Disrupting Deep Uncertainty Estimation Without Harming Accuracy

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks (DNNs) have proven to be powerful predictors and are widely used for various tasks. Credible uncertainty estimation of their predictions, however, is crucial for their deployment in many risk-sensitive applications. In this paper we present a novel and simple attack, which unlike adversarial attacks, does not cause incorrect predictions but instead cripples the network's capacity for uncertainty estimation. The result is that after the attack, the DNN is more confident of its incorrect predictions than about its correct ones **without** having its accuracy reduced. We present two versions of the attack. The first scenario focuses on a black-box regime (where the attacker has no knowledge of the target network) and the second scenario attacks a white-box setting. The proposed attack is only required to be of minuscule magnitude for its perturbations to cause severe uncertainty estimation damage, with larger magnitudes resulting in completely unusable uncertainty estimations. We demonstrate successful attacks on three of the most popular uncertainty estimation methods: the vanilla softmax score, Deep Ensembles and MC-Dropout. Additionally, we show an attack on SelectiveNet, the selective classification architecture. We test the proposed attack on several contemporary architectures such as MobileNetV2 and EfficientNetB0, all trained to classify ImageNet.

1 Introduction

Deep neural networks (DNNs) show great and improving performance in a wide variety of application domains including computer vision and natural language processing. Successful deployment of these models, however, is critically dependent on providing effective *uncertainty estimation* for their predictions, or employing some kind of *selective prediction* [6]. For example, when self-driving cars cannot make sufficiently good decisions, they must apply fail-safe procedures such as slowing down the vehicle, disengaging the auto pilot and passing control to a human driver, or requesting remote assistance. The trigger initiating these measures depends inherently on the system's level of confidence in its predictions. It is easy to envision many other critical application domains, such as medical diagnosis, where uncertainty estimation is required for effective deployment.

Theoretically, ideal uncertainty estimation can be obtained within a Bayesian framework. Unfortunately, viable Bayesian DNNs are not yet available. In the context of classification, practical and well-known uncertainty estimation techniques include: (1) the classification prediction's *softmax score* [2, 3], which quantifies the embedding margin between an instance to the decision boundary; (2) *MC-Dropout* [5], which is argued to proxy *Bayesian networks* [19, 17, 20]; (3) *Deep Ensembles* [15], which have shown state-of-the-art results in various estimation settings; and (4) *SelectiveNet* [7], which learns to estimate uncertainty in a way that produces optimal risk for a specific desired coverage.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

In this paper we show that all these well-known uncertainty estimation techniques are vulnerable to a new and different type of attack that can completely destroy their usefulness and that works in both *black-box* (where the attacker can only query the attacked model for predicted labels and has no knowledge of the model itself) and *white-box* (the attacker has complete knowledge about the attacked model) settings. While standard adversarial attacks target model accuracy, the proposed attack is designed to *keep accuracy performance intact* and refrains from changing the original classification predictions made by the attacked model. To demonstrate the relevance of our findings, we test our attack on modern architectures, such as MobileNetV2 [22], EfficientNet-B0 [25], as well as on standard baselines such as ResNet50 [11], DenseNet161 [12] and VGG16 [23]. Our attacks are focused on the task of ImageNet classification [4]. Figure 1 conceptually illustrates how our attack

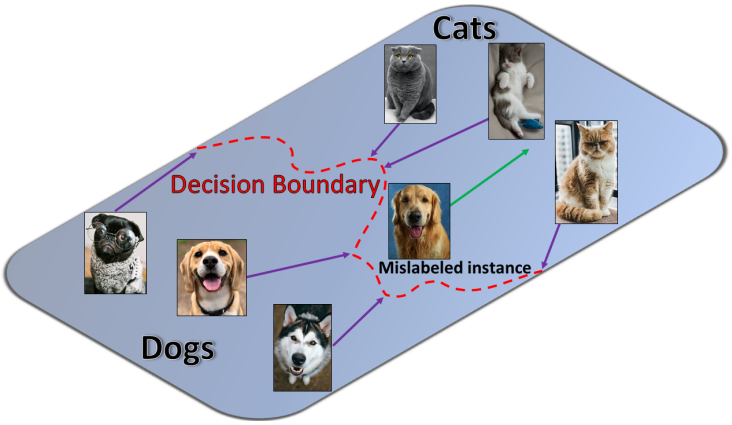

Figure 1: The intuition driving the attack on a classifier of cats and dogs, when the uncertainty measure used is softmax. Correct predictions are moved closer to the decision boundary without crossing it, thus increasing their uncertainty (purple arrows), and incorrect predictions are moved further away from the decision boundary to decrease their uncertainty (green arrow).

