Training-Free Uncertainty Estimation for Neural Networks

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. None of the previous methods are able to take an arbitrary model off the shelf and generate uncertainty estimation without retraining or redesigning it. To address this gap, we perform the first systematic exploration into training-free uncertainty estimation.

We propose three simple and scalable methods to analyze the variance of output 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 other 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 other training-required state-of-the-art methods. Last but not least, we demonstrate that the uncertainty from our proposed methods can be used to improve the neural network training.

1 Introduction

Deep learning is already able to achieve excellent 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. 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. Others utilize the randomness induced during training and inference (e.g., dropout and ensembling) to obtain an uncertainty estimation (Gal & Ghahramani, 2015; Lakshminarayanan et al., 2017).

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 never 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., 2015a; Hou et al., 2016).

To fill this gap, we first propose and define the problem of *training-free uncertainty estimation*: how to obtain an uncertainty estimation of any given model without re-designing, re-training, or finetuning it. We focus on two scenarios: black-box uncertainty estimation (BBUE), where one has access to the 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). To the best of our knowledge, ours is the first systematic exploration into the problem.

We propose a set of simple and scalable training-free methods to analyze the variance of output from a trained network. Our main idea is to add a *tolerable* perturbation into inputs or feature maps during inference. Different from an adversarial perturbation aiming to change the outputs during inference (Madry et al., 2017), a tolerable perturbation does not dramatically alter the original distribution while allowing generation of multiple diverse outputs that could later be used for uncertainty estimation. The first method, which we call *infer-transformation*, is to apply transformation that exploits the natural characteristics of a CNN: that it is variant to input transformation such as rotation (Cohen & Welling, 2016). Transformations have been frequently used for 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, which we call *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.

For classification, we note that the softmax output in classification models is naturally a distribution, the entropy of which could be directly used for training-free uncertainty estimation. Hence, using entropy for uncertainty estimation qualifies as a training-free method. We evaluate this method in two classification tasks (see details in Appendix A.1) and find that it already yields satisfactory uncertainty estimation (even more correlated with error compared with MC dropout). Therefore in this paper we focus on regression tasks where output distributions are not readily available. Following the previous work, we evaluate our proposed methods on two regression tasks, monocular depth estimation and single image super resolution, shown in Figure 1. Our major contributions are thus:

1. We perform, to the best of our knowledge, the first systematic exploration of training-free uncertainty estimation during inference with a post-hoc analysis, given any trained models.
2. We propose simple and scalable methods for regression models, using a tolerable perturbation such as infer-transformation or infer-noise injection to effectively and efficiently estimate uncertainty.
3. Surprisingly, we find that our methods are able to generate a comparable or higher correlation between variance and error than baseline methods, MC dropout and deep ensemble, in both large-scale regression tasks.
4. We demonstrate that the uncertainty from the proposed methods can be used to improve neural network training.

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 (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) 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, 2015; 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, 2015) injects noise 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.

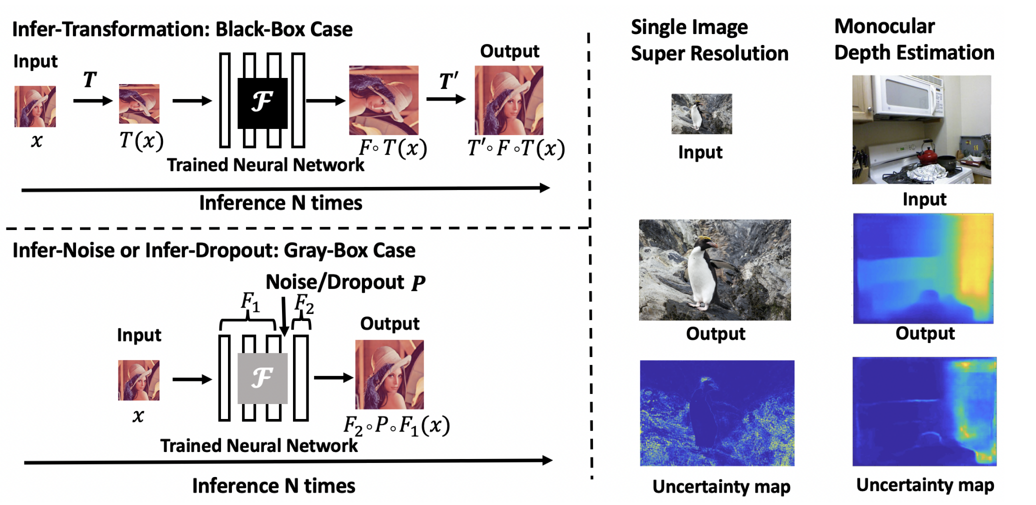


Figure 1: Left: Method description of training-free uncertainty estimation: Apply infer-transformation T (top) and infer-noise or infer-dropout P (bottom) to a trained neural network F during inference for N times. Right: Examples of uncertainty maps generated using our proposed method in two different tasks, single image super resolution and monocular depth estimation.

Non-probabilistic Neural Networks for Uncertainty Estimation: Other scalable strategies such as deep ensemble (Lakshminarayanan et al., 2017) 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 more efficient method is to train the network to have both original outputs and uncertainty predictions, jointly optimize for both (Zhang et al., 2019). However, such a design requires re-training, introduces heavy implementation overhead, and sometimes makes the optimization process more challenging.

