Blurs Make Results Clearer: Spatial Smoothings to Improve Accuracy, Uncertainty, and Robustness

Namuk Park  
Yonsei University  
namuk.park@yonsei.ac.kr

Songkuk Kim  
Yonsei University  
songkuk@yonsei.ac.kr

Abstract

Bayesian neural networks (BNNs) have shown success in the areas of uncertainty estimation and robustness. However, a crucial challenge prohibits their use in practice: Bayesian NNs require a large number of predictions to produce reliable results, leading to a significant increase in computational cost. To alleviate this issue, we propose spatial smoothing, a method that ensembles neighboring feature map points of CNNs. By simply adding a few blur layers to the models, we empirically show that the spatial smoothing improves accuracy, uncertainty estimation, and robustness of BNNs across a whole range of ensemble sizes. In particular, BNNs incorporating the spatial smoothing achieve high predictive performance merely with a handful of ensembles. Moreover, this method also can be applied to canonical deterministic neural networks to improve the performances. A number of evidences suggest that the improvements can be attributed to the smoothing and flattening of the loss landscape. In addition, we provide a fundamental explanation for prior works—namely, global average pooling, pre-activation, and ReLU6—by addressing to them as special cases of the spatial smoothing. These not only enhance accuracy, but also improve uncertainty estimation and robustness by making the loss landscape smoother in the same manner as the spatial smoothing. The code is available at https://github.com/xxxnell/spatial-smoothing.

1 Introduction

In a real-world environment where many unexpected events occur, machine learning systems cannot be guaranteed to always produce accurate predictions. In order to handle this issue, we make system decisions more reliable by considering estimated uncertainties, in addition to predictions. Uncertainty quantification is particularly crucial in building a trustworthy system in the field of safety-critical applications, including medical analysis and autonomous vehicle control. However, canonical deep neural networks (NNs)—or deterministic NNs—cannot produce reliable estimations of uncertainties [14], and their accuracy is often severely compromised by natural data corruptions from noise, blur, and weather changes [9] [2].

Bayesian neural networks (BNNs), such as MC dropout [12], provide a probabilistic representation of NN weights. They combine a number of models selected...
Figure 2: Comparison of three different Bayesian neural network inferences: canonical BNN inference, the temporal smoothing [29], and the spatial smoothing (ours). In this figure, \( x_0 \) is observed data, \( p_i \) are predictions \( p(y|x_0, \omega_i) \) or \( p(y|x_i, \omega_i) \), \( \pi_i \) are importances \( \pi(x_i|x_0) \), and \( N \) is ensemble size.

Based on weight probability to make predictions of desired results. Thanks to this feature, BNNs have been widely used in the areas of uncertainty estimation [20] and robustness [28]. They are also promising in other fields like out-of-distribution detection [27] and meta-learning [45].

Nevertheless, there remains a significant challenge that prohibits their use in practice. BNNs require an ensemble size of up to fifty to achieve high predictive performance, which results in a fiftyfold increase in computational cost [20][26]. Therefore, if BNNs can achieve high predictive performance merely with a handful of ensembles, they could be applied to a much wider range of areas.

1.1 Preliminary

We would first like to discuss BNN inference in detail, then move on to VQ-BNN inference [29], an efficient approximated BNN inference.

BNN inference. Suppose we have access to posterior probability of NN weight \( p(\omega|D) \) for training dataset \( D \). The predictive result of BNN is given by the following predictive distribution:

\[
p(y|x_0, D) = \int p(y|x_0, \omega) \ p(\omega|D)\ d\omega
\]

where \( x_0 \) is observed input data vector, \( y \) is output vector, and \( p(y|x, \omega) \) is the probabilistic prediction parameterized by the result of NN for an input \( x \) and weight \( \omega \). In most cases, the integral cannot be solved analytically. Thus, we use the MC estimator to approximate it as follows:

\[
p(y|x_0, D) \approx \sum_{i=0}^{N-1} \frac{1}{N} \ p(y|x_0, \omega_i)
\]

where \( \omega_i \sim p(\omega|D) \) and \( N \) is the number of the samples. The equation indicates that BNN inference is ensemble average of NN predictions for one observed data as shown on the left of Fig. 2. Using \( N \) number of ensembles would requires \( N \) times more computational complexity than one NN execution.
VQ-BNN inference. To improve the computational performance of BNN inference, VQ-BNN executes NN for an observed data only once and complements the result with previously calculated predictions of other data. If we have access to previous predictions, the computational performance of VQ-BNN becomes comparable to that of one NN execution. To be specific, VQ-BNN inference is:

\[ p(y | x_0, D) \approx \sum_{i=0}^{N-1} \pi(x_i | x_0) p(y | x_i, w_i) \]  

where \( \pi(x_i | x_0) \) is the importance of data \( x_i \) with respect to the observed data \( x_0 \), and it is defined as a similarity between \( x_i \) and \( x_0 \). \( p(y | x_i, w_i) \) is the newly calculated prediction, and \( \{ p(y | x_1, w_1), \cdots \} \) are previously calculated predictions. To accurately infer the results, the previous predictions should consist of predictions for “data similar to the observed data”.

Thanks to the temporal consistency of real-world data streams, aggregating predictions for similar data in data streams is straightforward. Since temporally proximate data sequences tend to be similar, we can memorize recent predictions and calculates their average using exponentially decreasing importance. In other words, VQ-BNN inference for data streams is simply the temporal smoothing of recent predictions as shown in the middle of Fig. 2.

VQ-BNN may be a promising approach to obtain reliable results in an efficient way, but it also has two limitations. First, it was only applicable to data streams such as video sequences. Applying VQ-BNN to images is challenging because it is impossible to memorize all similar images in advance. Second, Park et al. [29] used VQ-BNN only in the testing phase, not in the training phase. We find that ensembling predictions for similar data helps in NN training by smoothing the loss landscape.

1.2 Contribution

Our main contribution is threefold:

- Spatially neighboring points in visual imagery tend to be similar, as do feature maps of convolutional neural networks (CNNs). By exploiting this spatial consistency, we propose spatial smoothing as a method of ensembling nearby feature map points to improve the efficiency of ensemble size in BNN inference. The right side of Fig. 2 visualizes the spatial smoothing layer aggregating neighboring feature maps.

- We empirically demonstrate that the spatial smoothing improves the efficiency in vision tasks, such as image classification on CIFAR and ImageNet, without any additional training parameters. Figure 3 shows that negative log-likelihood (NLL) of “MC dropout + spatial smoothing” with an ensemble size of two is comparable to that of vanilla MC dropout with an ensemble size of fifty. We also demonstrate that the spatial smoothing improves accuracy, uncertainty, and robustness all at the same time. Figure 1 shows that the spatial smoothing improves both the accuracy and uncertainty of various deterministic and Bayesian NNs with an ensemble size of fifty.

- Global average pooling (GAP) [25, 48], pre-activation [16], and ReLU6 [21, 34] have been widely used in vision tasks. However, their motives are largely justified by the experiments. We provide an explanation for these methods by addressing them as special cases of the spatial smoothing. Experiments support the claim by showing that the methods improve not only accuracy but also uncertainty and robustness.

![Figure 3: Predictive performance of ResNet-18 on CIFAR-100.](image-url)
2 Probabilistic Spatial Smoothing

To improve computational performance of BNN inference, VQ-BNN [29] executes neural network prediction only once and complements the result with previously calculated predictions. The key to the success of the approach depends on the collection of previous predictions for proximate data. It is easy to gather proximate data and their predictions from data streams. By using temporal consistency, we aggregate recent data and predictions. In contrast, time-independent data such as image do not have such consistency and we cannot use it.

**Ensembling nearby feature map points.** Instead of temporal consistency, we use the spatial consistency of real-world images, which means that nearby pixels of images are similar. By using this hypothesis, we take the feature maps as predictions and aggregate the neighboring feature maps. Most CNN architectures, such as ResNet, consist of multiple stages starting with increasing the number of channels while reducing the spatial dimension of the input volume. We first decompose entire BNN inference into several steps by rewriting each stage in a recurrence relation as follows:

$$p(z_{i+1}|z_i, D) = \int p(z_{i+1}|z_i, w_i) p(w_i|D) dw_i$$  \hspace{1cm} (4)

where $z_i$ is input volume of $i$-th stage, and the first and the last are the input data and the output, respectively. $w_i$ and $p(w_i|D)$ are NN weights in $i$-th stage and its probability, $p(z_{i+1}|z_i, w_i)$ is output probability of $z_{i+1}$ with respect to the input volume $z_i$. To derive the probability from the output feature map, we transform each point of the feature map into a Bernoulli distribution. To do so, a composition of tanh and ReLU, a function from value of range $[-\infty, \infty]$ into probability, is added after each stage. To put it shortly, we use neural networks for point-wise binary classification.

