Universal Adversarial Perturbation via Prior Driven Uncertainty Approximation

Hong Liu[1], Rongrong Ji[12*], Jie Li[1], Baochang Zhang[3], Yue Gao[4], Yongjian Wu[5], Feiyue Huang[5]
[1]Department of Artificial Intelligence, School of Informatics, Xiamen University, [2]Peng Cheng Lab
[3]Beihang University, [4]Tsinghua University, [5]Tencent Youtu Lab

Abstract

Deep learning models have shown their vulnerabilities to universal adversarial perturbations (UAP), which are quasi-imperceptible. Compared to the conventional supervised UAPs that suffer from the knowledge of training data, the data-independent unsupervised UAPs are more applicable. Existing unsupervised methods fail to take advantage of the model uncertainty to produce robust perturbations. In this paper, we propose a new unsupervised universal adversarial perturbation method, termed as Prior Driven Uncertainty Approximation (PD-UA), to generate a robust UAP by fully exploiting the model uncertainty. Specifically, a Monte Carlo sampling method is deployed to activate more neurons to increase the model uncertainty for a better adversarial perturbation. Thereafter, a textural bias prior revealing a statistical uncertainty is proposed, which helps to improve the attacking performance. The UAP is crafted by the stochastic gradient descent algorithm with a boosted momentum optimizer, and a Laplacian pyramid frequency model is finally used to maintain the statistical uncertainty. Extensive experiments demonstrate that our method achieves well attacking performances on the ImageNet validation set, and significantly improves the fooling rate compared with the state-of-the-art methods.

1. Introduction

The success of deep learning models [11] have been witnessed in various computer vision tasks, such as image classification [14], instance segmentation [20] and objective detection [6]. However, existing deep models have shown to be sensitive to adversarial examples [2, 12, 31], *i.e.*, adding a perturbation to the input image. In general, given a convolutional neural network (CNN) $f(\boldsymbol{x})$, which maps the input image $\boldsymbol{x}$ to a class label $\boldsymbol{y}$, the target of the adversarial attacking is to find an optimal perturbation $\boldsymbol{\delta}$ to fool $f(\boldsymbol{x})$ as:

$$f(\boldsymbol{x}+\boldsymbol{\delta}) \neq f(\boldsymbol{x}),\ \text{s.t.}\ \|\boldsymbol{\delta}\|_p < \epsilon, \tag{1}$$

where ϵ is a positive number to control the magnitude of perturbation, and $\|\cdot\|_p$ is the p-norm.

The perturbation $\boldsymbol{\delta}$ is quasi-imperceptible, which is designed to fool the model to misclassify the perturbed image [1], or to force to output a wrong target class [13]. Various approaches have been proposed, such as model distilling [26], transfer learning [19] and gradient updating [1]. However, the effectiveness and efficiency remains an open problem in many practical applications, as such approaches are not *universal* and require specific and complex optimization algorithms for crafting the adversarial perturbation online.

Recently, universal adversarial perturbations (UAP) has been introduced in [21] that employs a single noise to adversarially perturb the corresponding CNN outputs for different images. UAP is also capable of conducting transfer attacking, *i.e.,* cross-model and cross-data attacking, which is suitable for both white-box and black-box attacking tasks [1, 19]. Two kinds of UAP methods, data-dependent and data-independent, are available for extensive applications. The data-dependent methods craft UAP using an objective function as shown in Eq.1, where both the training data and the model architecture must be known beforehand [21]. The performance of data-dependent methods is thereby sensitive to the number of training samples [15, 24]. On the contrary, the data-independent UAP is more flexible [23, 24], which only needs the model architecture and parameters, without knowing the training samples in use. By maximizing the activation of convolutional neurons based on a random Gaussian initialization, it can optimize the UAP by directly attacking the stability of a given CNN model. It is thus given much more attention than the data-dependent UAP.

In essence, UAP leverages the uncertainty of a CNN model to disturb its output reliability under input observations. Thus, the vital issue of UAP lies in how to estimate such model uncertainty, which innovates us to investigate data-independent UAP from a new perspective. It is also supported by recent works [7, 18, 29] that it is possible to obtain the estimation of model uncertainty by casting dropout technology in CNNs.

There exists two major types of uncertainty one can model: the *Epistemic uncertainty* and the *Aleatoric uncer-*

*R. Ji (rrji@xmu.edu.cn) is the corresponding author.

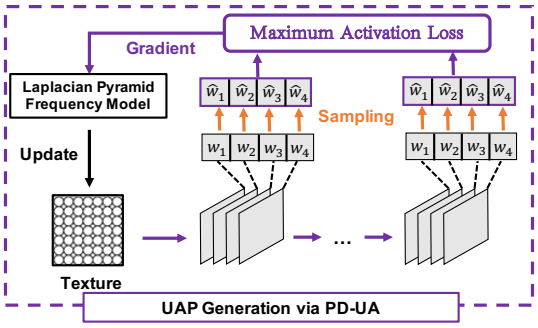


Figure 1. The framework of the proposed data-independent UAP. We employ a MC dropout to approximate the Epistemic uncertainty at each layer. A uncertainty activation loss is introduced with information flow represented in the purple lines. By combining all components, we employ a new gradient updating scheme to generate a robust data-independent UAP.

tainty [4]. The Epistemic uncertainty accounts for the uncertainty in the model parameters that are best fitted to the training data, such as ImageNet [3]. Intuitively, in our case, the uncertainty of a CNN can be reflected by the number of activated neurons at each convolutional layer. During the model output, the more credible neurons are activated, the more uncertainty is achieved, leading to a better UAP. To this end, virtual model uncertainty is introduced in our UAP learning, which aims at activating more neurons to increase the model uncertainty for each convolutional layer with the Monte Carlo Dropout (MC Dropout) [8].