works. Consider a classifier for cats vs. dogs that uses its prediction's softmax score as its uncertainty estimation measurement. An end user asks the model to classify several images, and output only the ones in which it has the most confidence. Since softmax quantifies the margin from an instance to the decision boundary, we visualize it on a 2D plane where the instances' distance to the decision boundary reflect their softmax score. In the example shown in Figure 1, the classifier was mistaken about one image of a dog, classifying it as a cat, but fortunately its confidence in this prediction is the lowest among its predictions. A malicious attacker targeting the images in which the model has the most confidence would want to increase the confidence in the mislabeled instance by pushing it away from the decision boundary, and decrease the confidence in the correctly labeled instances by pushing them closer to the decision boundary.

This attack, however, can be accomplished without harming the model's accuracy, and thus avoid raising suspicion about a possible attack. Even if the mislabeled dog is assigned more confidence than only a single correctly predicted image, this attack will undermine the model's uncertainty estimation performance, since based on its estimation measurements, it cannot offer its newly least-confident (and correct) instance without offering the now more-confident mislabeled dog. It will either have to exclude both from the predictions it returns, reducing its coverage and its usefulness to the user, or include both and return predictions with mistakes, increasing its risk.

Our attack, unlike adversarial attacks, creates damage even with small magnitudes of perturbations, befitting even an attacker with very limited resources (see Section 5). This benefit is inherent in our goal, which is different from standard adversarial attacks: adversarial attacks seek to move the attacked instance across the model's decision boundary, and thus require the magnitude of the perturbations it causes to be sufficiently large to allow this. Our objective is to simply move the correctly and incorrectly predicted instances, relative to one another within the boundaries, which requires a less powerful "nudge". One could think that restricting our attack from changing the predictions would substantially limit its damage. The results, however, show that this attack deals a devastating blow to uncertainty estimation. Figure 2 exemplifies such an attack on EfficientNet, showing how the model would be more confident of binoculars being a tank than a correctly-labeled

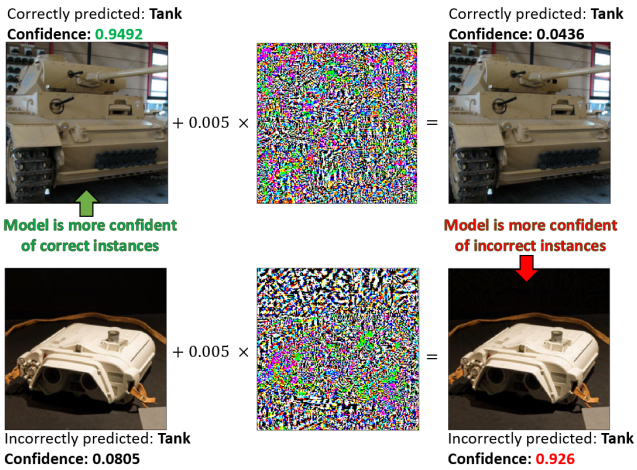

Figure 2: Demonstration of attacking EfficientNet with softmax as the uncertainty estimation method. Left: Two images being predicted to be tanks. The top one is correctly labeled and the model has high confidence in its label. The bottom image is incorrectly labeled as binoculars, with the model having low confidence in its prediction. Right: After adding perturbations to the original images, the model has lower confidence in its (correct) tank label than in the wrongly labeled binoculars.

tank. Figure 4 shows the *Risk-Coverage* (RC) curve (see Appendix A for an explanation of RC curves) for EfficientNet under our attack, and that for the strongest attack tested (in terms of perturbation magnitude, which is still a very small amount), approximately 20% of the model's most confident predictions were wrong. To the best of our knowledge, this paper is the first to offer an algorithm that attacks only the uncertainty estimation of any model using any uncertainty estimation technique in both black-box and white-box settings.

2 Related work

The most common and well-known approach to estimate uncertainty in DNNs for classification is using softmax at the last layer. Its values correspond to the distance from the decision boundary, with high values indicating high confidence in the prediction. The first ones to suggest this approach using DNNs were [2, 3].

Lakshminarayanan et al. [15] showed that an ensemble consisting of several trained DNN models gives reliable uncertainty estimates. For classification tasks, the estimates are obtained by calculating a mean softmax score over the ensemble. This approach was shown to outperform the softmax response on CIFAR-10, CIFAR-100, SVHN and ImageNet [9].

Motivated by Bayesian reasoning, Gal and Ghahramani [5] proposed *Monte-Carlo dropout* (MC-Dropout) as a way of estimating uncertainty in DNNs. At test time, for regression tasks this technique estimates uncertainty using variance statistics on several dropout-enabled forward passes. In classification, the mean softmax score of these passes is calculated, and then a predictive entropy score is calculated to be used as the final uncertainty estimate.