Since Eq. (4) is a kind of BNN inference, it can be approximated using Eq. (3). In other words, each stage predicts feature map points only once and complements the predictions with similar feature maps. Under the spatial consistency, it averages spatially nearby feature map probabilities, which is well known as blur operation in image processing. For the sake of implementation simplicity, average pooling with kernel size of 2 and stride of 1 is used as a box blur. This operation ensembles four neighboring probabilities with the same importances.

In summary, as shown in Fig. 4, we propose probabilistic spatial smoothing layer as follows:

$$\text{Smooth}(z) = \text{Blur} \circ \text{Prob}(z)$$ \hspace{1cm} (5)

where Prob($\cdot$) is point-wise function from feature map to probability, and Blur($\cdot$) is importance-weighted average to ensemble spatially neighboring probabilities from feature maps. The Smooth layer is added after every stage. Prob and Blur will be further elaborated below.

**Prob: Feature maps to probabilities.** Prob is a function that transforms a real-valued feature map into probability. For that purpose, we use the tanh–ReLU composition. However, it is generally known that tanh suffers from vanishing gradient problem. To alleviate this issue, we propose temperature-scaled tanh as follows:

$$\text{tanh}_\tau(z) = \tau \text{tanh}(z/\tau)$$ \hspace{1cm} (6)

where $\tau$ is hyperparameter called temperature. It is conventional tanh when $\tau$ is 1, and the identity function when $\tau$ is $\infty$ as shown in Fig. B.2. $\text{tanh}_\tau$ imposes an upper bound on a value, but does not limit it to 1.

An unnormalized probability, ranging from 0 to $\tau$, is allowed as the output of Prob. Then, thanks to the linearity of integration, we obtain an unnormalized predictive distribution accordingly. Taking
Figure 5: Standard deviation of feature map for block depth with ResNet-50 on CIFAR-100. c1 to c4 and s1 to s4 stand for stages and the spatial smoothing layers, respectively. Model uncertainty is represented by the average standard deviation of several feature maps obtained from multiple NN executions. Data uncertainty is represented by the standard deviation of feature map points obtained from one NN execution.

this into account, we propose Prob as follows:

$$\text{Prob}(z) = \text{ReLU} \circ \tanh_\tau(z)$$  \hspace{1cm} (7)

where $\tau > 1$. Most CNN stages end with ReLU. Therefore, in order to improve computational performance, ReLU can be omitted from Prob in these cases, because $\tanh_\tau$ and ReLU are commutative.

We empirically determine $\tau$ to minimize NLL, a metric that measures both accuracy and uncertainty. When $\tau$ is small, the regularization of Prob dominates, so we obtain better uncertainty; however, it degrade accuracy. $\tau$ is optimized when accuracy and uncertainty are balanced. See Appendix B.1 and Fig. B.3 for more details.

We expect upper-bounded functions, e.g. ReLU6 which is $\min(\max(z, 6), 0)$ and constant scaling $z/\tau$ which is non-trainable BatchNorm represented in Fig. B.1, to be able to replace $\tanh_\tau$ in Prob. Appendix B.1 and Table B.1 provide experimental results indicating that these alternatives also improve uncertainty as well as accuracy. The results suggest that the bounded functions in front of Blur help ensemble of spatial information. In addition, given the similarity between Prob represented by Eq. (7) and activation ReLU $\circ$ BatchNorm, they imply that it is effective to place the activations before convolution layers which is an alternative to Blur. Experiments show that this pre-activation arrangement improves not only accuracy but uncertainty. See Appendix C.2 and Appendix C.3 for detailed discussions of pre-activation and ReLU6.

As discussed above, we take the perspective that each point in feature map is a prediction for binary classification by deriving the Bernoulli distributions from the feature map. It is in contrast to previous works known as sampling-free BNNs \cite{18, 40, 43} attempting to approximate the distribution of feature map with one Gaussian distribution. We do not use any assumptions on the distribution of feature map, and exactly represent the Bernoulli distributions and their averages. However, sampling-free BNNs are error-prone because there is no guarantee that feature maps will follow a Gaussian distribution.

Blur: Averaging neighboring probabilities. Blur averages the probabilities from feature map. In this work, we mainly use the average pool with kernel size of 2 and stride of 1 as the implementation of Blur for the sake of simplicity. Nevertheless, we can generalize it by using depth-wise convolution, which acts on each input channel separately, with non-trainable kernel

$$K = \frac{1}{||k||^2} k \otimes k^\top$$  \hspace{1cm} (8)

where $k$ is a one-dimensional matrix, e.g. $k \in \{(1), (1,1), (1,2,1), (1,4,6,4,1)\}$. Different $k$ derives different importances for nearby feature map points as shown in Fig. B.4.
Loss landscape. How does the spatial smoothing helps optimization? BNNs have two types of uncertainties: One is model uncertainty and the other is data uncertainty [29]. These randomness hinder and destabilize NN training. We show that the spatial smoothing, a method that ensembles feature maps, reduces the randomness and helps NN to learn.

We investigate model and data uncertainties over depth to show that the spatial smoothing reduces the randomness. Figure 5 shows the model uncertainty and data uncertainty of Bayesian ResNet including MC dropout layers. In this figure, the uncertainties tend to increase as the depth of NN increases, and the spatial smoothing layers reduce both of them. As a result, the model with the spatial smoothing layers has less uncertainties than models without them. Figure B.5 also shows consistent result, i.e., the spatial smoothing layer close to the last layer significantly improves performance. In addition, deterministic NNs do not have model uncertainty but data uncertainty. Therefore, the spatial smoothing improves the performance of deterministic NNs as well as Bayesian NNs.

Loss landscape perspective provides an evidence that the spatial smoothing help train NNs. Figure 6 visualizes the loss landscapes, the contours of NLL on training dataset, by using filter normalization [24]. In Fig. 6a and Fig. 6b, MC dropout prevents overfitting of the model, but the loss landscape becomes chaotic and irregular due to the randomness of MC dropout. As Li et al. [24] pointed out, it may lead to poor generalization and predictive performance. The spatial smoothing reduces randomness as discussed above, and the spatial smoothing aids in optimization by smoothing and flattening the loss landscape of BNN as shown in Fig. 6c. Furthermore, from these observations and experiments, we propose a conjecture that the smoother the loss landscape, the better the uncertainty estimation, and vice versa.

Revisiting GAP. The success of GAP classifier in image classification is indisputable. In terms of predictive performance, the original motivation and the most widely accepted explanation for the success is that GAP prevents overfitting by using far fewer parameters than multi-layer perceptron (MLP) [25]. Additionally, global max pooling (GMaxP), which uses the same number of parameters as GAP, shows an accuracy comparable to that of GAP [48, 6].

However, we find out that the explanation is poorly supported. Appendix C.1 and Table C.1 compares GAP with other classifiers including MLP on CIFAR-100. On the contrary to the explanation, the results suggest that MLP do not overfit the training dataset. On the test dataset, GAP provides better results compared with MLP. The accuracy of GMaxP is comparable to that of GAP, which is consistent with prior works; however, in terms of NLL and ECE, the uncertainty of GMaxP is worse than that of GAP.
We argue that GAP is an extreme case of the spatial smoothing. In other words, the reason for the success of GAP is that it spatially ensembles the feature map and smoothes the loss landscape to help optimization. Figure 7 visualizes the loss landscape of MLP classifier. It is irregular and chaotic compared with that of GAP represented by Fig. 6b. In addition, the performance of global median pooling (GMedP), another noise reduction method, is better than that of GMaxP. Likewise, as shown in Fig. C.2, GAP is robust against corrupted data, but MPL is not. In conclusion, we infer that averaging feature map tends to help neural network optimization. For more detailed discussion, see Appendix C.1.

3 Experiments

This section presents two experiments and their results. The first experiment is image classification. It shows that the spatial smoothing improves not only ensemble efficiency, but also accuracy, uncertainty, and robustness of deterministic NN and MC dropout simultaneously. The second experiment is semantic segmentation on data streams. It shows that the spatial smoothing and the temporal smoothing [29] are complementary. Refer to Appendix A for more detailed configurations.

We measure three metrics in these experiments: NLL(↓), accuracy (↑), and expected calibration error (ECE, ↓) [14]. NLL represents both accuracy and uncertainty, and is the most widely used as a proper scoring rule. ECE measures discrepancy between accuracy and confidence.