The Aleatoric uncertainty is a data-independent but task-dependent uncertainty, which is a statistical uncertainty and represented as a quantity that stays stable for various input data but varies between different tasks. For a classification task as in [10], it is statistically demonstrated that a pre-trained model on ImageNet contains strong texture bias, which motivated us to use a texture image revealing the Aleatoric uncertainty to fool a CNN better. Accordingly, we introduce a texture bias as the information prior that is a constraint to model Aleatoric uncertainty during the perturbation learning. Since the texture image contains much low-frequency information [33], we introduce a Laplacian pyramid frequency model to improve the attacking performance with faster convergence effectively.

In this paper, we combine *Epistemic* and *Aleatoric* uncertainty in a unified framework, referred to as Prior-driven Uncertainty Approximation (PD-UA). Figure 1 presents the overall framework, which mainly innovates in the following three aspects:

1) To approximate the Epistemic uncertainty, we propose a virtual model uncertainty that makes our PD-UA mostly maximize the neuron activations corresponding to the perturbation input, which increase the uncertainty as well as improving the attacking performance.

2) To approximate the Aleatoric uncertainty, we are the first to introduce the texture bias to initialize UAP, which achieves a significant performance gain over state-of-the-arts in public benchmarks. We come to two basic and important conclusions: (a) A better initialization of the perturbation has a significant impact on the generative quality of UAP for deeper CNNs. (b) The texture-like perturbation can directly fool the CNNs without any training process.

3) We further propose a Laplacian pyramid frequency model to boost the gradient from the low-frequency part, whose output is employed efficiently to update the perturbation via SGD with momentum.

4) We compare the proposed method with the state-of-the-art data-independent UAPs on ImageNet dataset with six well-known CNN models, including GoogleNet, VGGs (VGG-F, VGG-16, VGG-19), and ResNets (both ResNet-50 and ResNet-150). Quantitative experiments demonstrate that our proposed PD-UA outperforms the state-of-the-art [23] with significant fooling rate improvement.

2. Related Work

Szegedy *et al.* [31] first observed that a neural network (especially CNNs) could be fooled by a specially structured perturbation that is quasi-imperceptible to human eyes. Later on, many gradient-based adversarial perturbation methods have been proposed, including but not limited to, Fast Gradient Sign Method [12], iterative-based Fast Gradient Sign Method [16] and Momentum Iterative Fast Gradient Sign Method [5]. Note that, as an underlying property, these methods are intrinsically data-dependent and model-dependent, and the adversarial examples are generally computed based on a complicated optimization, which is less practical in real-world applications.

Recent work in [21] has been demonstrated that a single adversarial noise, termed universal adversarial perturbation (UAP), is sufficient to fool most images from a data distribution with a given CNN model. Some data samples that have a similar distribution to the model's training data are needed to craft such a single perturbation using the Deepfool method [22] iteratively. It is therefore capable of adding such a UAP to the input image without any online optimization process, which shows promising practical values in various real-world applications [16, 17].

However, the performance relies heavily on the quality and quantity of training data [15, 24], where the fooling rate increases with the sample size that is indeed very expensive in practice. To handle these problems, some methods further craft UAPs with unsupervised or data-independent learning. The representative work that devoted to data-independent UAP has been mentioned in [23], which maximizes the activation of convolutional neurons based on a given deep architecture to optimize the UAP.

The methods mentioned above can be considered as a white-box attacking, where the data sample and the deep

model, or at least one of them, are known beforehand. Another line of research mainly focuses on the black-box attacking. To this end, the existing methods learn such perturbations based on evolutionary algorithm [13], transfer learning [19], or model distilling [26]. As shown in the previous works [21, 25], UAP has a strong ability to transfer attacking for different models, datasets, and computer vision tasks. Thus, UAP can also be used for black-box attacking in practice.

3. The Proposed Method

The adversarial perturbation fools the CNN model by increasing the predictive uncertainty of the model outputs. To this end, we propose a Prior Driven Uncertainty Approximation (PD-UA) method, which fully utilizes the uncertainty of a CNN model to learn a UAP. First, the Epistemic uncertainty of a CNN model becomes larger when more neurons are activated, which is overlooked in the field but can be beneficial to build robust UAP according to our observation. To this end, a Bernoulli distribution is introduced over the output of the neurons to approximate the uncertainty per layer, referred to as *virtual Epistemic uncertainty*. This process is done with the MC dropout method, as detailed in Section 3.1. Second, to better approximate the Aleatoric uncertainty that statistically reflects the inherent noise distribution, we introduce a texture bias as information prior that further increases the model uncertainty. This uncertainty significantly improves the attacking performance on the deeper CNN, such as ResNet, as detailed in Section 3.2. Finally, we combine these two uncertainties in a unified framework, which can be directly optimized via SGD. Besides, a Laplacian pyramid frequency model is introduced to normalize the gradient resulting in fast convergence. The model details and the corresponding optimization are elaborated in Section 3.3.

3.1. Virtual Epistemic Uncertainty

To capture the Epistemic uncertainty, we resort to the model uncertainty approximation with Bayesian probability theory [8]. The traditional Bayesian CNN [8] mainly models the Epistemic uncertainty via a dropout strategy to extract information from existing models. And the weight parameters of the Bayesian CNN is updated based on an SGD-based optimizer, which makes the output of the model be of higher confidence and lower uncertainty. Differently, we propose a new model to quantify the measure of uncertainty from the model's structural activation, rather than from the final probability output. We maximize such uncertainty by activating as many neurons at all convolutional layers as possible, which is termed as *virtual Epistemic uncertainty*. Our Epistemic uncertainty modeling differs from [7] in two-fold: *1*) Our goal is to increase the model uncertainty with a fixed perturbation. *2*) The data-independent hypothesis makes the output of CNN untrusted.