Geifman and El-Yaniv [7] introduced SelectiveNet, an architecture with a suitable loss function and optimization procedure that trains to minimize the risk for a specific required coverage. The SelectiveNet architecture can utilize any DNN as its backbone, and constructs three prediction heads on top of it – one of which is used solely for training, and the other two for prediction and rejection. SelectiveNet incorporates a soft "reject option" into its training process, allowing it to invest all its resources to achieve an optimal risk for its specific coverage objective. It was shown to achieve state-of-the-art results on several datasets for deep *selective classification* across many different coverage rates.

The *area under the risk-coverage curve* (AURC) metric was defined in [8] for quantifying the quality of uncertainty scores. This metric is motivated from a selective classification perspective: given a model, and using any uncertainty score for that model, an RC curve can be generated to indicate what

the resulting risk (y-axis) would be for any percentage of coverage (x-axis). AURC is then defined to be the area under this curve. More details and an example are provided in Appendix A.

Turning to *adversarial attacks*, one of the first *adversarial examples* was presented by Szegedy et al. [24]. They showed how small, barely perceptible perturbations of the input to a CNN, crafted by using the box-constrained L-BFGS (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno) algorithm, could alter the DNN's classification. Goodfellow et al. [10] developed a more efficient way to craft adversarial examples, called the *fast gradient sign method* (FGSM), requiring only a single backward pass of the network. Formally, with ϵ being a fixed magnitude for the image perturbation, y being the true label and a given $loss$ function, the FGSM adversarial example is simply $\tilde{x} = x + \epsilon \cdot sign(\nabla_x loss(f(x), y))$.

Liu et al. [16] demonstrated that crafting transferable adversarial examples (which are adversarial examples fooling multiple models, possibly of different architectures) for black-box attacks is especially challenging when targeting specific labels. The authors showed that using an ensemble results in highly transferable adversarial examples, presenting a successful black-box attack on Clarifai.com – a commercial company providing image classification services. The knowledge required for this method to create non-targeted attacks is only the ground-truth label and the attacked model's predicted label for targeted attacks.

3 Problem setup

Let $\mathcal{X}$ be the image space and $\mathcal{Y}$ be the label space. Let $P(\mathcal{X}, \mathcal{Y})$ be an unknown distribution over $\mathcal{X} \times \mathcal{Y}$. A model f is a prediction function $f : \mathcal{X} \to \mathcal{Y}$, and its predicted label for an image x is denoted by $\hat{y}_f(x)$. The model's *true* risk w.r.t. P is $R(f|P) = E_{P(\mathcal{X},\mathcal{Y})}[\ell(f(x), y)]$, where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a given loss function, for example, 0/1 loss for classification. Given a labeled training set $S_m = \{(x_i, y_i)\}_{i=1}^m \subseteq (\mathcal{X} \times \mathcal{Y})$, sampled i.i.d. from $P(\mathcal{X}, \mathcal{Y})$, the *empirical risk* of the model f is $\hat{r}(f|S_m) \triangleq \frac{1}{m}\sum_{i=1}^m \ell(f(x_i), y_i)$.

Note that in our context, the term *confidence* can be viewed as negative *uncertainty* (and vice versa), and we use these interchangeably. For a given model f, following [8], we define a *confidence score* function $\kappa(x, \hat{y}|f)$, where $x \in \mathcal{X}$, and $\hat{y} \in \mathcal{Y}$ is the model's prediction for x. The function κ should quantify confidence in the prediction of $\hat{y}$ for the input x, based on signals from the model f. This function should induce a partial order over points in $\mathcal{X}$, and is not required to distinguish between points with the same score. For example, for any classification model f with softmax at its last layer, the vanilla confidence score is its softmax response values: $\kappa(x, \hat{y}|f) \triangleq f(x)$. Note that we are not concerned with the values of the confidence function; moreover, they do not have to possess any probabilistic interpretation. They can be any value in $\mathbb{R}$. An optimal κ for a given classification model f and 0/1 loss should hold for every two instances with their ground truth, $(x_1, y_1) \sim P(\mathcal{X}, \mathcal{Y})$ and $(x_2, y_2) \sim P(\mathcal{X}, \mathcal{Y})$:

$$\kappa(x_1, \hat{y}_f(x_1)|f) \leq \kappa(x_2, \hat{y}_f(x_2)|f) \iff \mathbf{Pr}_P[\hat{y}_f(x_2) \neq y_2] \leq \mathbf{Pr}_P[\hat{y}_f(x_1) \neq y_1]$$

so that κ truly is monotone with respect to the loss. We strongly recommend reading Appendix A in which we present two standard ways of evaluating confidence score functions: the RC curve and the area under that curve (*AURC*).