3.1 Image Classification

We mainly discuss ResNet-18 [15] on CIFAR-10 and CIFAR-100 [22], and ResNet-50 on ImageNet [33]. In Appendix D.1, we also discuss other models, e.g. VGG [36], ResNeXt [44], and pre-activation models [15], which show the same trend as ResNet. The spatial smoothing also improves deep ensemble [23], another non-Bayesian probabilistic NN method. See Appendix D.1.

Performance. Fig. 3 and Fig. 8 show predictive performances of ResNet-18 on CIFAR-100 and ResNet-50 on ImageNet, indicating the spatial smoothing improves both accuracy and uncertainty in many respects. To be specific, first of all, the spatial smoothing improves efficiency of ensemble size. In these examples, NLL of “MC dropout + spatial smoothing” with ensemble size of 2 is comparable to or even better than that of MC dropout with ensemble size of 50. In other words, “MC dropout + spatial smoothing” is $25 \times$ faster than MC dropout with similar predictive performance. Second, the predictive performance of “MC dropout + spatial smoothing” is better than that of MC dropout, at ensemble size of 50. Third, the spatial smoothing improves the predictive performance of deterministic NN, as well as MC dropout, as we would expect. Fig. 9 is the reliability diagram of ResNet-18 on CIFAR-100, which shows that the spatial smoothing also improves deep ensemble [23], another non-Bayesian probabilistic NN method. See Appendix D.1.

[1] We use arrows to indicate which direction is better.
Figure 10: Predictive performance of ResNet-18 on CIFAR-100-C. In the top row, we use an ensemble size of fifty for MC dropout with and without the spatial smoothing.

smoothing improves the uncertainty of both deterministic and Bayesian NNs. Numerical comparisons are provided in Appendix D.1 and Table D.1.

A peculiarity of the results on ImageNet is that the spatial smoothing degrades ECE of ResNet-50. It is because the spatial smoothing significantly improves the accuracy in this case, and there tends to be a trade-off between accuracy and ECE, e.g. as shown in [14], Fig. A.1 and Fig. B.3 Instead, the spatial smoothing shows the improvement in NLL, another uncertainty metric.

Robustness. To evaluate robustness against data corruption, we measure predictive performance on CIFAR-100-C [17], which consist of data corrupted by 15 different types with 5 levels of intensity each. The top row of Fig. 10 shows the results as a box plot. They reveal that the spatial smoothing improves predictive performance for corrupted data. In particular, the spatial smoothing undoubtedly helps in predicting reliable uncertainty.

To summarize the performance of corrupted data in a single value, we use mean corruption NLL (mCNLL, ↓), mean corruption error (mCE, ↓) [17], and mean corruption ECE (mCECE, ↓). Informally, they are the averages of NLL, error, and ECE over intensities and corruption types, respectively. See Eqs. (11) to (13) for strict definitions. The bottom row of Fig. 10 shows that the spatial smoothing improves not only the efficiency but corruption robustness across a whole range of ensemble size.

Consistency. Neural networks for image classification should be invariant against shift-transformation, but they are not [9, 2]. We measure consistency (↑) [17] [47], a metric to evaluate the shift-invariance of models, and the results show that the spatial smoothing improves it. See Eq. (14) for a strict definition of consistency. Qualitative and quantitative results are provided in Appendix D.

3.2 Semantic Segmentation

We use U-Net [32] as the backbone architecture for the semantic segmentation experiment. Bayesian U-Net [32] includes six MC dropout layers and four spatial smoothing layers in front of the subsampling layers in the encoder. See Appendix A for more detailed configurations.

Table summarizes the results on CamVid dataset [4] consisting of real-world 360×480 pixels videos. The table shows that spatial smoothing improves predictive performance, which is consistent with the image classification experiment. Moreover, it reveals that the spatial smoothing and the temporal smoothing [29] are complementary. See Appendix D.2 and Table D.3 for more details.
Table 1: Predictive performance of MC dropout in semantic segmentation on CamVid for each method. SPAT and TEMP is the spatial smoothing and the temporal smoothing, respectively. CONS is consistency. The numbers in brackets denote the performance improvement over the baseline.

| SPAT | TEMP | NLL   | ACC (%) | ECE (%) | CONS (%) |
|------|------|-------|---------|---------|----------|
| ✓    | ✓    | 0.260 (-0.038) | 92.6 (+0.1) | 2.71 (-1.49) | 96.5 (+1.1) |
| ✓    | ✓    | 0.273 (-0.025) | 92.6 (+0.1) | 3.23 (-0.97) | 96.4 (+1.0) |
| ✓    | ✓    | 0.284 (-0.014) | 92.6 (+0.1) | 3.96 (-0.24) | 95.6 (+0.2) |
| ✓    | ✓    | 0.298 (-0.000) | 92.5 (+0.0) | 4.20 (-0.00) | 95.4 (+0.0) |

4 Related Work

The spatial smoothing can be compared with prior works in the following areas.

**Anti-aliased CNNs.** Local means [47, 49, 38] were introduced for the shift-invariance of deterministic CNNs in image classification. Their motivations were to prevent the aliasing effect of subsampling. Although the local filtering can result in loss of information, Zhang [47] experimentally observed an increase in accuracy, which was beyond expectation. We provide a fundamental explanation for this phenomenon: *Local mean is a spatial ensemble.* This ensemble results in improving not only accuracy, but also uncertainty and robustness of deterministic and Bayesian NNs. In Appendix D.1 we also find that the predictive performance improvement is not due to anti-aliasing of local mean. For a discussion of non-local means [41] and Transformers [7], see Section 5.

**Sampling-free BNNs.** Sampling-free BNNs [18, 40, 43] predict results based on one or a couple of NN executions. To do so, they assumed that posterior and feature maps follow Gaussian distributions. However, the discrepancy between reality and the assumption accumulates in every NN layer. As a result, to the best of our knowledge, most of the sampling-free BNNs could only be applied to shallow models such as LeNet, and were tested on small datasets. Postels et al. [31] applied sampling-free BNNs to SegNet; nonetheless, Park et al. [29] argued that it does not predict well-calibrated results.

**Efficient deep ensembles.** Deep ensemble [23, 10], a non-Bayesian probabilistic NN, is another approach to predicting reliable results. Batch Ensemble [42, 8] ensembles over a low-rank subspace to make deep ensemble more efficient. Depth uncertainty network [11] aggregates feature maps from different depths in one NN to predict the results efficiently. Despite being robust against data corruption, it provides poor predictive performance compared with deterministic NN as well as MC dropout.

5 Conclusion

We propose probabilistic spatial smoothing, a method that ensembles spatially neighboring feature maps. This tiny module is easy to implement and can be easily inserted into various convolutional neural networks for vision tasks. The results of image classification experiments show that the spatial smoothing improves accuracy, uncertainty estimation, and robustness of both deterministic and Bayesian NNs, without any additional training variables. In particular, Bayesian NNs using the spatial smoothing achieve high predictive performance merely with a handful of ensembles. We find the fundamental implication that spatial ensembles of feature maps—even GAP—helps in NN optimization by smoothing and flattening the loss landscape.

The limitation of the spatial smoothing is that designing its components requires inductive bias. In other words, the optimal shape is model-dependent. We believe this problem can be solved by introducing self-attention [39]. Transformers for vision tasks [7, 37, 5] can be deemed as trainable importance-weighted ensembles of feature maps. The observation that Transformers are more robust than expected [3] supports this claim. Therefore, using Transformers to generalize the spatial smoothing would be a promising future work because it not only expands our work, but also helps deepen our understanding of self-attention.
References

[1] Javier Antorán, James Urquhart Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. arXiv preprint arXiv:2006.08437, 2020.

[2] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? Journal of Machine Learning Research, 20(184):1–25, 2019.

[3] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. arXiv preprint arXiv:2103.14586, 2021.

[4] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In European conference on computer vision, pages 44–57. Springer, 2008.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In European Conference on Computer Vision, pages 213–229. Springer, 2020.

[6] Vincent Christlein, Lukas Spranger, Mathias Seuret, Anguelos Nicolaou, Pavel Král, and Andreas Maier. Deep generalized max pooling. In 2019 International Conference on Document Analysis and Recognition (ICDAR), pages 1090–1096. IEEE, 2019.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[8] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In International conference on machine learning, pages 2782–2792. PMLR, 2020.

[9] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In International Conference on Machine Learning, pages 1802–1811. PMLR, 2019.

