Uncertainty and Robustness Estimation Methods for Graphical Neural Networks
Towards a Better Understanding of Architecture Performance

Carlos Torres Tejaswi Nanjundaswamy Shivkumar Chandrasekaran
Mayachitra, Inc.,
{Torres, Tejaswi, Shiv}@mayachitra.com

Abstract

This study investigates the ubiquitous problem of random and adversarial noisy data, which can easily fool the most sophisticated Deep Neural Network (DNN) architectures. Noisy examples are difficult for humans to detect and very easy to generate in large amounts. DNN-based classifiers are believed to be the solution to image and video analysis. The hypothesis of this work assumes that training spaces (i.e., training data) are organized into small decision bubbles, which define the decision boundaries of that particular class. This hypothesis is supported by extensive results, which indicate that existing architectures are not only over-fitted and do not generalize well, but are very sensitive to random and targeted attacks. Therefore, the objective of this paper is to bring to light Uncertainty and Robustness Estimation Methods (UREM) to assess, evaluate, and visualize the robustness and generalizability of DNNs using bubble spaces and perturbation matrices. Experimental results on a standard data set (and its variants) with four popular CNN architectures indicate that the network depth and number of parameters decide the accuracy and robustness to random/targeted noise. We observe that robustness to random noise is generally correlated to depth of networks, i.e., deeper networks (e.g., ResNet), are more robust to random noise (and generally more accurate), compared to a shallow network (e.g., AlexNet). Whereas robustness to targeted noise depends on number of parameters in the network, i.e., networks with lesser parameters (e.g., InceptionV3) are more robust to targeted noise than networks with more parameters (e.g., VGG16).

1. Introduction

This study introduces perturbation based methods to estimate and visualize uncertainty in deep neural network (DNN) models and architectures. Inspired by the theory behind perturbation based condition number estimation [7] that measures the robustness of linear systems, the proposed approaches estimate the confidence of a model by analyzing its performance for different perturbations inputs and levels. In the context of adversarial attacks there has been significant amount of research in the machine learning community on perturbations to fool trained models [15]. The proposed approach derives uncertainty estimates and confidence measures of the model by measuring the amount of change required to fool a trained model. On the one hand, output changes of a model due to big changes to an input indicate that the model is highly robust. On the other hand, output changes due small perturbations to an input indicate that the model has low confidence and low robustness. The relationship between input modification and output alteration is shown, with an example, in Figure 1, where the diameter of the bubble spaces (i.e., decision boundaries for the instance of that class) indicate that larger modification is required to alter a network's output, which indicates higher (or lower) robustness and certainty levels.

[0]Carlos Torres and Tejaswi Nanjundaswamy co-authored this paper as equal contributors

1.1. Technical Background

The UREM methods are used to derive uncertainty estimates or a confidence measure of DNNs by measuring the amount of change required to fool a trained model. The change mechanism is based on random and targeted perturbations. For instance, if a big change is required to change the output of a model, it is an indication of its robustness. Similarly, output changes with small perturbations to its input indicate lower confidence of the model. Published research [4, 12, 21, 22, 25] indicates that "small bubble" or system overfitting is an active and relevant area of research.

Bubble Spaces. In layman terms, bubbles or bubble spaces are regions that contain class labels (with high confidence) and or decision statements, such as the ones shown in Figure 1, where obtaining a new image class label (i.e., a cat to be misclassified and confused with a dog) requires a modification of the image by various amounts. If the modification is small or imperceptible, then the bubbles are assumed to be adjacent to each other (and possibly small) as shown in Figure 1 (a). If the modification is large and perceptible, the assumption is that the bubbles are far apart (and possibly large) as shown in Figure 1 (b).