3 Methodology

The idea at the core of our approach is to impose a tolerable perturbation on the model's input or intermediate representations (feature maps). We distinguish between three cases: 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). In this paper we focus on the black-box and gray-box cases, for which we offer, correspondingly, two classes of methods.

As shown in Figure 1, for the same image, they first (1) perform different tolerable perturbations on the input or intermediate-layer neurons, (2) generate multiple outputs under different perturbations, and (3) use the variance of the outputs as uncertainty estimation. Concretely, for the black-box case, we propose *infer-transformation*, which exploits the model's dependence on input transformations, e.g. rotation/flip. 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. These three methods are illustrated in Figure 1 and are described 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. We focus on transformations that preserve pertinent characteristics of the input, such as rotations, flips, etc. This method draws partial inspiration from adversarial attacks (Madry et al., 2017). Instead of causing 'large' changes in the output from small perturbations in the input, such as adversarial attack, we introduce tolerable-perturbation transformations of the input, exploiting the model's dependence on input transformations in practice, shown in Equation 1, where $T \in \mathcal{T}$ is a transformation, and T' is the inverse operation. In our method, $T' \circ T$ is cancelled out since the transformation is invertible, and F is the function represented by black-box neural networks. If input x is an $H \times W$ matrix, the variance ϵ_t estimated through transformation is also an $H \times W$ matrix. Assuming $E_{T \in \mathcal{T}}[T' \circ F \circ T(x)] = F \circ T' \circ T(x) = F(x)$, by the law of large number we have,

$$T' \circ F \circ T(x) \rightarrow \mathcal{N}(F \circ T' \circ T(x), \epsilon_t) = \mathcal{N}(F(x), \epsilon_t) \tag{1}$$

For the sake of simplicity and with little loss of generality, we apply rotations and flips to the input in our implementation. While, in principle, arbitrary rotation angles could be chosen for the creation of different outputs, in order not to introduce interpolation noise, we limit this discussion to the simpler case where the rotations are pure multiples of 90 degrees. Combining both rotation and flip, at most 8 samples could be generated during inference. Note that more outputs could be easily generated by similarly exploiting other transformations incurring zero-interpolation noise. Meanwhile, if such transformations are also applied during training, one might expect a reduction in the variance computed from multiple outputs for transformed inputs during inference. In our experiment, we evaluate the state-of-the-art models in super resolution and depth estimation, trained without this imposed augmentation. We find the strong correlation with error is still comparable with the value from baselines such as MC dropout.

3.2 Gray-Box Uncertainty Estimation: Infer-Noise and Infer-Dropout

To manipulate the intermediate representations (feature maps) of the model, we consider another class of methods for generating multiple outputs from a distribution: sampling from different latent codes. The main added benefit of this method is that it allows us to modulate the perturbation strength to ensure its tolerability.

Specifically we propose *infer-noise*, which introduces Gaussian noise at an intermediate representation within the trained model, and *infer-dropout*, which uses dropout instead. For infer-noise, the noise will be evenly distributed at the feature maps of a certain layer. This noise source is randomly sampled during inference allowing the corresponding resultant diverse outputs generation. Infer-dropout is motivated by observations on network compression and pruning (Han et al., 2015b): the intermediate layer has the most redundant information and therefore is the most effective location to prune (He et al., 2014). Naturally, dropping features in such locations incurs more tolerable perturbations. During each inference, the dropout is performed randomly to generate output samples, the variance of which are then used as uncertainty estimation.

Our method could be interpreted as Equation 2, where F_1 represents the function of network layers before the perturbation, denoted as $P \in \mathcal{P}$ (i.e., Gaussian noise or dropout), and F_2 represents the function of network layers come after. $F_2 \circ F_1(x)$ is the gray-box network $F(x)$, and we could use this perturbation P to an output distribution centered at $F_2 \circ F_1(x)$. Similarly, if input x is an $H \times W$ matrix, the variance ϵ_n estimated through infer-noise or infer-dropout is also an $H \times W$ matrix. After performing inference of the model for multiple times, the computation of the variance ϵ_n can naturally ensue over these output samples. We find the tuning of a proper location takes effect for uncertainty performance, which is not explored in MC-dropout. Note that conventionally in MC dropout, the location of noise layer is added before the fully connected (FC) layer, while infer-dropout allows for noise injection at arbitrary layers. Assuming $E_{P \in \mathcal{P}}[F_2 \circ P \circ F_1(x)] = F_2 \circ F_1(x) = F(x)$, by the law of large number we have,

$$F_2 \circ P \circ F_1(x) \rightarrow \mathcal{N}(F_2 \circ F_1(x), \epsilon_n) = \mathcal{N}(F(x), \epsilon_n) \tag{2}$$

4 Experiments

In this section, we evaluate our three proposed approaches in two large scale regression tasks, super-resolution and depth estimation, and compare our methods with two state-of-the-art methods, MC dropout and deep ensemble.

4.1 Single Image Super Resolution

The task of Single Image Super Resolution is to restore a high-resolution (HR) image from a low-resolution (LR) input. This problem is undetermined by definition, that is, given any low resolution input, there exist diverse plausible solutions. Here we focus on analyzing the state-of-the-art SRGAN model (Ledig et al., 2017), which is able to 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.