[10] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. arXiv preprint arXiv:1912.02757, 2019.

[11] Jonathan Frankle, David J Schwab, and Ari S Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. arXiv preprint arXiv:2003.00152, 2020.

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

[13] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017.

[14] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In International Conference on Machine Learning, pages 1321–1330. PMLR, 2017.

[15] 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, pages 770–778, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In European conference on computer vision, pages 630–645. Springer, 2016.

[17] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261, 2019.
[18] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In International Conference on Machine Learning, pages 1861–1869. PMLR, 2015.

[19] A Kendall, V Badrinarayanan, and R Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In British Machine Vision Conference 2017, BMVC 2017, 2017.

[20] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In Advances in neural information processing systems, pages 5574–5584, 2017.

[21] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. Unpublished manuscript, 40(7):1–9, 2010.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Advances in neural information processing systems, pages 6402–6413, 2017.

[24] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. arXiv preprint arXiv:1712.09913, 2017.

[25] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.

[26] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. IEEE Robotics and Automation Letters, 5(2):3153–3160, 2020.

[27] A Malinin and M Gales. Predictive uncertainty estimation via prior networks. In NIPS ’18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, volume 31, pages 7047–7058. Curran Associates, Inc., 2018.

[28] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. In Advances in Neural Information Processing Systems, pages 13991–14002, 2019.

[29] Namuk Park, Taekyu Lee, and Songkuk Kim. Vector quantized bayesian neural network inference for data streams. In AAAI Conference on Artificial Intelligence, 2021.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703, 2019.

[31] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2931–2940, 2019.

[32] 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, pages 234–241. Springer, 2015.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520, 2018.
[35] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? Advances in neural information processing systems, (31), 2018.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877, 2020.

[38] Cristina Vasconcelos, Hugo Larochelle, Vincent Dumoulin, Nicolas Le Roux, and Ross Goroshin. An effective anti-aliasing approach for residual networks. arXiv preprint arXiv:2011.10675, 2020.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

[40] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Natural-parameter networks: A class of probabilistic neural networks. arXiv preprint arXiv:1611.00448, 2016.

[41] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7794–7803, 2018.

[42] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. arXiv preprint arXiv:2002.06715, 2020.

[43] Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, Jose Miguel Hernandez-Lobato, and Alexander L Gaunt. Deterministic variational inference for robust bayesian neural networks. arXiv preprint arXiv:1810.03958, 2018.

[44] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1492–1500, 2017.

[45] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages 7343–7353, 2018.

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.

[47] Richard Zhang. Making convolutional networks shift-invariant again. In International Conference on Machine Learning, pages 7324–7334. PMLR, 2019.

[48] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2921–2929, 2016.

[49] Xueyan Zou, Fanyi Xiao, Zhiding Yu, and Yong Jae Lee. Delving deeper into anti-aliasing in convnets. arXiv preprint arXiv:2008.09604, 2020.
A Experimental Setup and Datasets

We obtain the main experimental results with the Intel Xeon W-2123 Processor, 32GB memory, and a single GeForce RTX 2080 Ti for CIFAR [22] and CamVid [4]. For ImageNet [33], we use AMD Ryzen Threadripper 3960X 24-Core Processor, 256GB memory, and four GeForce RTX 2080 Ti. We conduct ablation studies with four Intel Broadwell CPUs, 15GB memory, and a single NVIDIA T4. Models are implemented in PyTorch [30]. The detailed configurations of image classification and semantic segmentation are as follows.

A.1 Image Classification

We use VGG [36], ResNet [15], pre-activation ResNet [15], and ResNeXt [44] in image classification. According to the structure suggested by Zagoruyko and Komodakis [46], each block of Bayesian NNs contains one MC dropout layer.

NNs are trained using categorical cross-entropy loss and SGD optimizer with initial learning rate of 0.1, momentum of 0.9, and weight decay of $5 \times 10^{-4}$. We also use multi-step learning rate scheduler with milestones at 60, 130, and 160, and gamma of 0.2 on CIFAR, and with milestones at 30, 60, and 80, and gamma of 0.2 on ImageNet. We train NNs for 200 epochs with batch size of 128 on CIFAR, and for 90 epochs with batch size of 256 on ImageNet. We start training with gradual warmup [13] for 1 epoch. Basic data augmentation namely random cropping and horizontal flipping are used. One exception is the training of ResNeXt on ImageNet. In this case, we use the batch size of 128 and learning rate of 0.05 because of memory limitation.

We use hyperparameters that minimizes NLL of ResNet. Table A.1 provides hyperparameters for deterministic and Bayesian NNs. For fair comparison, models with and without the spatial smoothing share hyperparameters such as MC dropout rate. However, Fig. A.1 shows that the spatial smoothing improves predictive performance at all dropout rates. The default ensemble size of MC dropout is 50. For CIFAR, we report averages of three evaluations, and error bars in figures represent min and max values. For ImageNet, the report metrics are the mean from last three epochs. Standard deviations are omitted from tables for better visualization. See the source code released on GitHub for other details.

A.2 Semantic Segmentation

We use U-Net [32] in semantic segmentation. Following Bayesian SegNet [19], Bayesian U-Net contains six MC dropout layers. We add the spatial smoothing before each subsampling layer in U-Net encoder. We use 5 previous predictions and decay rate of $e^{-0.8}$ for the temporal smoothing.

CamVid consists of $720 \times 960$ pixels road scene video sequences. We resize the image bilinearly to $360 \times 480$ pixels. We use a list reduced to 11 labels by following previous works, e.g. [20].

NNs are trained using categorical cross-entropy loss and Adam optimizer with initial learning rate of 0.001 and $\beta_1$ of 0.9, and $\beta_2$ of 0.999. We train NN for 130 epoch with batch size of 3. The learning rate decreases to 0.0002 at the 100 epoch. Random cropping and horizontal flipping are used for
Table A.1: Hyperparameters of models for image classification.

| DATASET     | MODEL     | MC DROPOUT RATE (%) | $|k|$ | TEMPERATURE |
|-------------|-----------|---------------------|------|-------------|
|             |           |                     |      |             |
| CIFAR-10 & CIFAR-100 | VGG | 30 | 2 | 10 |
|             | ResNet    | 30 | 2 | 10 |
|             | Preact-ResNet | 30 | 2 | 10 |
|             | ResNeXt   | 30 | 2 | 10 |
|             | ImageNet  | 5  | 2 | 10 |
|             |            | 5  | 2 | 10 |
|             |            | 5  | 2 | 10 |

data augmentation. Median frequency balancing is used to mitigate dataset imbalance. Other details follow Park et al. \[29\].

B Ablation Study

The probabilistic spatial smoothing proposed in this paper consists of two components: Prob and Blur. This section explores several candidates for each component and their properties.

B.1 Prob: Feature maps to probabilities

We define Prob as a composition of an upper-bounded function and ReLU, a function that imposes the lower bound of zero. Fig. B.1 shows widely used upper-bounded functions: $\tanh_r(x) = \tau \tanh(x/\tau)$, $\text{ReLU}_6(x) = \min(\max(x, 6), 0)$, and constant scaling which is $x/\tau$.

Table B.1 shows the predictive performance improvement by Prob with various upper-bounded functions on

Figure B.1: Upper-bounded functions for an element of Prob.
Table B.1: Predictive performance of MC dropout with various Probs on CIFAR-100.