Formally, following the similar mathematical definitions in Eq.1, let $f^{\boldsymbol{W}_i}(\boldsymbol{\delta})$ be the output of the i-th convolutional layer with weight parameters $\boldsymbol{W}_i$ under the single perturbation input $\boldsymbol{\delta}$. We define the probability of activated neurons $\boldsymbol{\Delta}$ as $p(\boldsymbol{\Delta}|f^{\boldsymbol{W}_i}(\boldsymbol{\delta}))$, where $p(\cdot)$ means the probability $p(\boldsymbol{\Delta}_{ij})$ to the j-th neuron at the i-th convolutional layer. Inspired by [8], the uncertainty probability of the output in the i-th layer is defined as follows:

$$\begin{aligned} p(\boldsymbol{\Delta}|f^{\boldsymbol{W}_i}(\boldsymbol{\delta})) &= \textstyle\sum_j p(\boldsymbol{\Delta}_{ij}|\boldsymbol{w}_j^i)p(\boldsymbol{w}_j^i|\boldsymbol{\delta}, \boldsymbol{W}_i) \\ &= \textstyle\sum_j p(\boldsymbol{\Delta}_{ij}|\boldsymbol{w}_j^i)q(\boldsymbol{w}_j^i), \end{aligned} \tag{2}$$

where $\boldsymbol{w}_j$ is the filter parameter corresponding to the j-th neuron at the i-th layer, $q(\boldsymbol{w}_j^i) = p(\boldsymbol{w}_j^i|\boldsymbol{\delta}, \boldsymbol{W}_i)$ means the selective probability of $\boldsymbol{w}_j$ under the perturbation $\boldsymbol{\delta}$ input, and $p(\boldsymbol{\Delta}_{ij}|\boldsymbol{\delta}, \boldsymbol{w}_j^i)$ means the probability of j-th activated neuron. Similar to [8], the posterior distribution of each filter $q(\boldsymbol{w}_j^i)$ is approximated as follows:

$$q(\boldsymbol{w}_j^i) \sim \hat{\boldsymbol{w}}_j^i = \boldsymbol{w}_j^i \cdot \boldsymbol{z}_j, \text{ s.t. } \boldsymbol{z}_j \sim \text{Bernoulli}(\alpha_j), \tag{3}$$

where $\boldsymbol{z}_j$ is a random variable that satisfies the Bernoulli distribution with a parameter α_j.

The l_2-norm is used to estimate the activation degree of each neuron, in which a larger degree means that the neurons are activated with a higher probability. When more neurons are activated, the uncertainty becomes more massive enough. Therefore, we can directly maximize the function in Eq.2 with an approximated posterior to define the uncertainty of the j-th neuron at the i-th layer as follows:

$$p(\boldsymbol{\Delta}_{ij}) = -\log\left(\|f_j^{\boldsymbol{W}_i}(\boldsymbol{\delta})\|_2 \cdot \boldsymbol{z}_j\right), \text{ s.t. } \|\boldsymbol{\delta}\|_p < \epsilon, \tag{4}$$

where $f_j^{\boldsymbol{W}_i}(\cdot)$ is the output value of the j-th neuron at the i-th layer. The binary variable $\boldsymbol{z}_j = 0$ denotes that the neuron $\boldsymbol{\Delta}_{ij}$ is dropped out as the input to the layer, which can be approximated by adding a dropout distribution on the neurons via a Monte Carlo (MC) dropout. Similar to the processing in [8], the MC dropout can be approximated by performing T stochastic forward passes through the network and average the results. Therefore, we estimate the virtual Epistemic uncertainty of each neuron $\boldsymbol{\Delta}_{ij}$ as:

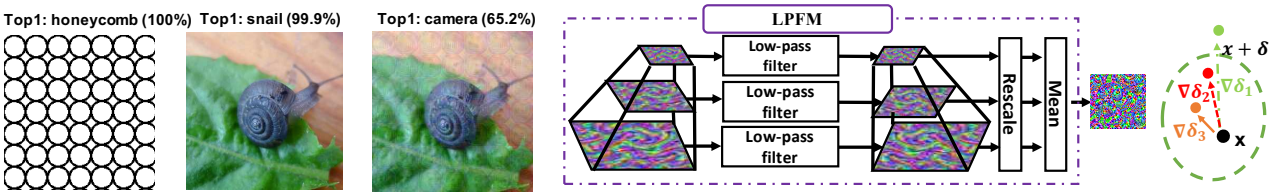
$$p_e(\boldsymbol{\Delta}_{ij}) = \frac{1}{T}\sum_t^T \left[-\log\left(\|f_j^{\boldsymbol{W}_i}(\boldsymbol{\delta})\|_2 \cdot \boldsymbol{z}_j^t\right)\right], \text{ s.t. } \|\boldsymbol{\delta}\|_p < \epsilon, \tag{5}$$

where $\boldsymbol{z}_j^t$ means the neuron $\boldsymbol{\Delta}_{ij}$ is dropped out through the t-th feedforward network.

Based on the uncertainty definition of each neuron, the loss function of the whole CNN model's uncertainty is easily achieved as:

$$U_e(\boldsymbol{\delta}) = \sum_i^K \sum_j p_e(\boldsymbol{\Delta}_{ij}), \text{ s.t. } \|\boldsymbol{\delta}\|_p < \epsilon, \tag{6}$$

where K is the number of the convolutional layer under a given CNN model.



(a) Texture Prior (b) Original Example (c) Adversarial Example (d) The architecture of the LPFM (e) The analysis.

Figure 2. **Subfigure** (a) is a specific texture image which makes the VGG-16 output the label "honeycomb" with 100% confidence. **Subfigure** (b) is the original image, and the VGG-16 can predict it to the right label with high confidence. **Subfigure** (c) is the adversarial example with adding texture-based perturbation in subfigure (a), which output the wrong label "reflex_camera" with 65.2% confidence. **Subfigure** (d) shows the architecture of the Laplacian Pyramid Frequency model (LPFM), and the subfigure (d) give the analysis of the proposed method. In subfigure (d), LPFM contains four parts: spatial pyramid structure, low-pass filter, rescaling process, and mean calculation. Given a perturbation input, LPFM first construct the n-level spatial pyramid model. Then, we use low-pass filter to reduce the high frequency information. Finally, rescaling and mean process are used to generate the final perturbation we needed. In **Subfigure** (e), the black point x is the original sample that can be correctly classified, the green point $x + \delta$ is the adversarial example with the best direction $\nabla\delta_1$ (green line). $\nabla\delta_2$ and $\nabla\delta_3$ are the approximated directions during iteration. When a similar point as red point is considered, the gradient will be pulled back into the semantic space.