4 How to attack uncertainty estimation

Attacking the uncertainty estimation "capacity" of a model is essentially sabotaging the partial order induced by its confidence score κ, for example, by forcing κ to assign low confidence scores to correct predictions, and high confidence scores to incorrect predictions (an even more refined attack, discussed in Section 6, is to completely reverse the partial order, which could be useful when the loss used is not 0/1). An adversarial example that specifically targets uncertainty estimation can be crafted by finding a minimal perturbation ϵ for an input x, such that $\tilde{x} = x + \epsilon$ would cause κ to produce a bad (in terms of selective risk) partial order on its inputs **without** changing the model's accuracy. The most *harmful* attack on κ for a model f should hold for every two adversarial instances with their ground truth:

$$\kappa(\tilde{x_1}, \hat{y}_f(\tilde{x_1})|f) \leq \kappa(\tilde{x_2}, \hat{y}_f(\tilde{x_2})|f) \iff \mathbf{Pr}_P[\hat{y}_f(\tilde{x_1}) \neq y_1] \leq \mathbf{Pr}_P[\hat{y}_f(\tilde{x_2}) \neq y_2].$$

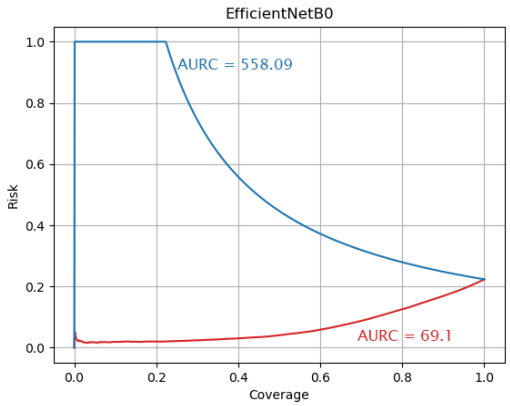

Figure 3: The blue curve shows the hypothetical worse possible RC curve of EfficientNetB0 trained on ImageNet, compared to its actual empirically measured RC curve (red curve).

Figure 3 shows such a hypothetical worse-case RC curve scenario, in which all of its incorrectly predicted instances are assigned more confidence than its correctly predicted ones. An end user setting a threshold to get a coverage of 0.22 over inputs would suffer a 100% error rate. An RC curve of our proposed attack showing similar actual damage can be observed in Figure 4. Note that in this example when a 0/1 loss is used, if we were to completely reverse the partial order induced by κ, it would not result in any more damage to the selective risk for any coverage.

The restriction of not changing the model's accuracy allows an attacker to only harm the uncertainty estimation without attracting suspicion by causing a higher error rate for the model. This constraint underscores the fact that it is not necessary to change the model's predictions in order to harm its capacity to estimate uncertainty. This restriction could be lifted based on the goal of the attack (if, for example, the attacker does not mind that the attacked model's accuracy is being reduced).

This type of attack also enjoys two benefits unique to it and not shared with common adversarial attacks: Firstly, the perturbations needed for this type of attack are inherently smaller than those needed for most adversarial attacks. Perturbations with too large magnitudes could cause the crafted example to cross the decision boundary and thus harm the model's accuracy (violating our restriction). Secondly, its success is not binary: The goal of most adversarial attacks is to push an input over the decision boundary. If the perturbation magnitude was insufficient to do that, the attack has failed. In our approach, the attack is partially successful even if it causes a single incorrect prediction to get a higher confidence score than a correct one, with the magnitude of the perturbations only contributing to the potency of the attack (creating a worse partial order and creating a larger confidence gap, numerically, between the correct and incorrect samples).

Algorithm 1 summarizes the general template for our proposed attack, dubbed 'ACE' (Attack on Confidence Estimation). In black-box settings, the proxy $\hat{f}$ for the attacked model f is an ensemble of models not consisting of f itself (with $\hat{y}_f(x)$ requiring a query to f to obtain its label for the input x), and in white-box settings, $\hat{f} = f$. The derivative of κ w.r.t. x, in the case of the softmax score, is the derivative of the y prediction component of $\hat{f}$ w.r.t. to the input x, where $y = y_f(x)$. We show in our white-box experiments that attacking the softmax score specifically is sometimes preferable, even if the κ used is not the softmax score, which is why we target the softmax score in black-box settings. We speculate that this is due to the fact that softmax allows us a direct "handle" on the prediction's margin in cases where κ does not (for example, in Section 5.4 we show that targeting the selector head of SelectiveNet that is used for rejection is less effective than targeting its softmax score).