| MODEL  | SMOOTH | NLL     | ACC (%) | ECE (%) |
|--------|--------|---------|---------|---------|
| VGG-16 |        |         |         |         |
|        |        | 1.133 (-0.000) | 68.8 (+0.0) | 3.66 (+0.00) |
| ReLU  o tanh | 1.064 (-0.069) | 70.4 (+1.6) | 2.99 (-0.67) |
| ReLU  o RelU6 | 1.093 (-0.040) | 69.8 (+1.0) | 4.26 (+0.60) |
| ReLU  o Constant | 0.995 (-0.138) | 72.5 (+3.7) | 2.11 (-1.55) |
|        | Blur   | 0.985 (-0.000) | 72.4 (+0.0) | 1.77 (+0.00) |
|        | Blur  o ReLU  o tanh | 0.984 (-0.001) | 72.7 (+0.3) | 2.07 (+0.30) |
|        | Blur  o ReLU  o RelU6 | 0.982 (-0.003) | 72.5 (+0.1) | 1.84 (+0.07) |
|        | Blur  o ReLU  o Constant | 0.991 (+0.005) | 72.9 (+0.5) | 1.03 (-0.74) |
| VGG-19 |        |         |         |         |
|        |        | 1.215 (-0.000) | 67.3 (+0.0) | 6.37 (+0.00) |
| ReLU  o tanh | 1.131 (-0.084) | 69.2 (+1.9) | 5.23 (-1.14) |
| ReLU  o RelU6 | 1.166 (-0.049) | 68.3 (+1.0) | 6.44 (-0.06) |
| ReLU  o Constant | 0.997 (-0.218) | 72.5 (+5.2) | 1.09 (-5.29) |
|        | Blur   | 1.039 (-0.000) | 71.1 (+0.0) | 3.12 (+0.00) |
|        | Blur  o ReLU  o tanh | 1.034 (-0.005) | 71.3 (+0.2) | 3.31 (+0.19) |
|        | Blur  o ReLU  o RelU6 | 1.038 (-0.002) | 71.3 (+0.2) | 3.84 (+0.72) |
|        | Blur  o ReLU  o Constant | 0.995 (-0.045) | 72.3 (+1.2) | 1.41 (-1.71) |
| ResNet-18 |        |         |         |         |
|        |        | 0.848 (-0.000) | 77.3 (+0.0) | 3.01 (+0.00) |
| ReLU  o tanh | 0.838 (-0.010) | 77.7 (+0.4) | 2.92 (-0.08) |
| ReLU  o RelU6 | 0.844 (-0.004) | 77.4 (+0.1) | 2.74 (-0.27) |
| ReLU  o Constant | 0.825 (-0.023) | 77.7 (+0.4) | 1.87 (-1.14) |
|        | Blur   | 0.806 (-0.000) | 78.6 (+0.0) | 2.56 (+0.00) |
|        | Blur  o ReLU  o tanh | 0.801 (-0.005) | 78.9 (+0.3) | 2.56 (-0.01) |
|        | Blur  o ReLU  o RelU6 | 0.805 (-0.001) | 78.9 (+0.2) | 2.59 (+0.03) |
|        | Blur  o ReLU  o Constant | 0.811 (+0.005) | 78.5 (-0.2) | 1.84 (-0.72) |
| ResNet-50 |        |         |         |         |
|        |        | 0.822 (-0.000) | 79.1 (+0.0) | 6.63 (+0.00) |
| ReLU  o tanh | 0.812 (-0.010) | 79.3 (+0.2) | 6.74 (+0.11) |
| ReLU  o RelU6 | 0.799 (-0.023) | 79.4 (+0.3) | 6.71 (+0.08) |
| ReLU  o Constant | 0.788 (-0.034) | 79.6 (+0.5) | 5.22 (-1.41) |
|        | Blur   | 0.798 (-0.000) | 80.0 (+0.0) | 7.21 (+0.00) |
|        | Blur  o ReLU  o tanh | 0.800 (+0.002) | 80.1 (+0.1) | 7.25 (+0.04) |
|        | Blur  o ReLU  o RelU6 | 0.800 (+0.002) | 80.2 (+0.2) | 7.30 (+0.09) |
|        | Blur  o ReLU  o Constant | 0.779 (-0.019) | 80.4 (+0.4) | 5.81 (-1.40) |
The characteristics of temperature-scaled tanh depends on $\tau$. Figure B.2 plots $\tanh_\tau$ and their first derivatives with various temperatures. As shown in this figure, $\tanh_\tau$ has a couple of useful properties. First, $\tanh_\tau$ has an upper bound of $\tau$. Second, the first derivative of $\tanh_\tau$ at $x = 0$ does not depend on $\tau$.

Fig. B.3 shows the predictive performance of ResNet with MC dropout and the spatial smoothing for the temperature on CIFAR-100. In this figure, the accuracy increases as the temperature increases. In terms of ECE, NN predicts more underconfident results as $\tau$ decreases. It is a misinterpretation that the result is overconfident at low $\tau$ because ECE is high. By definition, ECE relies on the absolute value of the difference between confidence and accuracy. In this example, at low $\tau$, the accuracy is greater than the confidence, which leads to a high ECE. Moreover, at $\tau = 0.2$, ECE with $N = 50$ is greater than that with $N = 1$, which means that the result is severely underconfident. NLL, a metric representing both accuracy and uncertainty, is minimized when the accuracy and the uncertainty are balanced.
(a) $k = (1)$
(b) $k = (1, 1)$
(c) $k = (1, 2, 1)$
(d) $k = (1, 4, 6, 4, 1)$

Figure B.4: Kernels for Blur. Brighter background indicates higher importance.

Table B.2: Predictive performance of MC dropout using the spatial smoothing with various size of Blur kernels on CIFAR-100.

| Model      | $|k|$ | NLL   | Acc (%) | ECE (%) |
|------------|------|-------|---------|---------|
| VGG-16     | 1    | 1.087 (-0.000) | 69.8 (+0.0) | 3.43 (-0.00) |
|            | 2    | 1.034 (-0.053) | 71.4 (+1.6) | 1.06 (-2.37) |
|            | 3    | 0.986 (-0.101) | 72.7 (+2.9) | 1.03 (-2.40) |
|            | 5    | 1.018 (-0.069) | 72.0 (+2.2) | 1.32 (-2.11) |
| VGG-19     | 1    | 1.096 (-0.000) | 69.8 (+0.0) | 4.74 (-0.00) |
|            | 2    | 1.071 (-0.025) | 70.4 (+0.6) | 2.15 (-2.59) |
|            | 3    | 1.026 (-0.070) | 71.9 (+2.1) | 2.56 (-2.18) |
|            | 5    | 1.032 (-0.064) | 71.6 (+1.8) | 2.16 (-2.58) |
| ResNet-18  | 1    | 0.840 (-0.000) | 77.6 (+0.0) | 2.63 (-0.00) |
|            | 2    | 0.801 (-0.039) | 78.9 (+1.4) | 2.56 (-0.07) |
|            | 3    | 0.822 (-0.018) | 78.7 (+1.1) | 2.86 (-0.23) |
|            | 5    | 0.837 (-0.003) | 78.4 (+0.8) | 3.05 (-0.42) |
| ResNet-50  | 1    | 0.814 (-0.000) | 79.5 (+0.0) | 6.56 (-0.00) |
|            | 2    | 0.806 (-0.008) | 80.0 (+0.5) | 7.35 (+0.79) |
|            | 3    | 0.796 (-0.019) | 79.9 (+0.4) | 7.38 (+0.82) |
|            | 5    | 0.816 (+0.001) | 79.4 (-0.1) | 7.38 (+0.82) |
Figure B.5: Predictive performance of ResNet-18 with one spatial smoothing after each stage on CIFAR-100. None indicates vanilla MC dropout.

B.2 Blur: Averaging neighboring probabilities

Blur is a depth-wise convolution with a kernel. The kernel given by Eq. (8) is derived from various ks such as $k \in \{(1), (1, 1), (1, 2, 1), (1, 4, 6, 4, 1)\}$. In these examples, if $|k|=1$, Blur is identity. If $|k|=2$, Blur is a box blur, which is used in the main experiments. If $|k|=3$ or $5$, Blur is an approximated Gaussian blur.

Table B.2 shows predictive performance of models using the spatial smoothing with the kernels on CIFAR-100. This results show that most kernels improve both accuracy and uncertainty. However, the most effective kernel size depends on the model. Refer to Appendix D.1 for discussion of model properties and the performance improvements.

B.3 Position of the Spatial Smoothing.

As shown in Fig. 5, the magnitude of uncertainty tends to increase as the depth increases. Therefore, we expect that the spatial smoothing close to the output layer will mainly drive performance improvement.

We investigate the predictive performance of models with MC dropout using only one spatial smoothing layer. Figure B.5 shows the predictive performance of ResNet-18 with one spatial smoothing after each stage on CIFAR-100. The results suggest that spatial smoothing after s3 is the most important for improving performance. Surprisingly, the spatial smoothing after s4 is the least important. This is because GAP, the most extreme case of the spatial smoothing, already exists there.

C Revisiting Prior Works

As mentioned in Section 2, prior works—namely, GAP, pre-activation, and ReLU6—are spacial cases of the spatial smoothing. This section discusses them in detail.

C.1 Global Average Pooling

The composition of GAP and a fully connected layer is the most popular classifier in classification tasks. The original motivation and the most widely accepted explanation for the success is that GAP classifier prevents overfitting because it uses significantly fewer parameters than MLP \cite{25}. To verify this claim, we measure the predictive performance of MLP, GAP, and global max pooling (GMaP), a classifier that uses the same number of parameters as GAP, on training dataset.