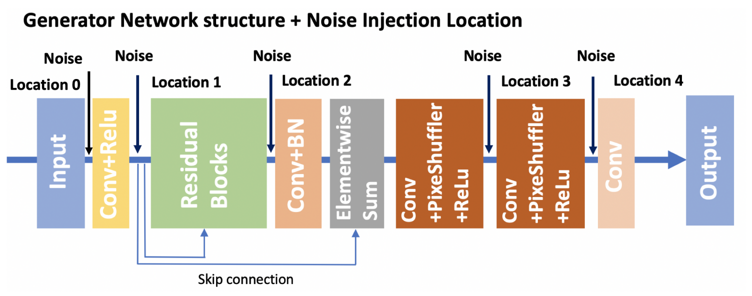


Figure 2: Different locations for infer-noise and infer-dropout in SRGAN and SRresnet for super resolution. For each experiment, the noise is injected at a single location with one perturbation level.

We apply our proposed 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 (details in Appendix A.2). We evaluate our methods on both models, and conduct experiments corresponding to the black-box/gray-box methodologies previously introduced:

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 mean 0 and standard deviation $\sigma \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, in different locations (layers). We choose 5 different locations for noise injection, as shown in Figure 2, including the layers right after the input and right before the output, as well as some intermediate layers. For each experiment, we only add the layer into one location with a specific σ value. During the inference, the noise will be randomly sampled from the Gaussian distribution, and we could get one sample from each inference pass. 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 in different locations. We choose dropout rates ρ from the set $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ use the same of locations as the infer-noise. Similarly, 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: The first baseline we test is MC-dropout (Gal & Ghahramani, 2015), instead of training the model with a dropout layer from scratch, we take the trained model released from (Dong et al., 2017) as a pre-trained model. We then add the dropout layer with varied 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. After adding dropout layer, we train the model for more epochs until it converges. The same dropout rate is used for sampling during inference. We try different sample numbers of 8 and 32. The second baseline we test is deep ensemble (Lakshminarayanan et al., 2017): we train the model from scratch for multiple times (specified as 4 and 8 in our experiments). We add the randomness into the initial weight, which is sampling from one Gaussian distribution, with the mean of zero and standard deviation σ of 0.02. 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.

4.2 Monocular Depth Estimation

Uncertainty estimation for the task of monocular depth estimation has been studied by many prior works (Postels et al., 2019; Kendall & Gal, 2017). Here we use one of the state-of-the-art models for depth estimation using fully convolutional residual network (FCRN) (Laina et al., 2016). It's trained with a Huber loss (Zwald & Lambert-Lacroix, 2012). We directly used 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. Note that the model has a dropout layer before the final fully connected (FC) layer during training; this dropout layer could directly be used during inference as an experiment of MC dropout. We compare MC dropout, where dropout is applied in the same layer

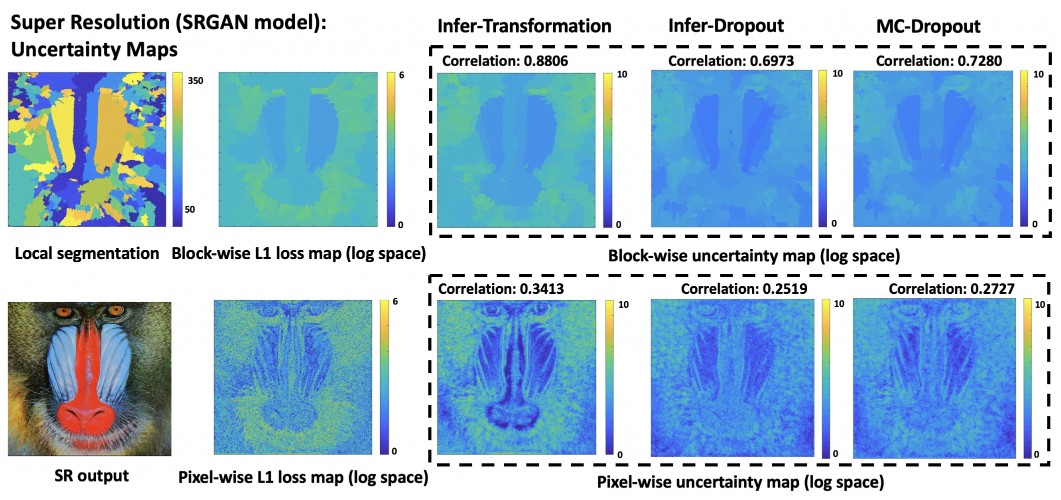


Figure 3: Block-wise and pixel-wise uncertainty (variance) maps generated using infer-transformation, infer-dropout, MC-dropout (Gal & Ghahramani, 2016) compared with L_1 loss map. Correlation values between uncertainty map and error map are also presented.

during training and inference, and infer-dropout, where dropout can be applied in arbitrary layers during inference.

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. Therefore we only apply horizontal flip and generate 2 samples for uncertainty estimation. For infer-dropout, we choose two locations to add dropout layer and conduct similar experiments as described in the SR task. For the baseline MC dropout, we directly perform sampling from the existing dropout layer. Sample numbers of 2 or 8 are evaluated for both cases adding dropout. Experiment details are shown in Appendix A.3.