Definition of Generalization. Assuming a two-class problem, where $f^{-1}(1)$ has a single connected component, which contains all the training samples of class 1, indicates that f has "strongly generalized" class 1 and similarly for class 2. However, if a single connected component of $f^{-1}(1)$ has 2 training samples of class 1, this indicates that f "weakly generalizes" those 2 training samples. Alternatively, if f is the "true" classifier, then all components of $f^{-1}(1)$ are truly in class 1. Note that this cannot be determined by looking at the training set, and requires an oracle. The "how" to compute an f, which strongly generalizes to all instances remains an *open* problem.

Instead, if the connected component containing a training sample x in class 1 is very close to a connected component of class 2, then clearly such a classifier will not behave well in practice. It can be deduced that such classifier is not robust – this is the small bubble case. The absence of this is easy to check; however, its presence is harder to verify. The assessment of robustness by checking for small bubble spaces is discussed in the following subsections.

Uncertainty Estimation. This work introduces methods to derive and measure two types of uncertainty estimates for neural network models: aleatoric or random uncertainty, which is attributed to ill-defined problems such as sub-pixel precision in segmentation boundaries; epistemic (type C) or systematic uncertainty, which is attributed to multiple hypothesis (i.e., multiple viable sets of training parameters) and can be addressed by infinite data.

Variational Bayesian Inference. One of the most successful approaches to scalable variational bayesian inference is the "reparametrization trick"[9] to estimate the gradients for training deep neural networks. This trick significantly reduces the variance of the gradient estimation compared to other approaches [5, 16], and more accurate gradients means more efficient and scalable training. It forms the foundation of numerous recent papers and frameworks for deep variational inference [1, 3, 14, 23].

There are several improvements to the variational reparametrization trick that have been proposed since its original publication. The local reparametrization trick [8] further reduces the variance of the gradient estimator when the posterior used in a neural network is a factorized diagonal Gaussian (i.e., learning a mean and variance for each parameter independently), which is a very common assumption. Their work also proposes using this with multiplicative Gaussian dropout on the intermediate feature activations of a neural network, and proposes an approximate prior to enable using the reparametrization trick to learn the variance of the Gaussian dropout noise. Another development using the reparametrization trick is importance-weighted autoencoders [2], which use multiple stochastic samples in parallel to improve the training of the estimator. There are two ways to consider uncertainty in the input object primitive detections (or in general missing data). UREM assumes that the input is "missing at random", or "missing not at random"; where the latter case implies that there is information conveyed via the presence or absence of a feature. Such information can be used to infer the value of the missing data.

For instance, in the medical field, when measuring test detection reliability, one has to consider that healthy patients do not usually take test if it is expensive. Then, a patient not having taken the test (missing such data) indicates that the patient is more likely to not have that ailment, since healthy patients usually do not undergo expensive testing. In overhead observations, the uncertainty of an object detector can be correlated with certain information. For example, a boat detector might be weaker when detecting a boat on a trailer in a parking lot, because it is missing the context of water that boats are usually floating on. Or, perhaps the confidence of a car detector is lower when they are closely packed together, as in a parking lot. The circumstances where detectors confidence is not be truly "random", but where be a small amount of contextual information helps to minimize uncertainty.

Literature search indicates that the "large bubble" or system generalizability is in its infancy and can be of great help in understanding training limitations and define industry practices and techniques to train better systems. UREM approaches are based on random and targeted perturbations to the layers. UREM compares four popular neural network architectures developed in the recent past for object identification using the ImageNet dataset.