Predictive performance. Table C.1 shows the experimental results on the training and the test dataset of CIFAR-100, suggesting that the explanation is poorly supported. On both the training and the test dataset, most predictive performance of MLP is worse than that of GAP. It is a counterintuitive result meaning that MLP do not overfit the training dataset. In addition, the performance improvement by GAP is remarkable in VGG, which has irregular loss landscape. The predictive performance of GMaP is better than that of MLP, but worse than that of GAP. This shows that using
Table C.1: Predictive performance of MC dropout with various classifiers on CIFAR-100.

| Model   | Classifier | Train | Test |
|---------|------------|-------|------|
|         |            | NLL   | Err (%) | ECE (%) | NLL | Acc (%) | ECE (%) |
| VGG-16  | GAP        | 0.0852 | 0.461  | 6.75     | 1.030 | 72.3     | 3.24     |
|         | MLP        | 0.5492 | 13.1   | 13.8     | 1.133 | 68.8     | 3.66     |
|         | GMaxP      | 0.0846 | 0.470  | 6.67     | 1.050 | 72.2     | 3.60     |
|         | GMedP      | 0.0867 | 0.501  | 6.80     | 1.042 | 72.2     | 3.35     |
| VGG-19  | GAP        | 0.1825 | 2.50   | 10.4     | 1.035 | 71.9     | 1.46     |
|         | MLP        | 0.7144 | 17.7   | 14.8     | 1.215 | 67.3     | 6.37     |
|         | GMaxP      | 0.1939 | 2.85   | 10.6     | 1.063 | 71.5     | 2.10     |
|         | GMedP      | 0.1938 | 2.80   | 10.6     | 1.051 | 71.7     | 1.70     |
| ResNet-18 | GAP        | 0.0124 | 0.0287 | 1.19     | 0.841 | 77.5     | 2.92     |
|         | MLP        | 0.0076 | 0.0347 | 7.22     | 1.040 | 74.8     | 9.55     |
|         | GMaxP      | 0.0113 | 0.0233 | 1.41     | 0.905 | 76.3     | 5.23     |
|         | GMedP      | 0.0156 | 0.0347 | 1.46     | 0.889 | 76.4     | 5.03     |
| ResNet-50 | GAP        | 0.0061 | 0.0220 | 0.48     | 0.822 | 79.1     | 6.63     |
|         | MLP        | 0.0071 | 0.0370 | 8.53     | 1.029 | 76.9     | 11.8     |
|         | GMaxP      | 0.0074 | 0.0313 | 1.09     | 0.887 | 77.2     | 5.67     |
|         | GMedP      | 0.0053 | 0.0287 | 0.47     | 0.849 | 78.5     | 6.29     |

fewer parameters partially helps to improve predictive performance; however, it is insufficient to explain the predictive performance improvement by GAP. Finally, global median pooling (GMedP) provides better predictive performance than GMaxP. It implies that using other noise reduction methods instead of average pooling helps to improve predictive performance.

To understand the mechanism of GAP performance improvement, we investigate the loss landscape. Figure C.1 shows the loss landscape sequences of ResNet with MC dropout. In this figure, each sequence shares the bases, but it fluctuates due to the randomness of the MC dropout. Figure C.1a is the loss landscape of the model using MLP classifier instead of GAP classifier. The loss landscape is chaotic and irregular, resulting in hindering and destabilizing NN optimization. Figure C.1b is the loss landscape sequence of ResNet with GAP classifier. Since GAP ensembles all of the feature map points at the last stage, it flattens and stabilizes the loss landscape. Likewise, as shown in Figure C.1c, the spatial smoothing layers at the end of all stages also flattens and stabilizes the loss landscape. In conclusion, averaging feature map points tends to help neural network optimization by smoothing, flattening, and stabilizing the loss landscape. We observe a similar phenomenon for deterministic NNs.

In these experiments, we use MLP incorporating dropout layers with a rate of 50% as the classifier. Since the dropout is one of the factors that makes MLP underfit the training dataset, we also evaluate MLP using dropouts with a rate of 0%. Nevertheless, the results still shows that the predictive performance of MLP is worse than that of GAP on the training dataset. Moreover, it severely degrades predictive performance of ResNet on the test dataset.

**Robustness.** To evaluate the robustness of the classifiers, we measure the predictive performance of ResNet-18 using MC dropout with the classifiers on CIFAR-100-C. Figure C.2 shows the experimental results. This figure suggests that MLP is not robust against data corruption, as we would expect. In terms of accuracy, the robustness of GMaxP and GMedP is relatively comparable to that of GAP; however, in terms of uncertainty, GAP is the most robust. These are consistent results with other spatial smoothing experiments.
Figure C.1: Loss landscape sequences of ResNet-18 with MC dropout on CIFAR-100. Although each sequence shares the bases, it fluctuates due to the randomness of the MC dropout.

Figure C.2: Predictive performance of ResNet-18 using MC dropout with classifiers on CIFAR-100-C.
Table C.2: Predictive performance of models with pre-activation arrangement on CIFAR-100.

| MODEL  | MC DROPOUT | PRE-ACT | NLL       | ACC (%) | ECE (%) |
|--------|------------|---------|-----------|---------|---------|
| VGG-16 | ·          | ·       | 2.047 (-0.000) | 71.6 (+0.0) | 19.2 (-0.0) |
|        | ·          | ✓       | 1.827 (-0.219) | 72.5 (+0.9) | 19.8 (+0.6) |
|        | ✓          | ·       | 1.133 (-0.000) | 68.8 (+0.0) | 3.66 (-0.00) |
|        | ✓          | ✓       | 1.036 (-0.096) | 71.7 (+2.9) | 3.55 (-0.11) |
| VGG-19 | ·          | ·       | 2.016 (-0.000) | 67.6 (+0.0) | 21.2 (-0.0) |
|        | ·          | ✓       | 1.799 (-0.217) | 64.4 (-3.2) | 17.2 (-4.0) |
|        | ✓          | ·       | 1.215 (-0.000) | 67.3 (+0.0) | 6.37 (-0.00) |
|        | ✓          | ✓       | 1.084 (-0.131) | 70.1 (+3.7) | 4.23 (-2.14) |
| ResNet-18 | ·      | ·       | 0.983 (-0.000) | 77.1 (+0.0) | 7.75 (-0.00) |
|         | ·          | ✓       | 0.934 (-0.049) | 77.6 (+0.5) | 8.04 (+0.29) |
|         | ✓          | ·       | 0.937 (-0.000) | 76.9 (+0.0) | 5.11 (-0.00) |
|         | ✓          | ✓       | 0.872 (-0.065) | 77.6 (+0.7) | 5.53 (+0.42) |
| ResNet-50 | ·        | ·       | 0.880 (-0.000) | 79.0 (+0.0) | 8.35 (-0.00) |
|          | ·          | ✓       | 0.870 (-0.010) | 79.4 (+0.4) | 8.27 (-0.08) |
|          | ✓          | ·       | 0.831 (-0.000) | 78.6 (+0.0) | 6.06 (-0.00) |
|          | ✓          | ✓       | 0.819 (-0.012) | 79.5 (+0.9) | 6.29 (+0.23) |

C.2 Pre-activation

He et al. [16] experimentally showed that the pre-activation arrangement, in which the activation \( \text{ReLU} \circ \text{BatchNorm} \) is placed before the convolution, improves the accuracy of ResNet. Since \( \gamma \)s of most BatchNorms in CNNs are near-zero [11], BatchNorms reduce the magnitude of feature maps. As shown in Fig. B.1, constant scaling is a non-trainable BatchNorm with no bias, and it also reduces the magnitude of feature map. In Appendix B.1, we show that constant scaling improves predictive performance. Considering the similarity between \( \text{Prob} \) with constant scaling and conventional activation, i.e., the similarity between \( \text{ReLU} \circ \text{ConstantScaling} \) and \( \text{ReLU} \circ \text{BatchNorm} \), we find that the pre-activation arrangement improves uncertainty as well as accuracy, because convolutions act as a Blur.

To show this, we change the post-activation of all layers to pre-activation, and measure the predictive performance. For ResNet, we follow the original paper by He et al. [16]. Table C.2 shows the predictive performance of models with pre-activation. The results suggest that pre-activation improves both accuracy and uncertainty in most cases. For deterministic VGG-19, pre-activation significantly degrades accuracy but improves NLL. In conclusion, they imply that pre-activation is a special case of the spatial smoothing.

Santurkar et al. [35] argued that BatchNorm helps in optimization by flattening the loss landscape. We show that the spatial smoothing flattens and smoothens the loss landscape, which is a consistent explanation. It will be interesting to investigate if BatchNorm helps in ensembling feature maps.