4.3 Experiment Results

Evaluation Metrics Here we define three evaluation metrics: *pixel-wise correlation*, *mean correlation*, and *block-wise correlation*. Take L_1 loss as an example, the uncertainty (variance) V_{ij} estimated in these methods and the regression error $L_{1,ij}$ are both pixel-wise values; naturally the pixel-wise L_1 correlation is defined as $corr(\{V_{ij}\}, \{L_{1,ij}\})$. The mean L_1 correlation is defined as $corr(\{\overline{V}_z\}, \{\overline{L}_{1,z}\})$, where $\overline{V}_z$ and $\overline{L}_{1,z}$ are the average variance and average error of a single image z, respectively. This metric has been used in (Zhang et al., 2019). The third evaluation metric is block-wise correlation – a new metric we propose in this work. To compute block-wise correlation, we first run a local segmentation algorithm on the output of the trained model to generate each cluster C_i (block) with similar low-level context. Here we use the local center of mass approach (Aganj et al., 2018). The local segmentation results with clusters are shown in Figure 3. The block-wise L_1 correlation is defined as $corr(\{\widetilde{V}_i\}, \{\widetilde{L}_{1,i}\})$, where $\widetilde{V}_i$ and $\widetilde{L}_{1,i}$ are the average pixel-wise variance and error inside each cluster. The intuition is that in many situations it is more instructive and meaningful when uncertainty is visualized in each sub-region (e.g., in a sub-region with a possible tumor). Detailed descriptions of these metrics are in Appendix A.4.

Figure 3 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. Specifically, in our methods, high variance occurs in areas with high randomness and high-frequency spectral contents. For a depth estimation task however, high variance usually occurs in the area with high spatial resolution and large depth. More examples are available in Appendix A.6.2.

The Role of Tolerable Perturbations Tolerable perturbations play a crucial role in obtaining effective uncertainty estimation. Here tolerability can be measured by the disturbance of accuracy or loss. In general we observe that better tolerability corresponds to be better uncertainty estimation. Figure 4 shows the correlation for different amount of perturbations (noise or dropout) in different locations. As we can see, the optimal cases to generate uncertainty maps with high correlation require that the loss should remain small after perturbation (high tolerability). Interestingly, different methods have different ways of achieving high tolerability: (1) For MC dropout, involving dropout

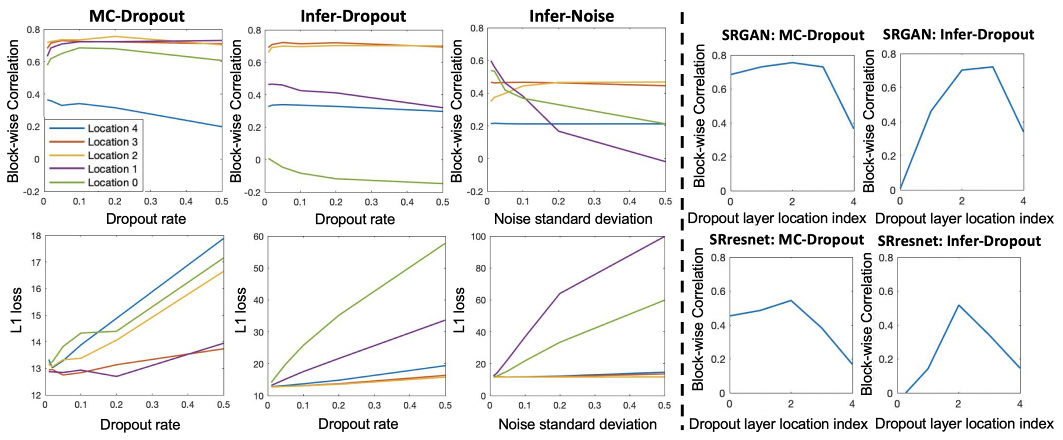


Figure 4: Left: Correlation changes with different locations where dropout and noise layers are inserted. Various dropout rates and noise levels have been evaluated. We compare our infer-noise and infer-dropout with baseline MC-dropout (Gal & Ghahramani, 2016). Location 0 is right after the input; location 4 is right before the last convolutional layer; location 1, 2, 3 are intermediate layers. Right: Intermediate layers are optimal locations to add dropout layers for uncertainty estimation, validated in both infer-dropout and MC-dropout methods. The corresponding correlation is the optimal value of each location, after tuning with dropout rates.

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 the information is the most redundant, 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; it is different from convention where dropout is applied before the final FC layers. Applying infer-dropout in these intermediate layers, we could achieve comparable or even better correlation values compared to training-required approaches. For infer-noise, locations do not have similar effect, and therefore further tuning of locations and σ is also required. The conclusion above is also consistent with the evaluation of other models (see more details in Appendix A.6.1).

Comparable Performance with Training-required Baselines We summarize the correlation value using the optimal hyper-parameters in different methods in Table 1 and Table 2. As depicted in both tasks, our methods infer-transformation and infer-dropout are able to provide comparable results with the training-required state-of-the-art method, MC-dropout, and are better than deep ensemble. For the super-resolution task, we find that SRGAN has a higher correlation than the SR-resnet model. One explanation is that the model trained with adversarial loss is more sensitive to perturbation, and will cause various artifacts in the area of high uncertainty during each inference.

For depth estimation, we find that using infer-dropout in the intermediate layers outperforms other methods. MC dropout only allows perturbation before the last convolutional layer, the location added with dropout layer during training for the original trained model, and therefore produces a highly localized variance map with poor correlation (more details available in Appendix A.6.2). 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. Interestingly, even with only 2 samples, infer-transformation can still outperform all other methods.

5 Applications

We demonstrate two applications of uncertainty maps estimated by our methods, in the context of super resolution. In general, estimated uncertainty can be adopted to improve the training process. For more details see Sec. A.5. The first application is to take the pixel-wise uncertainty map as a weight term for regression loss, and provide a higher penalty in highly uncertain regions. As expected, loss weighted by uncertainty can provide a more photo-realistic SR output with finer structures and sharper edges, as shown in Figure 5.