Related Work. Perhaps the most similar work to the proposed UREM includes the two common methods used to successfully attack networks: fast gradient sign method (FGSM) and targeted fast gradient sign method (T-FGSM) and its variant (I-FSGM) [11]. FGSM computes an adversarial image by adding pixel-wide perturbations in the same direction as the gradient i.e.:

$$x^{adv} = x + \epsilon\cdot \operatorname{sign}(\bigtriangledown_x J(x, y_{true})), \quad (1)$$

where x is the unmodified input image, x^{adv} is the perturbed adversarial image, J is the classification loss function, and y_{true} is the true label for input x. Similarly, the targeted fast gradient sign method (T-FGSM) computes a gradient at each step in the opposite direction of the gradient with respect to the largest class given by:

$$x^{adv} = x - \epsilon\cdot \operatorname{sign}(\bigtriangledown_x J(x, y_{target})), \quad (2)$$

where y_{target} represents the target label for the adversarial attack. The T-FSGM variation known as the iterative fast gradient sign method (I-FSGM) takes T-gradient steps each of magnitude α instead of a single step t such that:

$$x_{t+1}^{adv} = x_{t+1}^{adv} + \alpha \cdot \operatorname{sign}(\nabla_x J(x_t^{adv}, y)), \tag{3}$$

with $x_0^{adv} = x$ and $\alpha = \frac{\epsilon}{T}$.

The main difference between UREM and the previously listed methods is that in UREM ϵ is not bounded. UREM uses the Adam optimizer with variable-step size, where ϵ is the target variable, representing robustness levels. A targeted perturbation similar to UREM is proposed in [13], but they change the target label in every iteration to speed up the process, instead the proposed methods do a comprehensive search for every target label. A related topic to this work is estimating robustness certificates [19, 26, 24] by approximating the classifier structure. However, the practical results and applications are limited to only smaller datasets (e.g., CIFAR/MNIST) or a limited number of layers. Instead the proposed approach is generic and applicable to all networks, where back propagation of the gradient calculation is possible.

1.2. Problem Statement as Bubble Spaces.

Let the input be in $\mathbb{R}^n$. The classifier f is then a function from $\mathbb{R}^n$ to $\mathbb{R}$. For simplicity assume that is a 2-class problem. Suppose $f(x) = 1$ means x is in class 1 and $f(x) = 0$ means x is in class 2. Now consider the subset of $\mathbb{R}^n$ given by $f^{-1}(1)$. The level set f corresponds to the value 1, where the assumptions is that f is a continuous function. Hence, $f^{-1}(1)$ is composed of multiple connected components. In layman terms, bubbles or bubble spaces are regions that contain class labels (with high confidence) or decision statements, such as the ones shown in Figure 1. In this example, obtaining a new image class label (i.e., a cat to be miss-classified as a dog) requires a modification of the image. If the modification is small or imperceptible we say that the bubbles are adjacent to each other (and possibly small if this can be done for a large number of inputs). If the modification is large and perceptible then the bubbles are considered to be far apart and, possibly, large.

The "small bubble", or model overfitting, is an active area of research of very high relevance [4, 12, 21, 22, 25]. However, this also indicates that the "large bubble", or model generalizability, is in its infancy and can be of great help in understanding training limitations and define industry practices and techniques to train better systems. The performance of neural networks is evaluated in terms of accuracy, Precision and Recall, or TPR and FNR. However, it is hypothesized that neural networks are over-fitted solutions to the data; therefore, these metrics might not be a complete performance representation. Hence we propose

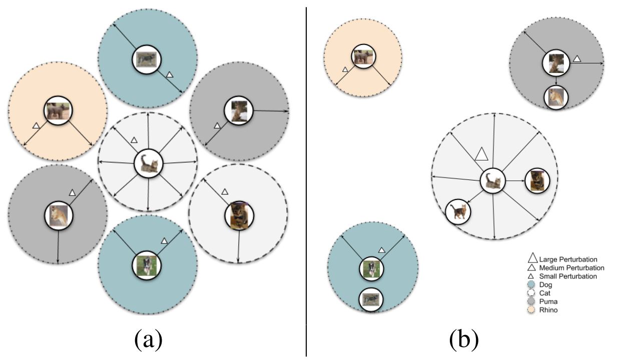


Figure 1. Bubble spaces of (a) are smaller than that of (b) indicating that (a) is more over-fitted. We say that the network has *weakly generalized* if a bubble contains more than two training samples from the correct class. The bubbles in this figure are for the class labels: dog, cat, puma rhino.

approaches for uncertainty estimation that are based on introducing random and targeted perturbations. The methods compare four well known neural network models developed in the recent past for object identification, namely, AlexNet [10], VGG16 [18], ResNet [6], and InceptionV3[20] trained using the ImageNet dataset [17].