3.2. Aleatoric uncertainty with Texture Bias

In the above section, we capture the Epistemic uncertainty over the model parameters to approximate the probability of $p(\boldsymbol{\Delta}|f^{\boldsymbol{W}_i}(\boldsymbol{\delta}))$. To capture the Aleatoric uncertainty, we need to tune the distribution of the perturbation $\boldsymbol{\delta}$ and utilize such prior as a regularizer during the perturbation learning. The Aleatoric uncertainty reflects the influence of the observation noise to the model output, such that different style of the perturbation will have different attacking performance. The key issue turns to how to initialize and utilize such a perturbation during UAP learning.

All existing UAP methods initialize the adversarial noise $\boldsymbol{\delta}$ with a simple random Gaussian or uniform distribution. Recent observations in [10] show that CNNs pre-trained on ImageNet are strongly biased towards recognizing texture information. Therefore, we argue that a *texture style bias* can help to maximize the activation of the texton at each layer, which can further increase the model uncertainty with simple input noise observation. The bias can be a better initialized to improve the existing UAP algorithms for attacking, which, however, is left unexploited in the literature.

To validate the above assumption, we propose to use a texture image as a special UAP to attack VGG-16 [28] pre-trained on ImageNet [3]. The fooling rate achieves 49% on ImageNet validation set by adding a *texture bias*, as shown in Figure 2 (a), which performs equally well to the cutting-edge GD-UAP [23]. We replace the initialization of UAP with the texture image and use a similar learning approach as in GD-UAP. The fooling rate significantly improves 18% over the GD-UAP in a data-independent manner. A conclusion can be made that a simple texture perturbation prior can help to improve existing data-independent methods.

On the other hand, after initialization, another key problem is how to utilize this texture style information during perturbation learning. Inspired by the work in texture synthesis [9], we use the style loss that specifically encourages the reproduction of texture details as follows:

$$L_a = \mathbb{E}[\boldsymbol{G}_{ij}(\boldsymbol{\delta}) - \boldsymbol{G}_{ij}(\boldsymbol{\delta}_0)], \boldsymbol{G}_{ij}(\boldsymbol{\delta}) = \sum\nolimits_k \boldsymbol{F}^l_{ik}(\boldsymbol{\delta})\boldsymbol{F}^l_{jk}(\boldsymbol{\delta}), \tag{7}$$

where $\boldsymbol{G}$ is the Gram matrix of the features extracted from certain layers of the pre-trained classification network, $\boldsymbol{F}^l_{ik}$ is the activation of the i-th filter at position k in the layer l, and $\boldsymbol{\delta}_0$ is the texture style image that is fixed during training, as shown in Figure 2 (a). We use one layers of the VGG-16 ($relu3\_2$) to define our style loss.

3.3. Optimization

We proposed to capture both Epistemic uncertainty and Aleatoric uncertainty in our perturbation learning. To this end, we change the style loss with a texture bias in Eq.7 resulting in the virtual Epistemic uncertainty as a regularizer, and we have:

$$L(\boldsymbol{\delta}) = U_e(\boldsymbol{\delta}) + \rho \times L_a(\boldsymbol{\delta}), \tag{8}$$

where ρ are the trade-off parameter to control the weight of Aleatoric uncertainty. The gradient of Eq.8 can be easily computed because all functions are convex and smooth. The gradient descent algorithm is used to update the perturbation at the i-th iteration as:

$$\boldsymbol{\delta}_i = \boldsymbol{\delta}_{i-1} - \lambda * \nabla L(\boldsymbol{\delta}), \tag{9}$$

where λ is a learning rate. $\nabla L(\boldsymbol{\delta})$ is the gradient of Eq.8 that can be easily achieved by the Adam optimizer.

With the Aleatoric uncertainty, the UAP is designed to learn the low-frequency information, which is similar to the texture-like patterns [33]. Compared to the low-frequency part, the gradient magnitude of the high-frequency part tends to be relatively large. As a result, we consider using a Laplacian pyramid frequency model (LPFM) to increase the low-frequency part of the UAP. LPFM first constructs

Algorithm 1 Prior Driven Uncertainty Approximation

Input: Parameters learning rate λ, dropout probability p.
Output: Universal perturbation vector δ.

1: Initialize $\boldsymbol{\delta}$ with texture image.
2: **repeat**
3: Compute $f^{\boldsymbol{W}_i}(\boldsymbol{\delta})$ at i-th convolutional layer
4: Approximate $\boldsymbol{z}_j = 0$ via MC dropout;
5: Compute the loss function in Eq.8;
6: Update the perturbation vector via Eq.10 - Eq.12.
7: **until** the max iteration or convergence.
8: Output the universal perturbation $\boldsymbol{\delta}$.

an n-level Laplacian spatial pyramid of input gradient and then outputs the gradient in each level with a given low-pass filter. The framework is shown in Figure 2 (d). At last, we sum gradients in all scales to obtain the final gradient with a whitening process. The final updating scheme with a boosted momentum is rewritten as follows:

$$\boldsymbol{g}_i = \mu \cdot \boldsymbol{g}_{i-1} + \mathrm{N}(\nabla L(\boldsymbol{\delta}_i)), \tag{10}$$

$$\boldsymbol{\delta}_i = \boldsymbol{\delta}_{i-1} + \lambda \cdot \boldsymbol{g}_i, \tag{11}$$

$$\boldsymbol{\delta}_i = \min\big(\max(-\epsilon, \boldsymbol{\delta}_i), \epsilon\big), \tag{12}$$

where $\mathrm{N}(\cdot)$ is the calculation of LPFM, g_i is the momentum at the i-th iteration, and λ is the learning rate. In Eq.12, we first maximize the pixel value between each gradient value and the constraint $-\epsilon$ and then compare the output value to ϵ that further constrains the output to be less than ϵ. We summarize the overall procedure of the proposed PD-UA method in Algorithm 1.