While the scope of this paper is classification, the algorithm could easily be modified for regression tasks, in which the uncertainty is often measured by the variance of several outputs made by the model (e.g., the variance of outputs provided by an ensemble of DNNs). A simple conversion of this algorithm could, for example, define any instance with a loss above some threshold (such as the median loss on some validation set) as an "incorrect prediction" and loss values below the threshold as "correct predictions". The κ being attacked would be the variance of the outputs, causing an

Algorithm 1 Attack on Confidence Estimation

function ACE($f, \hat{f}, \kappa, x, y, \epsilon, \epsilon_{decay}, max_iterations$)
  $\eta \leftarrow sign(\nabla_x \kappa(x, \hat{y}_f(x)|\hat{f}))$ ▷ κ is calculated with $\hat{f}$ on the label predicted by f
  for $i < max_iterations$ **do**
    if $\hat{y}_f(x) == y$ **then** ▷ f is correct, decrease confidence for x
      $\tilde{x} \leftarrow x - \epsilon \cdot \eta$
    else ▷ f is incorrect, increase confidence for x
      $\tilde{x} \leftarrow x + \epsilon \cdot \eta$
    if $\hat{y}_f(\tilde{x}) == \hat{y}_f(x)$ **then**
      return $\tilde{x}$ ▷ label is unchanged so accuracy will be unharmed
    else
      $\epsilon \leftarrow \epsilon \cdot \epsilon_{decay}$
  return x ▷ insufficient $max_iterations$ & ϵ too big

Algorithm's arguments: f is the attacked model, $\hat{f}$ is the proxy for f, κ is the confidence scoring function, x is the input, y is the true label, ϵ is the initial magnitude of the perturbation, ϵ_{decay} is the rate at which ϵ decreases when the attack changes the input's prediction and $max_iterations$ is the number of ϵ decrease steps.

increase in variance for the inputs for which the model had a low loss, and a decrease in variance for inputs for which the model had a high loss.

5 Experiments

We evaluate the potency of ACE using several metrics: AURC ($\times 10^3$), *Negative Log-likelihood* (NLL) and *Brier score* [1] commonly used for uncertainty estimation in DNNs, and defined in Appendix B. We test several architectures on the ImageNet [4] validation set (which contains 50,000 images), implemented and pretrained by PyTorch [21], with the exception of EfficientNetB0, which was implemented and pretrained by [26]. We test ACE in black-box settings using three values for ϵ: 0.0005, 0.005 and 0.05 (with RGB intensity values between 0 and 1). We follow [16] in using an ensemble to craft the adversarial examples, in our case consisting of ten ResNet50 models trained on ImageNet. For "easier" white-box settings we test ACE with even smaller ϵ values: 0.00005, 0.0005 and 0.005. To the best of our knowledge, some of our tested ϵ values were never used by standard adversarial attacks. Due to ACE refraining from changing the originally predicted labels and decreasing ϵ iteratively if needed, the effective ϵ values selected by ACE are smaller and noted in the results (as "Effective ϵ"). In all of our experiments, we set the hyperparameters arbitrarily: $\epsilon_{decay} = 0.5, max_iterations = 15$. We did not try to optimize these, and we expect the algorithm to reach better results with a smaller ϵ_{decay} if we also increase $max_iterations$ (since it would find bigger ϵ values that could be used without changing the predictions).

5.1 Attacking softmax uncertainty

Table 1 shows the results of using ACE for different values of ϵ under black-box settings. Note that for larger values of ϵ, the effective ϵ is about half the size, meaning that even fewer resources are required for a very harmful attack. Under attack by ACE with $\epsilon = 0.05$, the AURC is $3.5 - 6.5$ times worse, and both the NLL and Brier score have worsened by about 100%, meaning that the uncertainty estimation of the model is devastated while the accuracy is left unharmed – just as ACE intended. The RC curves included in Appendix C.2 provide an important insight the metrics cannot: while on average, larger ϵ values harm uncertainty estimation more, sometimes for larger coverages, a smaller ϵ is more harmful. For example, the selective risk for VGG16 above coverage 0.6 is *higher* for $\epsilon = 0.005$ than it is for $\epsilon = 0.05$ (indicated by the purple curve being under the blue curve). This artifact may either be inherent to such black-box attacks, or it may be due to many attacked instances losing their "adversarial trajectory" by taking overly large steps, and could be fixed with an iterative version of ACE further discussed in Section 6. Figure 4 shows an RC curve for EfficientNetB0 with ACE under white-box settings (the legend presents the different values for ϵ, and the number next to each curve with its matching color is the area under the curve (AURC) $\times 1000$). Results for the even more potent white-box setting and its RC curves are included in Appendix C.1. From these results,

we observe that for $\epsilon = 0.005$, the AURC is about seven times worse and the NLL and Brier score are about three times worse.