The system overview is shown in Figure 2, with variations developed for random and targeted perturbations. The methods are devised and implemented to assess aleatoric and epistemic uncertainty.

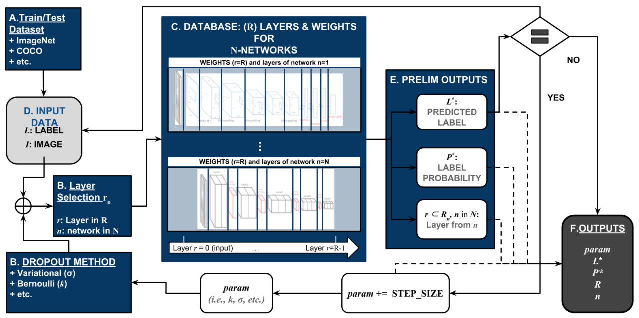


Figure 2. Overall block diagram of the process to estimate uncertainty and robustness of deep neural networks (DNNs). Block (A) represents the training input dataset used to train a network. The attack properties are selected in block (B), where the options include: attack type (random or targeted), and set of layers (0: input layer, ..., R-1: output layer, R: network weights). The bank/collection of DNNs is shown in block (C), where various architectures are available. The test input data is introduced in block (D). Preliminary outputs are shown in block (E) for each iteration. The final output summary of the network analyses is shown in block (F).

Model fitting to a dataset is not unique. There can be many equivalent hypotheses (sets of parameters) that fit the data, which results in solution(s) ambiguities. Standard methods to deal with ambiguities include regulation imposed by priors or by adding more training data, which is not always possible. This study tackles this problem by combining the processes described in the random and targeted attack diagrams to perturb individual network layers, evaluate the network robustness to perturbations, and assess uncertainty of the associated predictions.

Technical Innovations & Contributions Include:

- Evaluation methods for DNN robustness to targeted and random attacks
- A technique to select and attack DNN layers
- Robustness and certainty quantification methods based on Bubble Spaces to visualize and evaluate DNNs
- Comprehensive results for full ImageNet classifier and dataset
- A prototype system to evaluate uncertainty and robustness of DNN systems and architectures[1]

2. Datasets

The methods are implemented and evaluated on the standard dataset of ImageNet (1000 classes) and its variants. The objective of including the variants is to account for potential artifacts introduced by properties of the data in regards to sparse vs. dense (i.e., sparse: large number of classes with small number of examples per class; dense: small number of classes with large number of examples per class).

ImageNet. ImageNet can be considered a sparse dataset. The above described neural network models are trained on the ImageNet dataset [17]. The uncertainty estimation methods use the corresponding validation set from the ImageNet database, which consists of 1000 object classes with 50 images per class. The overall dataset consists of more than 14 million hand-annotated images, classified into over 20000 categories. UREM uses the "trimmed" 1000 non-overlapping classes from the annual ImageNet challenge.

3. System Overview

The proposed approach perturbs the input to measure the model's uncertainty. The attacks are random and targeted and the baseline prediction is the network's classification output using the unmodified version of the input image.

3.1. Robustness and Uncertainty Based Attacks

Random Attacks. The random attack process adds random Gaussian noise to each individual pixel from each channel of the input images. The standard deviation (σ) of the noise is increased to a breaking point, which is defined as the standard deviation value at which the classifier output deviates from the base prediction. This breaking point is measured for each input data of the validation set that was correctly classified and averaged to compute an overall robustness score for each model. The overall block diagram of the system is shown in Figure 3(a). A higher overall score (i.e., standard deviation value) for a model indicates better robustness to image random manipulations.