3.4. Analysis

Recent prior arts [32, 34] have shown that, when activating all neurons at each layer, neurons with a similar concept are repeatedly activated, which results in the information redundancy. Generally, most adversarial learning algorithms aim to search for the direction that pushes the current input to be out of the existing class space, which are usually implemented in the iterative manner. As the red and orange lines are shown in Figure 2 (e), if the current gradient is computationally dependent on redundant neurons, the value of the gradient will be reduced, even in the opposite direction. Such information bias makes the space search more complicated and time-consuming. Moreover, due to the lack of perturbation prior, the explicit semantic directions are harder to be computed.

With the proposed texture prior, the direction of the universal perturbation is more reasonable, which leads to a series of repeated patterns, as shown in Figure 3 (a). To handle this issue, the proposed method in Eq.6 approximates the CNN mode uncertainty that can adaptively drop out neurons at each layer, which efficiently solves the information bias problem. From our extensive experiments in Section 4, the proposed method can significantly improve the attacking performance with a faster convergence rate.

4. Experiments

Datasets. We evaluate the proposed PD-UA method to fool a serial of CNNs pre-trained on ImageNet [3], including GoogleNet [30], VGG-F [28], VGG-16 [28], VGG-19 [28], ResNet50 [14], and ResNet150 [14]. We use the ImageNet validation set [27] to evaluate the attacking performance.

Evaluation metrics. To quantitatively measure the attacking performance of the crafted UAP, we mainly consider the widely-used "fooling rate" metric [21]. Fooling rate (FR) presents the ratio of images whose predict label become incorrect by adding the UAP.

Comparative Methods. The proposed **PD-UA** method is compared to the state-of-the-art data-independent UAP: **GD-UAP** [23]. We also add the Aleatoric uncertainty loss in Eq.7 on GD-UAP, leading to **GD-UAP+P**. For a fair comparison, our propose method without Aleatoric uncertainty, named **UA**[1], which considers the virtual Epistemic uncertainty when $\rho = 0$ in Eq.8. We also report the attacking performance by directly using the texture bias perturbation, *i.e.,* texture image in Figure 2 (a), which is named **PP**. Similar to GD-UAP [23], we also compare our method when using a pseudo data prior[2], which approximates the real image by random Gaussian samples.

Implementation Details. For all comparative methods, we follow the same parameter setting in [23] and reproduce the experiments for all baselines. Our PD-UA method is implemented based on Tensorflow. For LPFM, we construct a 3-level spatial pyramid, and the binomial filter is set to $[1, 4, 6, 4, 1]$ that is the low-pass filter, both of which can lead to the best-attacking performance. The maximum perturbation ϵ is set to 10 with the pixel value in $[0, 255]$. The learning rate λ and the decay factor μ are set to 0.05 and 0.8, respectively. And the probability of the MC dropout is set to 0.1, and $\rho = 1e^{-3}$.

4.1. Quantitative Results on Attacking

We compare our PD-UA with recent data-independent UAPs on ImageNet Validation set [3]. The fooling rates of different methods are reported in Table 1, Table 2, and Table 3. We observe that PD-UA consistently achieves superior performances, no matter whether a prior (both pseudo data prior and texture bias) is used or not.

Table 1 shows the attacking performances for different UAPs without training data information by directly learning UAP from CNN models. We first report the fooling

[1]Note that, UA method uses the random Gaussian distribution prior for UAP initialization, which is the non-textural prior.

[2]Pseudo data prior is to simulate the numerical distribution of images, which are sampled from a dynamic Gaussian distribution.

Method	VGG-F	GoogleNet	VGG-16	VGG-19	ResNet-50	ResNet-150
GD-UAP	85.96	51.61	45.47	40.68	35.59	28.87
PP	46.40	40.66	43.69	43.56	22.96	24.12
UA	87.75	61.41	48.46	41.66	38.87	32.87
GD-UAP+P	86.53	58.37	51.63	46.83	63.71	50.35
PD-UA	**90.10**	**67.12**	**53.09**	**48.95**	**65.84**	**53.51**

Table 1. The evaluation results (FR%) of the proposed method and other data-independent UAPs. All perturbations are trained from the corresponding CNN models with a perturbation prior.

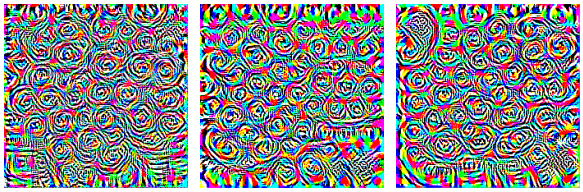
Method	VGG-16	VGG-19	ResNet-50	Resent-150
UAP	77.82	80.80	81.39	84.10
GD-UAP	63.08	52.67	53.17	39.56
GD-UAP+P	64.95	52.49	56.70	44.23
UA	65.71	61.52	59.70	39.77
PD-UA	**70.69**	**64.98**	**63.50**	**46.39**

Table 2. Fooling rate results for GD-UAP and PD-UA perturbation learned with pseudo data prior.

rates of PP directly using a texture image as UAP, which considers the Aleatoric uncertainty. We can observe that PP has achieved comparable attacking performance to GD-UAP, especially for deep CNN model, like ResNet-150. Moreover, we evaluate the attacking performance of GD-UAP with an adding Aleatoric uncertainty, and the improvements of the performance are significant on four deeper CNN models (VGG-16, VGG-19, ResNet-50, and ResNet-150). Then, we further compare UA to GD-UAP, where UA focuses on the Epistemic uncertainty. The results are shown in Table 1. UA has an averaged 7.68% improvement in terms of fooling rate compared to GD-UAP, which validates the effectiveness of the Epistemic uncertainty. Then, by combining these two uncertainties, the attacking performances are the best, which still achieves an averaged improvements of 4.13%, 14.99%, 2.83%, 4.53%, 3.34%, 3.80% on six CNN models (VGG-F, GoolgeNet, VGG-16, VGG-19, Resent-50 and Resent-150) compared to the GD-UAP+P, respectively.

