty Maps

The uncertainty maps generated using infer-transformation, infer-dropout, infer-noise compared with MC-dropout, and the corresponding error maps for images are shown in Fig. 12 for the super-resolution task and in Fig. 13 for the depth estimation task.

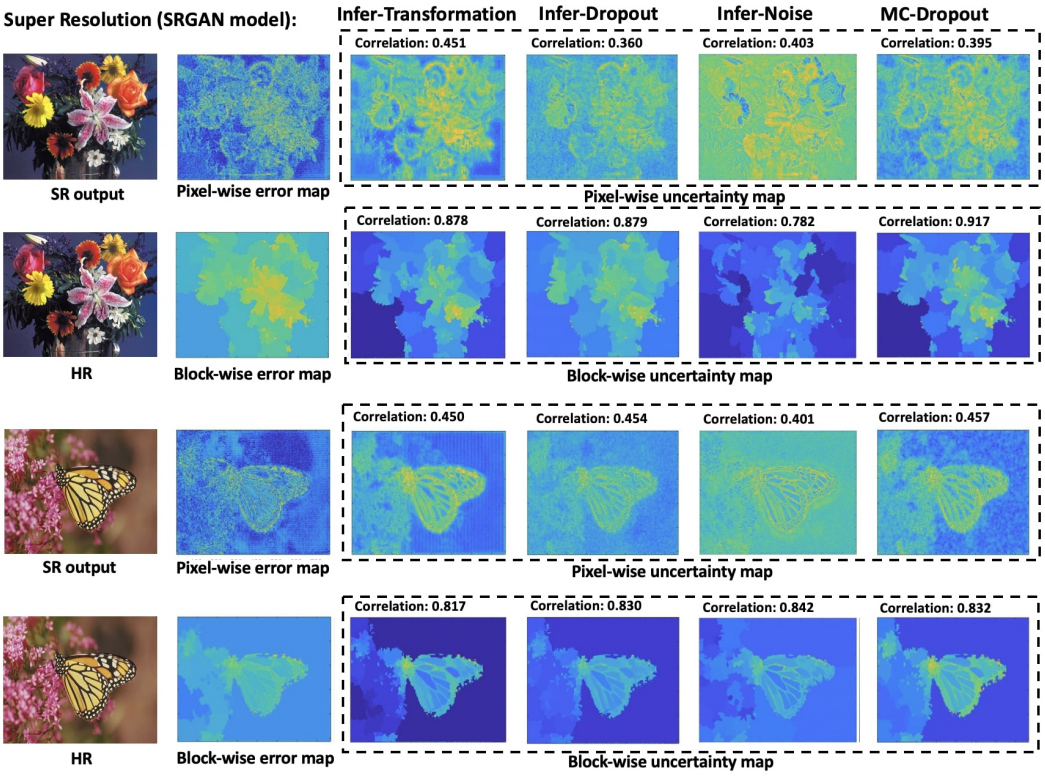

Figure 12: Visualization of uncertainty maps (log scale) and error map (log scale) from infer-transformation, infer-dropout, infer-noise compared with the baseline MC-dropout, evaluated on the SRGAN model on Set14 dataset for super-resolution task.

One interesting observation is that MC-dropout tends to capture local variance and ignore high-level semantics, in part because the dropout layer is always at the end of the network. As a result, it is difficult for MC-dropout to produce uncertainty estimates in detail-rich regions. For example, the input image in the bottom row of Fig. 13 contains a lot of details with chairs and desks. Unfortunately MC-dropout tends to ignore these details and only produce high variance in the upper half of the image (region with large depth), leading to poor correlation.

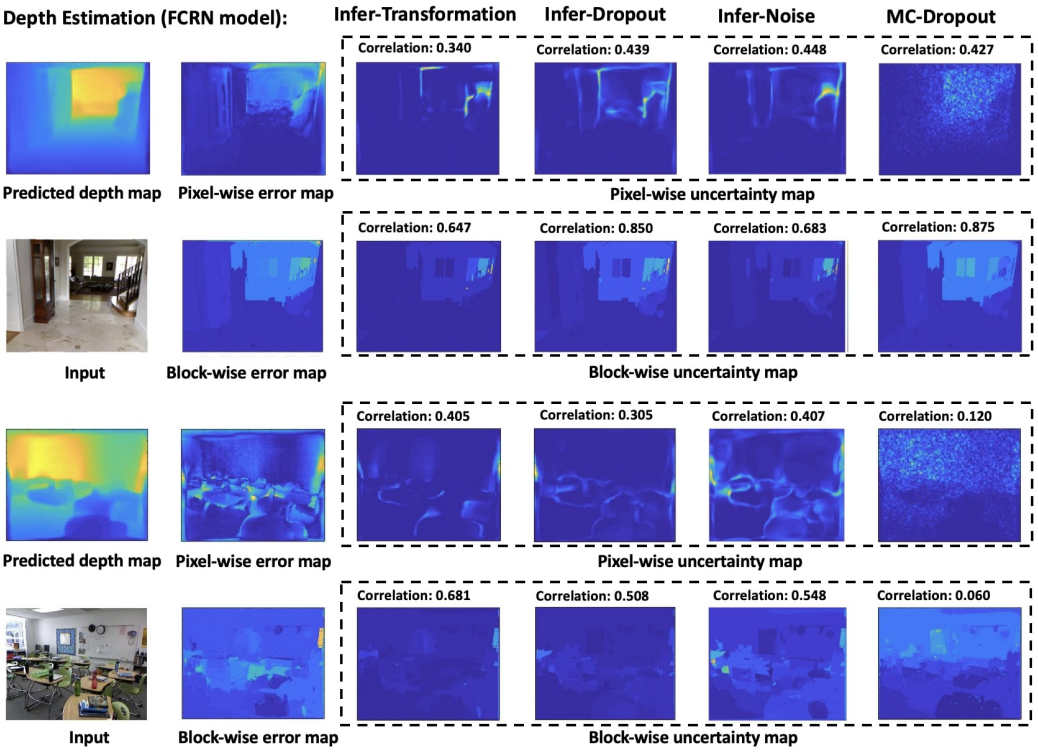

Figure 13: Visualization of uncertainty maps and the error map from infer-transformation, infer-dropout, infer-noise compared with the baseline MC-dropout, evaluated on the FCRN model on NYU depth dataset V2 for the depth estimation task.