Targeted Attacks. The random attacks approach is enhanced to include *targeted perturbations* that change the classifier output to a given target class with high confidence. The overall block diagram of the system is shown in Figure 3 (b). The targeted perturbations are introduced by calculating the gradient for each pixel of each channel and modifying the input using this gradient. We calculate the measure of uncertainty as the maximum of absolute difference between the original image and the modified image for which the model prediction matches the target class with 90% or more probability. This targeted perturbation is performed for each of the other classes to get a measure of a model's robustness to change for a given input image. A higher overall score for a model indicates better robustness to image manipulations. The evaluations compare well known architectures trained on ImageNet and COCO.

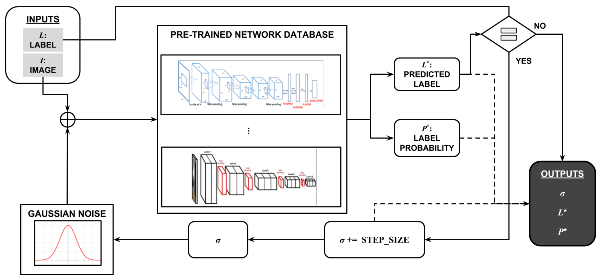
3.2. Neural Network Architectures

AlexNet: AlexNet [10] is the first deep convolutional neural network and it significantly outperformed all the prior approaches, reducing the top-5 error for ImageNet object classification from 26% to 15.3%. The network employees filters per layer with stacked convolutional layers. Convolutions of size 11×11, 5×5 and 3×3 are used along with max pooling, dropout, data augmentation ,and rectified linear unit (ReLU) activations after every convolutional and fully-connected layers. The optimization is done using stochastic gradient descent (SGD). Overall there are seven layers and approximately 60 million parameters.

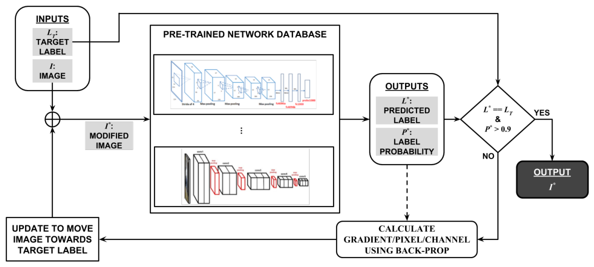
VGG16: VGG16 [18], similar to AlexNet, consists of 16 convolutional layers. It employs a very uniform architecture with 3×3 convolutions and several filters and is made up of approximately 138 million parameters.

ResNet: ResNet [6] introduced a novel architectural element of "skip connections" and also includes significantly more batch normalization. These skip connections allowed training a very deep network of 101 and 152 layers with a complexity lower than that of VGG16. Overall the models are made up of approximately 60 million parameters.

[1]docker image and code will be available online



(a)



(b)

Figure 3. Robustness to perturbation overview for (a) random perturbations and (b) targeted perturbations to the input (i.e., layer $r = 0$). For random perturbations, the inputs are an image and its class label; the image is passed through the inference process to estimate a label and its probability; we add Gaussian noise with variable variance (in "step_size" increments), until the network predicts a different label, which indicates that network is successfully fooled. The outputs are: the final value of the Gaussian-noise standard deviation, which fooled the network, and the predicted label (L^*) and label probability (P^*) for the noisy image. Similarly, for targeted perturbations the inputs are the raw image (I) and the target label (L_T); the preliminary outputs are the modified image (with modifications that veer it towards (L_T)); the final outputs are the modified image (I^*) and the maximum of the per-pixel differences between the input raw image and the modified image ($MAX(|I - I^*|)$).