Following the approach in [23], we add a pseudo data prior to craft the universal perturbation and report the results in Table 2. The proposed method PD-UA still achieve the best performances with such data prior. Note that, PD-UA for ResNet achieves a similar fooling rate as the data-dependent UAP, which shows that the pseudo data prior and texture bias (Aleatoric uncertainty) are both useful for crafting UAP in a data-independent way.

In addition, we report the black-box attacking performance between different CNNs in Table 3, where the UAP is trained on one CNN model, and then evaluated on others. PD-UA achieves a better black-box attacking performance than the previous GD-UAP [23]. Visually, by using the same perturbation bias, the final universal perturbations from different methods look similar to each other, as shown in Figure 3 and Figure 4. As a conclusion, PD-UA do help to craft robust universal perturbation, which improves both the white-box and black-box attacking performance.



(a) GD-UAP+P. (b) UA. (c) PD-UA.

Figure 3. The visualizations of VGG-F. (Best viewed in color.)

4.2. On the Aleatoric Uncertainty

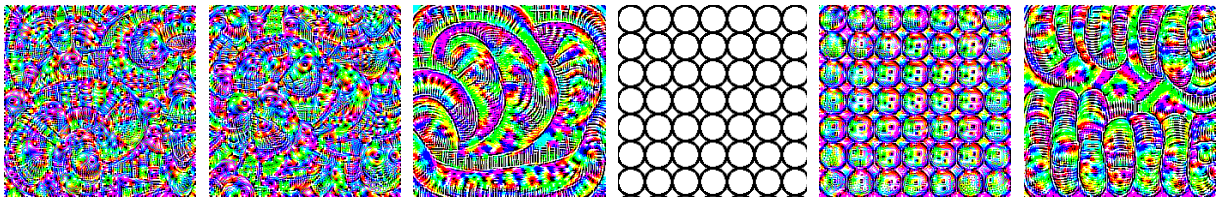
We analyze the influence of using a texture bias, which affects the Aleatoric uncertainty of the output. We observe that the improvement is significant for deeper CNNs (most of which achieve at least 15% improvement). The work in [24] reveals that the initialization with a pre-trained perturbation[3] can improve the attacking performance, which performs comparably well to our method, especially for a deeper CNN. It further demonstrates that a better texture bias is useful. Due to the irregularity of the texture information, the prior from a pre-trained CNN is not a good choice to synthesize the perturbation prior, as shown in Table 4. In addition to the pre-trained prior, we further compare the other two priors, *i.e.,* gradient prior and texture bias. The results show that texture bias is still better than these two different priors.

We evaluate the attacking performance by directly using the prior perturbation without training, which is chosen as our baseline (termed PP). We use a perturbation example in Figure 2 (a), where the image is randomly synthesized via Bunch sampling algorithm. More interestingly, directly using such a prior perturbation can achieve a similar fooling rate on deeper CNNs, *i.e.,* VGG and ResNet, when compared to the GD-UAP. To explain, there is strong texture information in the shallow and middle layers of deeper networks, which is not the case for smaller CNN models, *i.e.,* VGG-F and GoogleNet. The attacking performance of UA is, therefore, better than GD-UAP+P for the shallow CNN models, *i.e.,* VGG-F and GoogleNet. We analyze the phenomena via visualization of the corresponding perturbations, as shown in Figure 3 and Figure 4. We observe that the perturbation crafting from UA has similar textural pat-

[3]This pre-trained perturbation is computed with VGG-F's perturbation.

		VGG-16	VGG-19	ResNet-50	ResNet-150	VGG-F	GoogleNet
VGG-16	PD-UA	**53.09**	**49.30**	**33.61**	**30.31**	**48.98**	**39.05**
	GD-UAP+P	51.63	44.07	32.23	28.78	44.38	36.79
	UA	48.46	41.97	29.09	24.90	47.63	35.52
	GD-UAP	45.47	38.20	27.70	23.80	44.30	34.13

Table 3. Fooling rates for GD-UAP and PD-UA perturbation for the data-independent case, which are evaluated based on the ImageNet validation set. Each row presents the fooling rates for perturbation learned on one CNN to attack the other CNNs. The white-box attacking is performed when the source model and target model are the same, while the black-box attacking is done vice versa. Due to the paper limit, we only report the attacking results of VGG-16.



(a) GD-UAP. (b) UA. (c) UA with PDR. (d) Prior. (e) PD-UA. (f) PD-UA with PDR.

Figure 4. The visualization of UAPs, which are crafted for VGG16. "PDR" means the pseudo data prior. (Best viewed in color.)

	GoogleNet	VGG-16	VGG-19	ResNet-50
Pre-trained	60.58	50.34	48.64	59.33
Gradient	63.58	51.83	48.32	63.33
Texture	65.23	51.18	48.88	64.59

Table 4. Different initializations of texture bias. "Pre-trained" means the perturbation prior is synthesized from a small CNN, such as VGG-F. "Gradient" is the perturbation generated from gradient calculation on the 'inception_3' layer in GoogleNet. "Texture" is a texture prior as shown in Figure 2 (a).

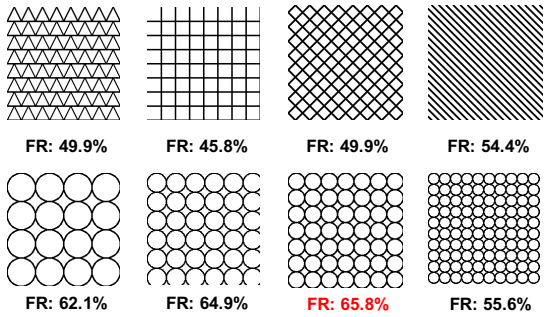


Figure 5. The results for different texture synthesis methods via different geometric structures. We mainly report the results based on circle, line, square, triangle, and diamond. All results are evaluated on ResNet-50 model in a data free fashion. The fooling rate results are reported under the corresponding images.