Table 1: Results of ACE under black-box settings on various architectures pretrained on ImageNet with softmax as the confidence score.

Softmax score	ϵ	Effective ϵ	AURC	NLL	Brier Score	Accuracy
ResNet50	0	0	69.9	0.963	0.336	76.01
	0.0005	0.00046	141.9	1.308	0.474	76.01
	0.005	0.003353	436.9	2.615	0.652	76.01
	0.05	0.020622	456.6	2.494	0.656	76.01
EfficientNetB0	0	0	69.1	0.958	0.322	77.67
	0.0005	0.000495	78.8	1	0.342	77.67
	0.005	0.004459	185.1	1.334	0.47	77.67
	0.05	0.029023	300.9	1.775	0.585	77.67
Mobilenet V2	0	0	89.7	1.147	0.386	71.85
	0.0005	0.000493	104.3	1.217	0.416	71.85
	0.005	0.004276	236.9	1.745	0.578	71.85
	0.05	0.025108	317.8	2.148	0.628	71.85
DenseNet161	0	0	66.8	0.945	0.326	77.15
	0.0005	0.000493	81	1.022	0.358	77.15
	0.005	0.00427	209.8	1.577	0.508	77.15
	0.05	0.026584	359.7	2.1	0.585	77.15
VGG16	0	0	80.5	1.065	0.366	73.48
	0.0005	0.000493	93.7	1.135	0.395	73.48
	0.005	0.004296	207.8	1.634	0.544	73.48
	0.05	0.026088	298.2	1.985	0.602	73.48

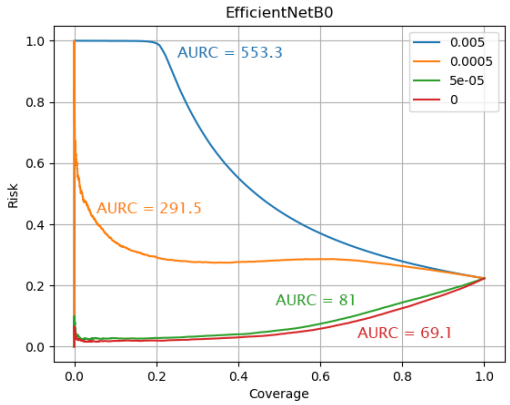

Figure 4: The RC curves of EfficientNet while under white-box attack by different values of ϵ. The colored numbers next to each matching colored curve correspond to the area under the curve (AURC) $\times 1000$. For attacks with $\epsilon = 0.005$, an end user asking the model only for its 20% most confident predictions will get almost 100% wrong predictions.

5.2 Attacking deep ensemble uncertainty

Table 2 shows the results of using ACE under black-box settings, in which we use one ensemble consisting of five ResNet50 models to attack another ensemble also consisting of five ResNet50 models, all trained separately on ImageNet. For $\epsilon = 0.05$, the AURC is about six times worse and the NLL and Brier score have worsened by 100% or more. In the resulting RC curve, which is

Table 2: Results of ACE under black-box settings on an ensemble of size 5 consisting of ResNet50 models trained on ImageNet.

	ϵ	Effective ϵ	AURC	NLL	Brier score	Accuracy
ResNet50 Ensemble	0	0	63.8	0.883	0.317	77.61
	0.0005	0.00049	82.7	0.966	0.357	77.61
	0.005	0.00396	279.1	1.582	0.563	77.61
	0.05	0.02342	381	2.000	0.633	77.61

included in Appendix D.2, we can make the same observation we did when attacking softmax: for large coverages, a smaller ϵ causes greater damage to the selective risk (in this case, the selective risk for any coverage above 0.45 is slightly higher for $\epsilon = 0.005$ than it is for $\epsilon = 0.05$). For white-box settings, we test ACE using ensembles of different sizes consisting of ResNet50 models trained on ImageNet. Appendix D.1 shows the results of these experiments, from which we observe that for $\epsilon = 0.005$, the AURC degrades about eightfold. An interesting observation is that the bigger the ensemble, the more resilient it is to ACE, and even the smallest ensemble is more resilient than a single ResNet50 model (in both black-box and white-box settings). The confidence score used is the average of the softmax scores over all the models in the ensemble. Note that since the scope of this paper does not include explicit attempts to enhance these techniques' resiliency to our attack, we have not used adversarial training for the ensemble as was suggested by [15] (which was found to improve uncertainty estimation). We did not want to give a perceived advantage to Deep Ensembles over other techniques in this respect, and as the results show, Deep Ensembles are more resilient even without adversarial training.