InceptionV3: Inception [20] V3 is the latest and improved version of the original architecture with 311 layers and just under 24 millions parameters. This network is unique because it has two output layers when training. The primary output is a linear layer at the end of the network. The secondary output, known as auxiliary, contains the AuxLogits part of the network. UREM evaluates the primary output.

Problem Statement. Let the input be in $\mathbb{R}^n$. The classifier f is then a function from $\mathbb{R}^n$ to $\mathbb{R}$. For simplicity assume that is a 2-class problem. Suppose $f(x) = 1$ means x is in class 1 and $f(x) = 0$ means x is in class 2. Now consider the subset of $\mathbb{R}^n$ given by $f^{-1}(1)$. We call this the level set of f corresponding to the value 1. We assume that f is a continuous function. It then follows that $f^{-1}(1)$ will be composed of multiple disconnected components.

A network corresponds to an image space with lots of randomly scattered small bubbles is considered both overfitted and non-robust. Such networks are unlikely to perform satisfactorily even if they scored highly on the testing set. Unfortunately, UREM shows that many modern highly used networks fall into this category. Alternatively, a network with widely separated bubbles containing multiple training samples is considered to be weakly generalized and robust. This is the best one can hope and the goal is to develop a new set of networks in this category. Furthermore, a network is strongly generalized if it is weakly generalized and an oracle verifies that the bubbles are fully correct. This check is computational expensive; however, this is required properly designed and trained networks.

The $f'(x(t)).x'(t) = 0$ condition gives an implicit ODE for $x(t)$ that can be solved with a Differential Algebraic Equation (DAE) solver (indicates the need to maintain the constraint $f(x(t)) = 1$). A solver success indicates that the pair $(x(0), x(1))$ is weakly generalized by the classifier f. Alternatively, a solver failure indicates inability to effectively assess the classifier's generalizability.

4. Experimental Results

Random Perturbation Results. The random attack adds Gaussian random noise to each channel of each pixel of the input image and progressively increases the variance of this noise to find a breaking point. We use a step_size $= 1$ for standard deviation of the noise for 1000 test classes from the ImageNet training dataset with one image per class. The standard deviation values that causes the networks to change their predictions are recorded and used to assess levels of network uncertainties (by averaging it over all classes). The overall results are summarized in Table 1.

These results clearly indicated that a network's accuracy and robustness to random noise increases proportionally with depth (i.e., number of layers). Figure 4 shows a sample image for Non-Noisy (no noise added) and Noisy (Gaussian noise added) images.

Targeted Perturbation Results. The inputs are the raw images (I) and the target labels (L_T); the preliminary outputs are the modified images (with modifications that veer them towards the target label (L_T)); the final outputs are the modified images (I^*) and the maximum of the per-pixel differences between the input and the modified images ($MAX(|I - I^*|)$, i.e., $L_\infty(|I - I^*|)$). The targeted perturbations are introduced by calculating the gradient for each pixel of each channel and modifying the input using this gradient.

NEURAL NETWORK ARCHITECTURE: ACCURACY & ROBUSTNESS

	AlexNet	VGG 16	ResNet 152	Inception V3
Top-5 Error Rate	20.91	9.62	5.94	6.44
Number of Parameters (in millions)	$\sim$ 60	$\sim$ 138	$\sim$ 60	$\sim$ 23
Random Perturbation Uncertainty	66.3 (25.9%)	79.8 (31.2%)	136.4 (53.2%)	196.6 (76.8%)
Targeted Perturbation Uncertainty	3.2 (1.2%)	1.6 (0.6%)	2.4 (0.9%)	3.6 (1.6%)

Table 1. Comparison of classification accuracy, number of parameters, and robustness (random and targeted) of four neural network architectures. Top-rows have the names of the architectures. The second row shows the top-5 error rate. The third row shows the number of parameters. The last two rows is a measure of robustness. The mean standard deviation as a percentage of maximum pixel value (255) in brackets for the bottom two rows for random and targeted perturbations.