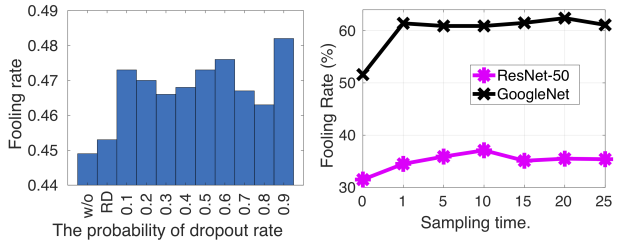
terns as GD-UAP+P and PD-UA, such as a circle textural pattern. Moreover, some general information such as "dog eye" is contained in the perturbation when the perturbation prior is used, as shown in Figure 4 (e) and (f), which help to activate the image-agnostic neurons at high-level layer. That is to say; there exists a specific semantic basis from the gradient feedback to the perturbation for smaller CNNs; on the contrary, there exists a more textual basis in deeper CNN models.

Moreover, we also compare the impact of 8 different perturbation priors, such as circle, lines, square, triangle, and diamond, whose results are shown in Figure 5. Quantitatively, the circle texture achieves overall the best performance among than them, in particular with seven circles in each row. We synthesize the texture images with a small image patch that are randomly cropped from the Figure 3 (b). Base on the new perturbation prior, the performance is 65.67%, which is similar to that of the seven circles in each row. The reason why choosing this basic geometrical patterns lies in that we want to explore the impact of simple and repeated texture cues on attack performance. Moreover, these results can help us select the corresponding texture patches effectively, which can achieve comparable attacking performance.

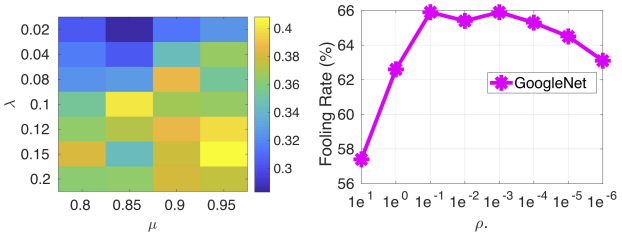
4.3. On the Epistemic Uncertainty

This subsection focuses on the influence of the proposed virtual Epistemic uncertainty, whose analysis is done by varying one value while fixing the others. Figure 6 (a) further shows the influence of the sampling probability in Eq.5 for MC dropout conducted on VGG-16 with random initialization. We also report the results of regular dropout (RD) and the original optimization without dropout (w/o). However, the regular dropout and no dropout perform just comparably to the MC dropout, while the gap of which affects the final performance to a certain extent. However, a higher probability will make the sampling space larger, which needs a larger calculation, but with is more time-consuming. We further evaluate the attacking performance against the T MC sampling in Eq.5. The results shown in Figure 6 (b) are based on the ResNet50 model. We observe that the performance is increased with the number of sampling, but then it changes little when $T \geq 10$. Therefore, we set $p = 0.1$ and $T = 10$ in all our experiments, which can obtain a balanced performance and well approximate the Epistemic uncertainty.

We describe the parameter analysis between λ and μ,

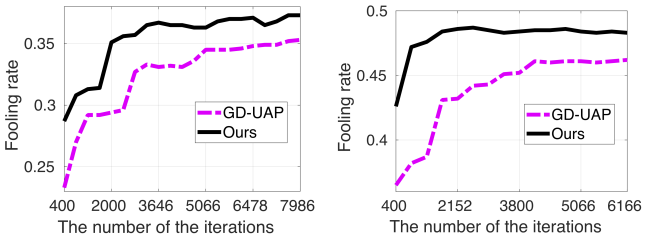


(a) Sampling Rate α. (b) Sampling Times T.



(c) λ and μ. (d) ρ.

Figure 6. Parameters analysis of the Epistemic Uncertainty. (Best view in color.)



(a) Analysis of Loss function. (b) Analysis of Optimization.

Figure 7. Parameters analysis. (Best view in color.)

	Lp	Bp	Hp	1-level	2-level	l_2	l_1
V	**53.09**	52.78	49.66	50.15	51.42	26.31	21.11
R	**65.84**	63.77	49.53	62.28	64.17	27.80	28.11

Table 5. The illustration of LPFM. The perturbations are trained on VGG-16 (V) and ResNet-50 (R). We evaluate three different filters, *i.e.*, low-pass filter (Lp), band-pass filter (Bp), and high-pass filter (Hp). We show the results with different levels in the pyramid model. And we also replace the LPFM with on two widely-used normalizations, such as l_2-norm (l_2) and l_1-norm (l_1).

which are important for optimization. We further evaluate on ResNet50 without the perturbation prior, and the results in Figure 6 (c) shows that a better performance is achieved with a larger momentum and learning rate. Therefore, we set the learning rate to 0.05, and the momentum coefficient to 0.8. In addition, to evaluate the effectiveness of two uncertainties in the loss function Eq.8, we evaluate the fooling rates by tuning the parameters ρ on GoogleNet. As the results shown in Figure 6 (d), we have observed that the best accuracy is achieved when empirically setting $\rho = 1e^{-3}$, which is consistent in all CNN models.

4.4. Ablation Study

In this subsection, we first compare the convergence of our model with the baseline (GD-UAP). For a fair comparison, both perturbations are trained from ResNet-50 with random initialization and the same optimization algorithm. As shown in Figure 7 (a), these two schemes can converge after $6,000$ iterations. But, the proposed method converges 2 times faster with better results on the validation set chosen from another dataset, such as PASCAL VOC. Except the loss function in Eq.8, we also use a similar scheme to evaluate the proposed optimization algorithm, as shown in Figure 7 (b). The proposed boosted momentum with LPFM optimization achieves better performance when compared with the classical Adam optimizer. In sum, the proposed PD-UA not only achieves better-attacking performance but also converges quickly after $2,000$ iterations.