SRGAN model: trained with adversarial loss and L_2 loss										
Condition	Training Free (Ours)						Training Required			
Method	Infer-transformation		Infer-dropout		Infer-noise		MC-dropout		Deep Ensemble	
Samples	4	8	8	32	8	32	8	32	4	8
mean L_1	0.9380	0.9366	0.8932	0.8918	0.7858	0.7867	0.9515	**0.9532**	0.6521	0.6803
mean L_2	0.8993	0.8990	0.8878	0.8808	0.8575	0.8559	**0.9131**	0.9115	0.6056	0.6307
block-wise L_1	0.7654	**0.7697**	0.7222	0.7313	0.5899	0.5984	0.7477	0.7551	0.6362	0.6536
block-wise L_2	0.6841	0.6905	0.6515	0.6615	0.6251	0.6334	0.6856	**0.6920**	0.5921	0.6120
pixel-wise L_1	0.3671	**0.3945**	0.3233	0.3762	0.2450	0.2877	0.3388	0.3898	0.2537	0.2962
pixel-wise L_2	0.2681	0.2956	0.2676	0.3105	0.2186	0.2566	0.2779	**0.3198**	0.2108	0.2474

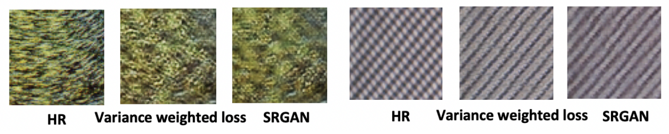
SRResnet model: trained with L_2 loss										
Condition	Training Free (Ours)						Training Required			
Method	Infer-transformation		Infer-dropout		Infer-noise		MC-dropout		Deep Ensemble	
Samples	4	8	8	32	8	32	8	32	4	8
mean L_1	0.3654	0.3843	0.5239	**0.5314**	0.1044	0.0909	0.4813	0.4628	-0.1595	-0.2974
mean L_2	0.2661	0.2804	0.3628	**0.3770**	0.3021	0.2933	0.3281	0.2866	-0.2440	-0.2935
block-wise L_1	0.5014	0.5197	0.5079	0.5178	0.3714	0.3854	0.5346	**0.5447**	0.2771	0.2726
block-wise L_2	0.4288	0.4468	0.4190	0.4286	0.3574	0.3688	0.4607	**0.4702**	0.2461	0.2275
pixel-wise L_1	0.2372	0.2690	0.2584	0.3025	0.1721	0.2156	0.2638	**0.3087**	0.1708	0.1740
pixel-wise L_2	0.1845	0.2097	0.2003	0.2339	0.1440	0.1796	0.2055	**0.2414**	0.1374	0.1390

Table 1: Mean/block-wise/pixel-wise correlation of L_1/L_2 loss and uncertainty on SR benchmark dataset set 14. Our infer-transformation and infer-noise injection are compared with MC dropout (Gal & Ghahramani, 2016) and deep ensemble (Lakshminarayanan et al., 2017). Models evaluated: SRresnet trained with L_2 loss and SRGAN trained with L_2 loss and adversarial loss.

FCRN model: Depth Estimation					
Condition	Training Free (Ours)			Training Required	
Method	Infer-transformation	Infer-dropout		MC-dropout	
Samples	2	2	8	2	8
mean L_1	0.5960	0.6302	**0.6770**	0.4711	0.4706
mean L_2	0.5995	0.5886	**0.6380**	0.4060	0.4039
block-wise L_1	0.3544	0.3541	**0.4487**	0.2117	0.2186
block-wise L_2	0.3748	0.3696	**0.4712**	0.1837	0.1891
pixel-wise L_1	0.2082	0.1815	**0.2836**	0.0745	0.1335
pixel-wise L_2	0.2047	0.1750	**0.2724**	0.0541	0.0956

Table 2: Mean/block-wise/pixel-wise correlation of L_1/L_2 loss and uncertainty on NYU Depth Dataset V2. Our infer-transformation and infer-dropout are compared with MC dropout (Gal & Ghahramani, 2016). Models evaluated: FCRN model for depth destimation.

Another natural application is active learning (Gal et al., 2017). In our experiment, we firstly use the original model trained on DIV2K training data to super-resolve the DIV2K test data (as our validation data here). Then the model using active learning strategy is fine-tuned with the 25% validation data with the highest uncertainty. Two baseline models are the original trained model and the model fine-tuned with 25% of validation data which is randomly chosen. Three models are evaluated in Set 14, image quality is quantified by standard metrics. We find that active learning based on our generated uncertainty estimation can improve the performance, shown in Figure 5.



Dataset	Set 14		
Method	Original	Random	Selective
SSIM	0.7719	0.7727	**0.7896**
PSNR	23.8412	23.6161	**23.9415**
L1 Loss	11.6228	12.3442	**11.4967**

Figure 5: Left: We compare SR results that use loss reweighted by variance map and that do not. HR represents high resolution image. Right: in active learning setting, choosing samples with high uncertainty yields better results than choosing randomly.

6 Discussion and Conclusion

In the work, we perform the first systematic exploration into training-free uncertainty estimation. We propose three simple, scalable, and effective methods, namely *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 uncertainty maps with high correlation to error maps for all methods we studied. Our future work includes evaluating our methods on distilled, pruned and binarized models, as well as generalizing our methods for more complicated noise/transformation (e.g., non-Gaussian noise arbitrary-angle rotation).