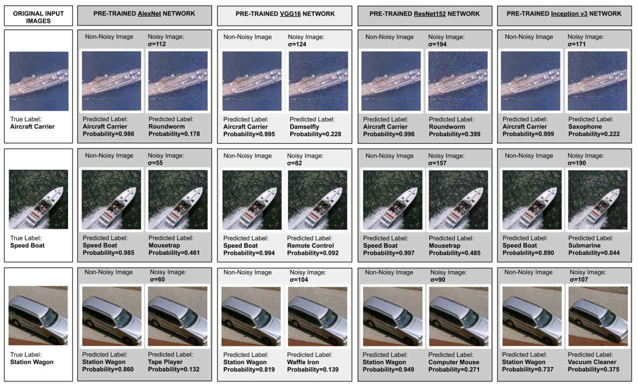

ORIGINAL INPUT IMAGES
PRE-TRAINED AlexNet NETWORK
PRE-TRAINED VGG16 NETWORK
PRE-TRAINED ResNet152 NETWORK
PRE-TRAINED Inception v3 NETWORK
Non-Noisy Image
Noisy Image: σ=112
Noisy Image: σ=124
Noisy Image: σ=194
Noisy Image: σ=171
True Label: Aircraft Carrier
Predicted Label: Aircraft Carrier Probability=0.986
Predicted Label: Roundworm Probability=0.178
Predicted Label: Aircraft Carrier Probability=0.995
Predicted Label: Damselfly Probability=0.228
Predicted Label: Aircraft Carrier Probability=0.996
Predicted Label: Roundworm Probability=0.399
Predicted Label: Aircraft Carrier Probability=0.999
Predicted Label: Saxophone Probability=0.222
Noisy Image: σ=55
Noisy Image: σ=82
Noisy Image: σ=157
Noisy Image: σ=190
True Label: Speed Boat
Predicted Label: Speed Boat Probability=0.985
Predicted Label: Mousetrap Probability=0.461
Predicted Label: Speed Boat Probability=0.994
Predicted Label: Remote Control Probability=0.092
Predicted Label: Speed Boat Probability=0.997
Predicted Label: Mousetrap Probability=0.485
Predicted Label: Speed Boat Probability=0.890
Predicted Label: Submarine Probability=0.044
Noisy Image: σ=60
Noisy Image: σ=104
Noisy Image: σ=90
Noisy Image: σ=107
True Label: Station Wagon
Predicted Label: Station Wagon Probability=0.860
Predicted Label: Tape Player Probability=0.132
Predicted Label: Station Wagon Probability=0.819
Predicted Label: Waffle Iron Probability=0.139
Predicted Label: Station Wagon Probability=0.949
Predicted Label: Computer Mouse Probability=0.271
Predicted Label: Station Wagon Probability=0.737
Predicted Label: Vacuum Cleaner Probability=0.375

Figure 4. Sample qualitative results. The first column (left-to-right) shows the original input images with their respective true labels. The second, third and fourth columns show a pair of images (Non-Noisy image and Noisy Image) that are classified using a pre-trained AlexNet, VGG16, ResNet152, and InceptionV3 networks, respectively. The information around the images includes: predicted labels and probabilities for the non-noisy image and noisy image and the noise standard deviation required to change the detection class.

An example image of a speed boat is selected, which is correctly classified by the original model, and find the modification required to change the classification label to another class with 90% probability. The image of a *speed boat* with visually imperceptible modifications that fooled the AlexNet network to classify it as a *cat* with very high confidence is shown in Figure 5. Notice that the per-channel differences shown in (d) do not have a clear or visible structure; therefore, these can be considered “noise”.