In addition, we also evaluate the impact on different settings of LPFM in optimization. As shown in Table 5, we first use different filters to evaluate whether low-frequency signals need to be preserved. In Table 5, low-pass filter has obtained the best performance, while the high-pass filter and band-pass filters are worse[4]. We believe that it is not a good choice with a full focus on the high-frequency signal, while the low-frequency signal is relatively important for generating UAP. As shown in Figure 3 (b) and (c), LPFM with a low-pass filter makes the perturbation contains low-frequency textural information. We analyze the effect of the number of pyramid layer, which shows that such a setting has little effect on the final results, while the 2-3 level in used can achieve satisfactory results.

5. Conclusion

In this paper, we propose a novel universal perturbation method, which mainly considers the model uncertainty to craft a robust universal perturbation. First, we maximize the activated convolutional neurons via MC dropout technology to approximate the model uncertainty, which can learn perturbation more effectively and efficiently. Then, to approximate Aleatoric uncertainty, a texture-based image is utilized as the perturbation, which can significantly improve the attacking performance. Based on a new iterative updating scheme, we synthesize a more robust universal perturbation. Extensive experiments verify that the proposed method can better attack the cutting-edge CNN models. In future work, we will investigate the other new prior selection to improve our performance further.

Acknowledgements. This work is supported by the National Key R&D Program (No.2017YFC0113000 and No.2016YFB1001503), Nature Science Foundation of China (No.U1705262, No.61772443, and No.61572410), Scientific Research Project of National Language Committee of China (No.YB135-49), and Nature Science Foundation of Fujian Province, China (No.2017J01125 and No.2018J01106).

[4]We replace the binomial filter with the band-pass filter ($[0, 1, 0, 1, 0]$) and the high-pass filter ($[4, 1, 0, 1, 4]$).

References

[1] Naveed Akhtar and Ajmal S Mian. Threat of Adversarial Attacks on Deep Learning in Computer Vision - A Survey. *Journal of the IEEE Access*, 2018. 1

[2] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of the S&P*, 2017. 1

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-scale Hierarchical Image Database. In *Proccedings of the CVPR*, 2009. 2, 4, 5

[4] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? Does it matter? *Journal of the Structural Safety*, 2009. 2

[5] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. Boosting Adversarial Attacks with Momentum. In *Proceedings of the CVPR*, 2018. 2

[6] Deng-Ping Fan, Wenguan Wang, Ming-Ming Cheng, and Jianbing Shen. Shifting more attention to video salient object detection. In *Proceedings of the CVPR*, 2019. 1

[7] Yarin Gal. *Uncertainty in deep learning*. PhD thesis, University of Cambridge, 2016. 1, 3

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation - Representing Model Uncertainty in Deep Learning. In *Proceedings of the ICML*, 2016. 2, 3

[9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture Synthesis Using Convolutional Neural Networks. In *Proceedings of the NeurIPS*, 2015. 4

[10] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained CNNs are Biased Towards Texture: Increasing Shape Bias Improves Accuracy and Robustness. In *Proceedings of the ICLR*, 2019. 2, 4

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press Cambridge, 2016. 1

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proceedings of the ICLR*, 2015. 1, 2

[13] Tamir Hazan, George Papandreou, and Daniel Tarlow, editors. *Perturbations, Optimization, and Statistics*. MIT Press, 2016. 1, 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the CVPR*, 2016. 1, 5

[15] Valentin Khrulkov and Ivan V Oseledets. Art of Singular Vectors and Universal Adversarial Perturbations. In *Proceedings of the CVPR*, 2018. 1, 2

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Examples in The Physical World. In *Proceedings of the ICLR*, 2017. 2

[17] Jie Li, Rongrong Ji, Hong Liu, Xiaopeng Hong, Yue Gao, and Qi Tian. Universal perturbation attack against image retrieval. In *Proceedings of the ICCV*, 2019. 2

[18] Yingzhen Li and Yarin Gal. Dropout Inference in Bayesian Neural Networks with Alpha-divergences. In *Proceedings of the ICML*, 2017. 1

[19] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-box Attacks. In *Proceedings of ICLR*, 2017. 1, 3

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the CVPR*, 2015. 1

[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In *Proceedings of the CVPR*, 2017. 1, 2, 3, 5

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool - A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of the CVPR*, 2016. 2

[23] Konda Reddy Mopuri, Aditya Ganeshan, and Venkatesh Babu Radhakrishnan. Generalizable Data-free Objective for Crafting Universal Adversarial Perturbations. *Journal of the IEEE TPAMI*, 2018. 1, 2, 4, 5, 6

[24] Konda Reddy Mopuri, Utsav Garg, and Venkatesh Babu Radhakrishnan. Fast feature fool - a data independent approach to universal adversarial perturbations. In *Proceedings of the BMVC*, 2017. 1, 2, 6

[25] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. NAG - Network for Adversary Generation. In *Proceedings of the CVPR*, 2018. 3

[26] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the ASIA CCS*, 2017. 1, 3

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet Large Scale Visual Recognition Challenge. *Journal of the IJCV*, 2015. 5

[28] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In *Proceedings of the ICLR*, 2015. 4, 5

[29] L. Smith and Y. Gal. Understanding measures of uncertainty for adversarial example detection. In *Proceedings of the UAI*, 2018. 1

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the CVPR*, 2015. 5

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *Proceedings of the ICLR*, 2014. 1, 2

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *Proceedings of the ICLR*, 2014. 5

[33] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010. 2, 4

[34] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba 0001. Interpretable Basis Decomposition for Visual Explanation. In *Proceedings of the ECCV*, 2018. 5