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.

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 *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2722–2730, 2015.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, 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 *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 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: Insights and applications. In *Deep Learning Workshop, ICML*, 2015.

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

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1183–1192. JMLR. org, 2017.

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

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

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 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 *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 245–249. IEEE, 2014.

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.

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 *Advances in neural information processing systems*, pp. 5574–5584, 2017.

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

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pp. 239–248. IEEE, 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

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

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 *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 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. *The journal of machine learning research*, 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.

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

Zizhao Zhang, Adriana Romero, Matthew J Muckley, Pascal Vincent, Lin Yang, and Michal Drozdzal. Reducing uncertainty in undersampled mri reconstruction with active acquisition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2049–2058, 2019.

Laurent Zwald and Sophie Lambert-Lacroix. The berhu penalty and the grouped effect. *arXiv preprint arXiv:1207.6868*, 2012.

A Appendix

A.1 Classification Tasks

For classification task, the most straightforward and commonly used method is to calculate the entropy of output probability as uncertainty, which already satisfy our claim of training-free method. And we compare with sampling method – MC-dropout, tuned on different locations and using 8 samples. Here we implement two experiments: one is classification on dataset CIFAR100 using Resnet (He et al., 2016), and another one is binary segmentation using UNET (Ronneberger et al., 2015) on biomedical public benmark dataset from SNEMI3D challenge. We calculate the correlation between entropy of softmax output and cross-entropy loss. And we find using entropy outperforms MC-dropout in correlation metric, shown in Table 3.

UNET model: segmentation & Resnet model: classification						
Task	Correlation	Entropy	MC-drop	Task	Entropy	MC-dropout
segmentation	mean	**0.9643**	0.8810	Classification	**0.6692**	0.5365
	pixel-wise	**0.7892**	0.3168		–	–

Table 3: Correlation of uncertainty and cross-entropy loss, comparing using entropy with baseline MC dropout, models evaluated are UNET for segmentation on SNEMI3D dataset and Resnet for classification on CIFAR100 dataset.

A.2 Super Resolution Task

The SRresnet is trained with L_2 loss. The SRGAN is trained with L_2 loss with a weight of 100 and adversarial loss with a weight of 1. Due to effect of adversarial loss, SRGAN model is able to generate more photo-realistic SR outputs, but it is worse than the SRresnet when the output quality is quantified in standard metrics, such as SSIM and PSNR. The SRresnet model is trained for 100 epochs. The SRGAN model use the SRresnet model as pre-trained model, and then it is trained for 2000 epochs. The training dataset is DIV2K train set, while the test data we evaluated here is benchmark Set 14 dataset.

A.3 Depth Estimation Task

Infer-Transformation: Different from super-resolution, for the depth estimation task, we avoid to apply 90 degree rotation to the input, since the output will have an apparent difference, which contradicts our pre-condition of tolerable perturbations. The reason we consider is that the spatial direction is an important prior for depth estimation, i.e. the sky or ceiling in the top region of one image naturally has a larger depth, while in the bottom region of the image usually has a relatively smaller depth. Hence, we only apply horizontal flips to inputs, and 2 samples will be generated. Surprisingly, we find even 2 samples are also available to provide satisfying correlation value, and better than the MC dropout. Interestingly, we compare the horizontal flips similar to the simulation of depth inference from human vision system or stereo camera using two or more perspective of views.

Infer-Dropout: In the task, we take the trained model and add a dropout layer with varied dropout rates in two different locations, location 2 and 3 shown in Figure 6. The dropout rate we choose ranges from 0.01, 0.02, 0.05, 0.1, 0.2, 0.5. More details of neural network structures are in (Laina et al., 2016). For each experiment, we add dropout layer into one location with a specified dropout rate. Based on the fact that this network structure has similar layer types, we can further infer that it is the layer location which takes effect for uncertainty estimation, instead of layer types. We test different sampling numbers of 2 or 8.

Baseline: We use MC-dropout as the baseline. Since the original trained network has a dropout layer before the final FC layer – location 4 shown in Figure 6. Then we directly perform sampling at that location during inference, with dropout rate ranged from 0.01, 0.02, 0.05, 0.1, 0.2, 0.5. We test different sampling numbers of 2 or 8.

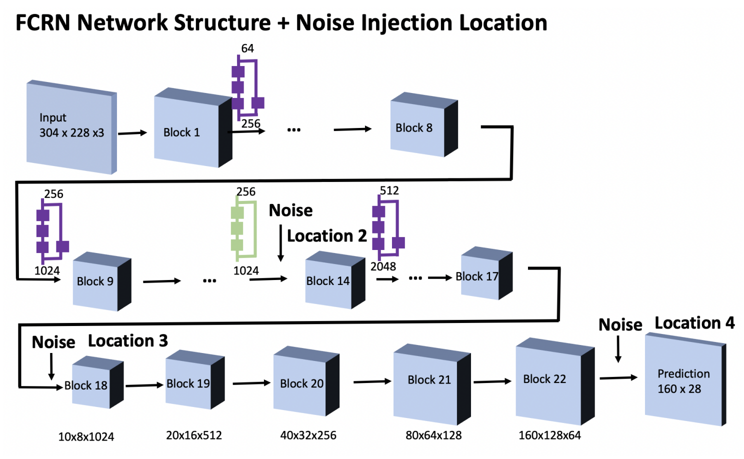


Figure 6: Different locations for infer-noise injection in the FCRN model for depth estimation Task.