5.3 Attacking Monte Carlo dropout

Table 3: Results for MC-Dropout of ACE under black-box settings on models pretrained on ImageNet with predictive entropy over several dropout-enabled passes producing the confidence score.

MC-Dropout	ϵ	Effective ϵ	AURC	NLL	Brier Score	Accuracy
MobileNet V2 30 passes	0	0	94.5	1.143	0.386	71.79
	0.00005	0.00049	104.4	1.207	0.413	71.79
	0.0005	0.00427	212.6	1.71	0.575	71.84
	0.005	0.02514	275.2	2.145	0.634	71.79
MobileNet V2 10 passes	0	0	95.1	1.148	0.387	71.58
	0.00005	0.00048	104.9	1.21	0.413	71.72
	0.0005	0.00427	213.2	1.713	0.573	71.7
	0.005	0.02514	275.3	2.146	0.631	71.61
VGG16 30 passes	0	0	86.1	1.056	0.366	73.22
	0.00005	0.00049	95	1.117	0.392	73.35
	0.0005	0.0043	188.9	1.573	0.541	73.34
	0.005	0.02611	261.1	1.978	0.613	73.28
VGG16 10 passes	0	0	86.5	1.066	0.368	73.18
	0.00005	0.00048	95.9	1.123	0.392	73.13
	0.0005	0.00429	189	1.578	0.537	73.08
	0.005	0.02608	261.5	1.984	0.609	73.04

We test ACE on MC-Dropout using predictive entropy as its confidence score. We first experiment with ACE under white-box settings and test two methods of attack: an *indirect method*, in which ACE targets the softmax score of a single dropout-disabled forward pass, and a *direct method*, in which ACE targets the predictive entropy produced from N dropout-enabled forward passes. In both methods, the model's confidence in the prediction is calculated by predictive entropy over N forward passes with dropout enabled. From the results presented in Appendix E.1, we can observe that while for small ϵ values the direct method is slightly more harmful, for bigger values the indirect softmax method is significantly more harmful and on par with results of attacking those models when their κ is softmax instead of being calculated by MC-Dropout. These results lead us to conclude that for most purposes, an indirect softmax attack could be more harmful than a direct attack on the predictive entropy used by MC-Dropout. Using this knowledge, we attack MC-Dropout in black-box settings identically to how we attacked softmax: by using an ensemble as a proxy and attacking its softmax score, without using dropout at all. Observing Table 3 for the results, we notice that for black-box settings, MC-Dropout seems somewhat more resilient compared to using softmax as κ for these same models. The number of forward passes made by the models do not seem to affect ACE's effectiveness. The RC curves for the black-box settings are provided in Appendix E.2.

5.4 Attacking SelectiveNet

Finally, we test ACE under white-box settings attacking SelectiveNet using two different architectures for its backbone, namely ResNet18 and VGG16, both trained on CIFAR-10 [13]. The confidence score is given by the selector head of SelectiveNet, which was trained to be used for a specific coverage. Since the model is trained to give predictions with an optimal selective risk for the coverage for which they were optimized, the selective risk at that coverage is the most relevant metric by which to measure the performance of a SelectiveNet model. The paper that introduced SelectiveNet [7]

emphasized the need to calibrate SelectiveNet's trained coverage to fit the desired *empirical coverage*. For fairness and to reach flawless calibration, we train the models to fit a trained coverage of 0.7, but evaluate the selective risk according to the model's true empirical coverage on the test set (as observed without any attack). For example, the empirical coverage of SelectiveNet with ResNet18 as its backbone on the test set is 0.648, and the selective risk we show in Table 4 is the model's risk for that exact coverage. As additional metrics, we also provide the AURC, NLL and Brier score (even though they do not reflect the true goal of the network in this context). We consider a *direct attack*

Table 4: Results of ACE on SelectiveNet when used with ResNet18 and VGG16 as a backbone, and trained on CIFAR-10. The empirical coverages (perfectly calibrated) for ResNet18 and VGG16 are 0.648 and 0.778, respectively. The selector is always used as the confidence score. A direct ACE attack targets the selector, and an indirect attack targets the softmax score.