C.3 ReLU6

ReLU6 was experimentally introduced to improve predictive performance [21]. Sandler et al. [34] used “ReLU6 as the non-linearity because of its robustness when used with low-precision computation”. In Appendix B.1, we show that ReLU6s at the end of stages helps to ensemble spatial information by transforming the feature map to Bernoulli distributions. Since the spatial smoothing improves robustness against data corruption, it seems reasonable that ReLU6 is robust to low-precision computation. A more abundant investigation into this topic is promising future works.
Table C.3: Predictive performance of models using ReLU6 instead of ReLU on CIFAR-100.

| Model       | MC Dropout | ReLU6 | NLL (%) | Acc (%) | ECE (%) |
|-------------|------------|-------|---------|---------|---------|
| VGG-16      | ·          | ·     | 2.047 (+0.000) | 71.6 (+0.0) | 19.2 (-0.0) |
|             | ✓          |       | 2.051 (+0.004) | 70.1 (-1.5) | 19.8 (+0.6) |
|             | ✓          | ✓     | 1.133 (+0.000) | 68.8 (+0.0) | 3.66 (-0.00) |
| VGG-19      | ·          | ·     | 2.016 (+0.000) | 67.6 (+0.0) | 21.2 (-0.0) |
|             | ✓          |       | 1.977 (-0.039) | 64.2 (-3.4) | 17.1 (-4.1) |
|             | ✓          | ✓     | 1.215 (+0.000) | 67.3 (+0.0) | 6.37 (-0.00) |
| ResNet-18   | ·          | ·     | 0.886 (+0.000) | 77.9 (+0.0) | 4.97 (-0.00) |
|             | ✓          |       | 0.888 (+0.002) | 77.9 (+0.0) | 4.85 (-0.12) |
|             | ✓          | ✓     | 0.848 (+0.000) | 77.3 (+0.0) | 3.01 (-0.00) |
| ResNet-50   | ·          | ·     | 0.835 (+0.000) | 79.9 (+0.0) | 8.88 (-0.00) |
|             | ✓          |       | 0.856 (+0.021) | 79.6 (-0.3) | 8.78 (-0.10) |
|             | ✓          | ✓     | 0.822 (+0.000) | 79.1 (+0.0) | 6.63 (-0.00) |

We measure the predictive performance of NNs using all activations as ReLU6 instead of ReLU. Table C.3 summarizes the results. In contrast to the results in Appendix B.1, these results are not consistent. We speculate that the reason is that a lot of ReLU6s overly regularize NNs.

D Extended Informations of Experiments

This section provides additional information on the experiments in Section 3.

D.1 Image Classification

We present numerical comparisons in the image classification experiment and discuss the results in detail.

Predictive performance. Table D.1 shows the predictive performance of various deterministic and Bayesian NNs with and without the spatial smoothing on CIFAR-10, CIFAR-100, and ImageNet. This table suggests the following: First, the spatial smoothing improves both accuracy and uncertainty in most cases. In particular, it improves the predictive performance of all models with MC dropouts. Second, the spatial smoothing significantly improves the predictive performance of VGG compared with ResNet. VGG has a chaotic loss landscape, which results in poor predictive performance [24], and the spatial smoothing smoothens its loss landscape effectively. Third, as the depth increases, the performance improvement decreases. Deeper NNs provide more overconfident results [14], but the number of spatial smoothing layers calibrating uncertainty is fixed. Last, the performance improvement of ResNeXt, which includes an ensemble in its internal structure, is relatively marginal.

Fig. D.1 shows predictive performance of MC dropout and deep ensemble for ensemble size. A deep ensemble with an ensemble size of 1 is a deterministic NN. This figure shows that the spatial smoothing improves efficiency of ensemble size and the predictive performance at ensemble size of 50. In addition, the spatial smoothing stabilizes NN training. It reduces the variance of the performance, especially in VGG.
Table D.1: Predictive performance of models with the spatial smoothing in image classification on CIFAR-10, CIFAR-100, and ImageNet.

| MODEL & DATASET | MC DROPOUT | SMOOTH | NLL (_%_) | ACC (_%) | ECE (_%) |
|-----------------|------------|--------|-----------|---------|---------|
| VGG-19 & CIFAR-10 | - | - | 0.401 (-0.000) | 93.1 (+0.0) | 3.80 (-0.0) |
| | - | ✓ | 0.376 (-0.002) | 93.2 (+0.1) | 5.49 (+1.69) |
| | ✓ | - | 0.238 (-0.000) | 92.6 (+0.0) | 3.55 (-0.00) |
| | ✓ | ✓ | 0.197 (-0.041) | 93.3 (+0.7) | 6.68 (-2.86) |
| ResNet-18 & CIFAR-10 | - | - | 0.182 (-0.000) | 95.2 (+0.0) | 2.75 (-0.00) |
| | - | ✓ | 0.173 (-0.009) | 95.4 (+0.2) | 2.31 (-0.44) |
| | ✓ | - | 0.157 (-0.000) | 95.2 (+0.0) | 1.14 (-0.00) |
| | ✓ | ✓ | 0.144 (-0.014) | 95.5 (+0.2) | 1.04 (-0.10) |
| VGG-16 & CIFAR-100 | ✓ | - | 2.047 (-0.000) | 71.6 (+0.0) | 19.2 (-0.0) |
| | ✓ | ✓ | 1.878 (-0.169) | 72.2 (+0.6) | 20.5 (+1.3) |
| ResNet-18 & CIFAR-100 | ✓ | - | 1.133 (-0.000) | 68.8 (+0.0) | 3.66 (-0.00) |
| | ✓ | ✓ | 1.034 (-0.099) | 71.4 (+2.6) | 1.06 (-2.60) |
| VGG-19 & CIFAR-100 | ✓ | - | 2.016 (-0.000) | 67.6 (+0.0) | 21.2 (-0.0) |
| | ✓ | ✓ | 1.851 (-0.165) | 71.7 (+4.0) | 2.15 (-4.22) |
| ResNet-18 & CIFAR-100 | ✓ | - | 0.886 (-0.000) | 77.9 (+0.0) | 4.97 (-0.00) |
| | ✓ | ✓ | 0.863 (-0.023) | 78.9 (+1.0) | 4.40 (-0.57) |
| ResNet-50 & CIFAR-100 | ✓ | - | 0.801 (-0.047) | 78.9 (+1.6) | 2.56 (-0.45) |
| | ✓ | ✓ | 0.814 (-0.000) | 79.9 (+0.0) | 8.88 (-0.00) |
| ResNet-50 & CIFAR-100 | ✓ | - | 0.835 (-0.000) | 79.9 (+0.0) | 8.88 (-0.00) |
| | ✓ | ✓ | 0.834 (-0.002) | 80.7 (+0.8) | 9.29 (+0.42) |
| ResNet-50 & CIFAR-100 | ✓ | - | 0.822 (-0.000) | 79.1 (+0.0) | 6.63 (-0.00) |
| | ✓ | ✓ | 0.800 (-0.022) | 80.1 (+1.0) | 7.25 (+0.62) |
| ResNet-50 & CIFAR-100 | ✓ | - | 0.804 (-0.000) | 80.6 (+0.0) | 8.23 (-0.00) |
| | ✓ | ✓ | 0.825 (+0.022) | 80.8 (+0.3) | 9.41 (+1.18) |
| ResNet-50 & CIFAR-100 | ✓ | - | 0.762 (-0.000) | 80.5 (+0.0) | 5.67 (-0.00) |
| | ✓ | ✓ | 0.759 (-0.002) | 80.7 (+0.2) | 6.62 (+0.94) |
| ResNet-18 & ImageNet | ✓ | - | 1.208 (-0.000) | 70.1 (+0.0) | 1.94 (-0.00) |
| | ✓ | ✓ | 1.177 (-0.031) | 70.7 (+0.6) | 1.23 (-0.71) |
| ResNet-50 & ImageNet | ✓ | - | 0.933 (-0.000) | 76.5 (+0.0) | 2.86 (-0.00) |
| | ✓ | ✓ | 0.910 (-0.023) | 77.0 (+0.5) | 3.23 (-0.37) |
| ResNet-50 & ImageNet | ✓ | - | 0.925 (-0.000) | 76.5 (+0.0) | 1.89 (-0.00) |
| | ✓ | ✓ | 0.904 (-0.021) | 77.1 (+0.6) | 2.28 (-0.39) |
| ResNet-50 & ImageNet | ✓ | - | 0.917 (-0.000) | 77.6 (+0.0) | 3.74 (-0.00) |
| | ✓ | ✓ | 0.899 (-0.018) | 78.0 (+0.4) | 4.18 (-0.44) |
| ResNeXt-50 & ImageNet | ✓ | - | 0.894 (-0.000) | 77.7 (+0.0) | 2.47 (-0.00) |
| | ✓ | ✓ | 0.887 (-0.007) | 78.2 (+0.5) | 3.37 (-0.90) |
Figure D.1: Predictive performance for ensemble size. Deep ensembles on ImageNet: TBD.
CNNs such as VGG, ResNet, and ResNeXt generally use post-activation arrangement. In other words, their stages end with BatchNorm and ReLU. Therefore, the spatial smoothing layers $\text{Smooth}(z) = \text{Blur} \circ \text{Prob}(z)$ in CNNs cooperates with BatchNorm and ReLU as follows:

$$\text{Prob}(z) = \text{ReLU} \circ \tanh_\tau \circ \text{ReLU} \circ \text{BatchNorm}(z)$$  \hspace{1cm} (9)

$$\text{Prob}(z) = \text{ReLU} \circ \tanh_\tau \circ \text{BatchNorm}(z)$$  \hspace{1cm} (10)

since ReLU and $\tanh_\tau$ are commutative, and $\text{ReLU} \circ \text{ReLU}$ is ReLU. This $\text{Prob}$ is trainable and is a general form of Eq. (7). If we only use $\text{Blur}$ as the spatial smoothing, the activations $\text{BatchNorm}$–ReLU play the role of $\text{Prob}$.

In order to analyze the roles of $\text{Prob}$ and $\text{Blur}$ more precisely, we measure the predictive performance of the model that does not use the post-activation. Figure D.2 shows NLL of pre-activation VGG-16 on CIFAR-100. The result shows that $\text{Blur}$ with $\text{Prob}$ improves the performance, but $\text{Blur}$ alone does not. In fact, contrary to [47], blur degrades the predictive performance since it results in loss of information. We also measure the performance of VGG-19, ResNet-18, ResNet-50, and BlurPool [47] with pre-activation, and observe the same phenomenon. As mentioned in Appendix C.2, pre-activation is a special case of the spatial smoothing. Therefore, the performance improvement of pre-activation by spatial smoothing is marginal compared to that of post-activation.

Robustness. We measure predictive performance on CIFAR-100-C [17] in order to evaluate the robustness of the models against 5 intensities and 15 types of data corruption. Hendrycks and Dietterich [17] introduced a corruption error (CE) for quantitative comparison. $CE^c_f$, which is CE for corruption type $c$ and model $f$, is as follows:

$$CE^c_f = \frac{\sum_{i=1}^{5} E^f_{i,c}}{\sum_{i=1}^{5} E^{\text{AlexNet}}_{i,c}}$$  \hspace{1cm} (11)

where $E^f_{i,c}$ is top-1 error of $f$ for corruption type $c$ and intensity $i$, and $E^{\text{AlexNet}}_{i,c}$ is the error of AlexNet. Mean CE or $mCE$ summarizes $CE^c_f$ by averaging them over 15 corruption types such as Gaussian noise, brightness, and show. Likewise, to evaluate robustness in terms of uncertainty, we introduce corruption NLL or $\text{CNLL}$ and corruption ECE or $\text{CECE}$ as follows:

$$\text{CNLL}^c_f = \frac{\sum_{i=1}^{5} \text{NLL}^f_{i,c}}{\sum_{i=1}^{5} \text{NLL}^{\text{AlexNet}}_{i,c}}$$  \hspace{1cm} (12)

and

$$\text{CECE}^c_f = \frac{\sum_{i=1}^{5} \text{ECE}^f_{i,c}}{\sum_{i=1}^{5} \text{ECE}^{\text{AlexNet}}_{i,c}}$$  \hspace{1cm} (13)

where $\text{NLL}^f_{i,c}$ and $\text{ECE}^f_{i,c}$ are NLL and ECE of $f$ for $c$ and $i$, respectively. $m\text{CNLL}$ and $m\text{CECE}$ are averages over corruption types. Experimental results show that the spatial smoothing improves the robustness against data corruption. See Fig. 10 for the results.

Consistency. To evaluate the translation invariance of models, we use consistency [17, 47], a metric representing translation consistency for shift-translated data sequences $S = \{x_1, \cdots, x_{M+1}\}$, as follows:

$$\text{Consistency} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}(g(x_i) = g(x_{i+1}))$$  \hspace{1cm} (14)
Table D.2: Consistency of ResNet-18 on CIFAR-10-P. Deterministic NN with \( N = 5 \) means deep ensemble.

| MC DROPOUT | SMOOTH | \( N \) | CONS (\%) | CEC |
|------------|--------|--------|-----------|-----|
| ✓          | ✓      | 1      | 97.9      | 0.0103 |
| ✓          | ✓      | 5      | 98.7      | 0.0122 |
| ✓          | ✓      | 50     | 98.2      | 0.0129 |
| ✓          | ✓      | 50     | 98.4      | 0.0134 |

where \( g(x) = \arg \max \) \( p(y|x, D) \). Table D.2 provides consistency of ResNet-18 on CIFAR-10-P [17]. The results show that MC dropout and deep ensemble improve consistency, and the spatial smoothing improves consistency of both deterministic and Bayesian NNs.

Prior works [47, 2] investigated the fluctuation of predictive confidence on shift-translated data sequence. However, surprisingly, we find that confidence fluctuation has little to do with consistency. To demonstrate this claim, we introduce cross-entropy consistency (CEC, ↓), a metric that represents the fluctuation of confidence on a shift-translated data sequence \( S = \{x_1, \cdots, x_{M+1}\} \), as follows:

\[
\text{CEC} = -\frac{1}{M} \sum_{i=1}^{M} f(x_i) \cdot \log(f(x_{i+1}))
\] (15)

where \( f(x) = p(y|x, D) \). In Table D.2, high consistency does not mean low CEC; conversely, high consistency tends to be high CEC. Canonical NNs predict overconfident probabilities, and their confidence sometimes changes drastically from near-zero to near-one. Correspondingly, it results in low consistency but low CEC. On the contrary, well-calibrated NNs such as MC dropout provide confidence that oscillates between zero and one, which results in high CEC.

To represent the NN reliability properly, we propose relative confidence (↑) as follows:

\[
\text{Relative confidence} = \frac{p(y_{\text{true}}|x, D)}{\max_{y} p(y|x, D)}
\] (16)

where \( \max_{y} p(y|x, D) \) is confidence of predictive result and \( p(y_{\text{true}}|x, D) \) is probability of the result for true label. It is 1 when NN classifies the image correctly, and less than 1 when NN classifies it incorrectly. Therefore, relative confidence is a metric that indicates the overconfidence of a prediction when NN’s prediction is incorrect.

Figure D.3 shows a qualitative example of consistency on CIFAR-10-P by using relative confidence. This figure suggests that the spatial smoothing improves consistency of both deterministic and Bayesian NN.

D.2 Semantic Segmentation

Table D.3 shows the performance of U-Net on the CamVid dataset. This table indicates that the spatial smoothing improves accuracy, uncertainty, and consistency of deterministic and Bayesian NNs. This is consistent with the results in image classification. In addition, the temporal smoothing leads to significant improvement in efficiency of ensemble size, accuracy, uncertainty, and consistency by...
Table D.3: Performance in semantic segmentation on CamVid dataset. \texttt{SPAT} and \texttt{TEMP} are the spatial smoothing and the temporal smoothing, respectively. \texttt{CONS} is consistency.

| MC DROPOUT | SPAT | TEMP | N   | NLL          | ACC  (\%) | ECE  (\%) | CONS  (\%) |
|------------|------|------|-----|--------------|----------|----------|----------|
| ✓          | ✓    | ✓    | 50  | 0.298 (+0.000) | 92.5 (+0.0) | 4.20 (+0.0) | 95.4 (+0.0) |
| ✓          | ✓    | ✓    | 50  | 0.284 (+0.014) | 92.6 (+0.1) | 3.96 (+0.24) | 95.6 (+0.2) |
| ✓          | ✓    | ✓    | 50  | 0.273 (+0.025) | 92.6 (+0.1) | 3.23 (+0.97) | 96.4 (+1.0) |

exploiting temporal information. Moreover, the temporal smoothing requires only one ensemble to achieve high predictive performance, since it cooperates with the temporally previous predictions. We obtain the best predictive and computational performance by using both the temporal smoothing and the spatial smoothing.