A.4 Evaluation Metrics

Assumed to have N outputs given the same input x from our infer-transformation and infer-noise injection methods, 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, correspondently, 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 $corr(\{\overline{V}_z\}, \{\overline{L}_{1,z}\})$. The third metric for evaluation is the block-wise correlation – a new metric we propose in this work. To evaluate results on this metric, we need to firstly apply the output from the trained model with a local segmentation algorithm to generate each cluster C_i with similar low-level context. And then the variance of K_{C_i} pixels inside each cluster (block) C_i is 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 same is applied for the L_1 loss of each pixel to calculate the block-wise loss $\widetilde{L}_{1,i}$. Then we calculate the pixel-wise correlation of each pixels with the updated value as the L_1 block-wise correlation $corr(\{\widetilde{V}_i\}, \{\widetilde{L}_{1,i}\})$.

A.5 Applications

A.5.1 Variance weighted Loss

We use the released state-of-the-art SRGAN model as the pre-trained model. Then we use the dot multiplication of the variance with L_2 loss for each pixel to replace the original L_2 loss term, while keep the adversarial loss term the same as in the original training process. Then the model is fine-tuned for 45 epochs.

A.5.2 Active learning

The original SRGAN model is trained on DIVI2K train set, and then we evaluate the original trained model on the DIVI2K test set – as our validation dataset here. Then we select 25% of the DIVI2K test images whose prediction have highest uncertainty into the fine-tune dataset for a fast fine-tune process. We re-train the original model for additional 30 epochs. The final model is evaluated on the Set14 dataset, the original trained model and from randomly add 25% of the images in DIVI2K test set for the same fine-tune process are tested as baselines.

A.6 Experiment Results

A.6.1 Correlation v.s. Location

The plot of correlation with varied dropout rates using for MC-dropout and infer-dropout respectively, for other two models, SRresnet for super-resolution task and FCRN for depth estimation task is shown in Figure 7. Their results are consistent with the results of SRGAN model.

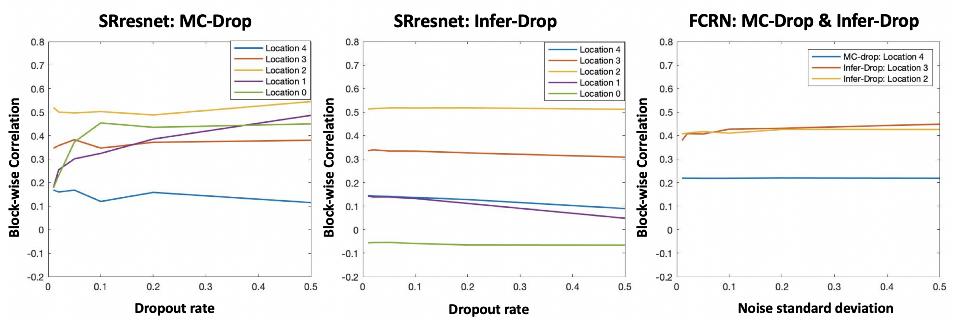


Figure 7: Correlation is varied with different dropout rates. Infer-dropout is compared with baseline MC-dropout, evaluated on the SRresnet model for super-resolution task and the FCRN model for depth estimation task.

A.6.2 Visualization of Uncertainty Map

The uncertainty maps generated with adding dropout layer into different locations are able to visualized in Figure 8. The uncertainty maps generated using infer-transformation, infer-dropout, compared with MC-dropout, and the corresponding error maps for images are able to be visualized in Figure 9 for super resolution task and in Figure 10 for depth estimation task.

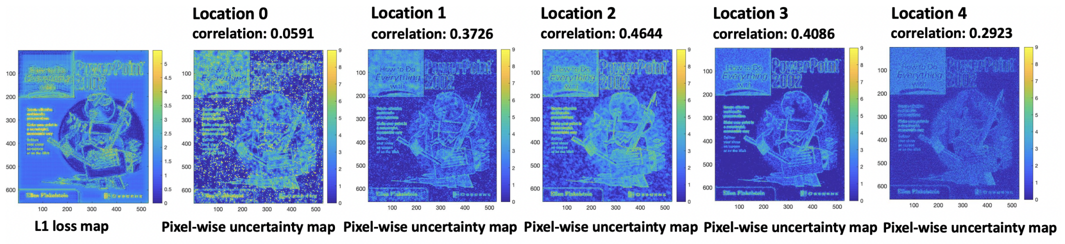


Figure 8: Visualization of error map and uncertainty maps generated from infer-dropout, each variance map is from a different location, evaluated on the SRGAN model on set14 dataset for super-resolution task.