	ϵ	**Effective ϵ**	**Selective Risk**	**AURC**	**NLL**	**Brier Score**	**Accuracy**
ResNet18 Backbone Direct (selector head)	0	0	0.0012	5.04	0.204	0.083	94.83
	0.00005	0.00005	0.0015	5.56	0.211	0.085	94.83
	0.0005	0.00049	0.0059	12.16	0.283	0.102	94.83
	0.005	0.00377	0.0747	132.08	0.547	0.151	94.83
ResNet18 Backbone Indirect (softmax score)	0	0	0.0012	5.04	0.204	0.083	94.83
	0.00005	0.00005	0.0015	5.5	0.215	0.087	94.83
	0.0005	0.00049	0.0056	11.69	0.314	0.114	94.83
	0.005	0.00385	0.0796	150.07	0.648	0.162	94.83
VGG16 Backbone Direct (selector head)	0	0	0.0067	7.46	0.259	0.105	93.54
	0.00005	0.00005	0.0075	7.87	0.267	0.107	93.54
	0.0005	0.00049	0.0163	12.57	0.34	0.128	93.54
	0.005	0.00402	0.0772	116.1	0.703	0.175	93.54
VGG16 Backbone Indirect (softmax score)	0	0	0.0067	7.46	0.259	0.105	93.54
	0.00005	0.00005	0.0075	7.86	0.268	0.108	93.54
	0.0005	0.00049	0.0163	12.52	0.35	0.134	93.54
	0.005	0.00404	0.0829	122.46	0.761	0.181	93.54

on the model's selector head and an *indirect attack* via the model's softmax score. In both cases the final confidence score is measured using the selector head. As seen in the results in Table 4, an indirect attack by softmax seems more harmful than a direct one. For example, for SelectiveNet with a VGG16 backbone, the error over the normal not "rejected" instances is 0.67%, but it is 7.72% for directly (selector) crafted adversarial instances ($\epsilon = 0.005$) and 8.29% for indirectly (softmax) crafted adversarial instances ($\epsilon = 0.005$). For context, we include in Appendix F ACE results on standard (not selective) architectures trained on CIFAR-10 with the confidence score being the softmax score.

6 Conclusions

We have presented a novel attack on uncertainty estimation, capable of causing significant damage with even small values of ϵ (compared to standard adversarial attacks) and without changing the model's original predictions. This attack severely harms some of the most popular techniques for estimating uncertainty. We can think of several ways in which the attack could be further improved: First is optimizing the perturbations in an iterative process, similar to how *BIM* [14] or *PGD* [18] take FGSM steps iteratively. This may not only increase ACE's potency, but may also fix the artifacts of larger values of ϵ leading to a lower selective risk than smaller values (observed in Sections 5.1 and 5.2). This might also have the added benefit of using more of the ϵ resources that were allocated to the attack, thus increasing the effective ϵ. Low ratios of $\frac{\text{Effective } \epsilon}{\epsilon}$ such as observed from our results mean unspent resources for the attack.

Another improvement could rely on the observation that in both MC-Dropout and SelectiveNet, targeting softmax with ACE is usually more harmful, even though it is not the κ used for confidence estimation. There might be an even better signal processing method with which to attack uncertainty estimation. The results of this attack also teach us that some methods are more resilient than others. We hope that this work will encourage efforts to examine the resiliency of other and new uncertainty estimation methods, as well as creating defense mechanisms for them. Finally, in the case of none-0/1 loss functions, an attack on uncertainty estimation might need to completely reverse the partial order observed by the model, which would capture the gaps in uncertainty between two correct instances or between two incorrect instances. Such an attack could be used for regression tasks as well, more naturally than our suggested simple conversion of ACE, discussed in Section 4, which would use a specific threshold on the value of the loss as a pivot.

References

[1] G. W. Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1):1, Jan. 1950.

[2] L. P. Cordella, C. De Stefano, F. Tortorella, and M. Vento. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147, 1995.

[3] C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. 2016.

[6] Y. Geifman and R. El-Yaniv. Selective classification for deep neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4885–4894, 2017.

[7] Y. Geifman and R. El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. *CoRR*, abs/1901.09192, 2019.

[8] Y. Geifman, G. Uziel, and R. El-Yaniv. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*, 2018.

[9] Y. Geifman, G. Uziel, and R. El-Yaniv. Boosting uncertainty estimation for deep neural classifiers. *CoRR*, abs/1805.08206, 2018.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[12] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[13] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world, 2017.

[15] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. 2017.

[16] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016.

[17] D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, May 1992.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[19] N. Tishby, E. Levin, and S.A. Solla. Consistent inference of probabilities in layered networks: Predictions and generalizations. pages 403–409 vol. 2, 1989.

[20] R. M. Neal. *Bayesian Learning for Neural Network*. PhD thesis, Dept. of Computer Science, University of Toronto, 1994.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[22] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks, 2014.

[25] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[26] R. Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code will be publicly available upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] The code will be publicly available upon acceptance.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]