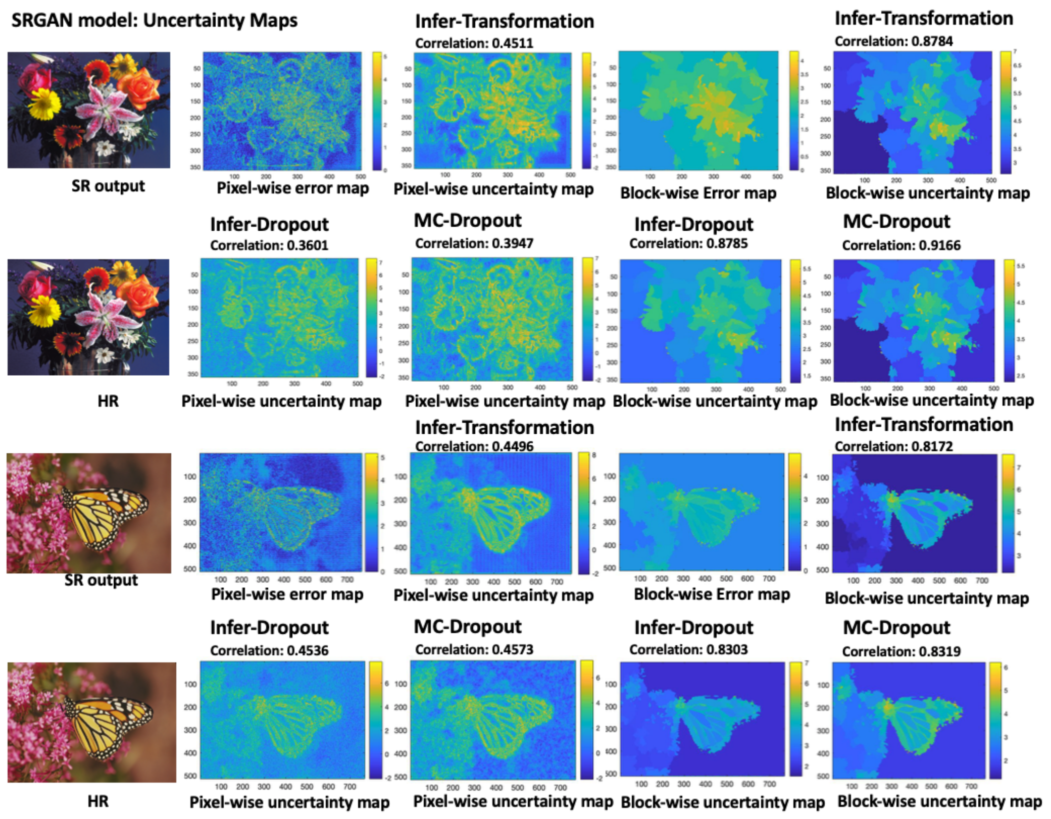


Figure 9: Visualization of uncertainty maps and error map from infer-transformation, infer-dropout compared with baseline MC-dropout, evaluated on the SRGAN model on Set14 dataset for super-resolution task.

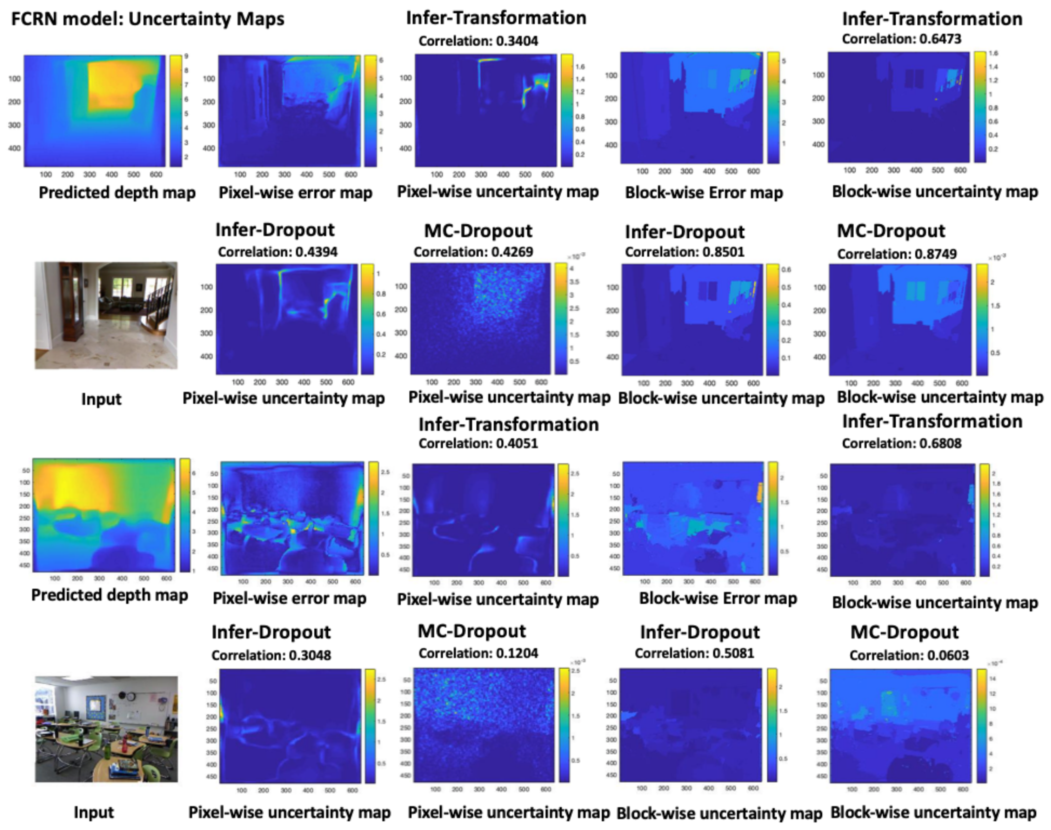


Figure 10: Visualization of uncertainty maps and error map from infer-transformation, infer-dropout compared with baseline MC-dropout, evaluated on the FCRN model on NYU depth dataset V2 for depth estimation task.