Bubble Spaces for Targeted Perturbations. Figure 6 shows the bubble spaces computed using the targeted per-

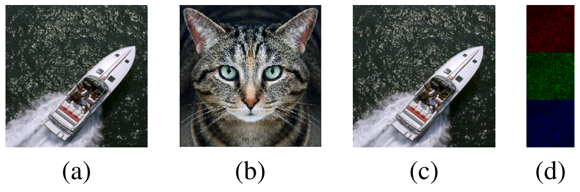


Figure 5. An example of neural network being fooled. (a) original speed boat image that is correctly classified; (b) the cat target-label image; (c) the image with visually imperceptible modifications that is misclassified by AlexNet as a cat with high confidence; and (d) the per-channel difference between images (a) and (c).

turbations method for AlexNet (a), VGG16 (b), ResNet152 (c), and InceptionV3 (d) architectures trained on ImageNet. The angular steps of 0.36 degrees correspond to the 1000 classes in the dataset, while the magnitude corresponds to the necessary max difference (as percent of the maximum pixel intensity) between the source and the modified image. The radius represents both the amount of modification needed to fool the network and its robustness.

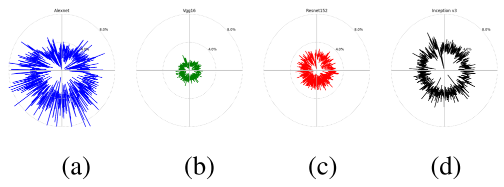
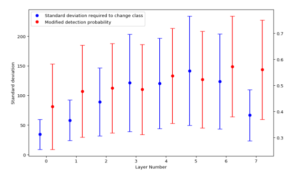


Figure 6. Bubble space computed for the *tabby-cat* image using (a) AlexNet (blue), (b) VGG16 (green), (c) ResNet152 (red), and (d) InceptionV3 (black) trained on ImageNet. Every angular step of 0.36° corresponds to one of 1000 classes and the magnitude corresponds to the max difference (percentage of maximum pixel value) between the original image and the modified image classified to target class with high probability.

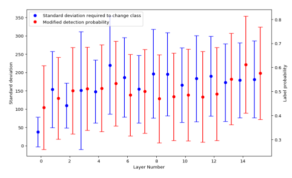
Targeted Perturbation Matrix Results The experiment uses targeted perturbations for pre-trained Alexnet, VGG16 ResNet152, and InceptionV3 models with one sample from each class perturbed to match each of the other 999 classes. For each of this perturbation we estimate the maximum difference $(MAX(|I - I^*|))$ to get a perturbation matrix to illustrate the robustness of a network for all the different classes. Figure 8 shows the perturbation matrices for the previously listed models. The results suggest that Alexnet is more robust as it requires larger perturbations to fool its predictions. Another interesting observation is the strong horizontal lines in these matrices. This indicates that the network is more robust for some classes than others.

Layer-wise Random Perturbations. This approach enables analysts to select the network and the layer (or layers) to attack with random noise and analyze for robustness. The random noise is multiplied (instead of being added) so that we can measure the percentage of noise required to cause output classification change. The noise multiplied is of 1 mean and standard deviation measures the percent noise.

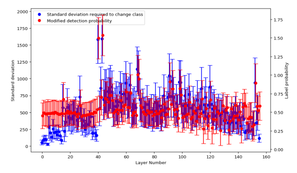
Experimental results are shown in Figure 7 for AlexNet (a), VGG16 (b), Resnet152 (c), and InceptionV3 (d). The results indicate that layers closer to the input are more resilient to targeted attacks since they require larger perturbations and produce predictions with low confidence, while deeper layers that are closer to the output are more sensitive to attacks as they require smaller perturbations to produce incorrect predictions with very high confidence.



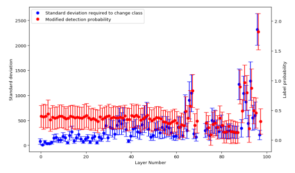
(a)



(b)



(c)



(d)

Figure 7. Results for layer-wise random perturbations. The blue points indicate the standard deviation (σ) of the noise required to fool the network (average indicated by point and variation is indicated by the error bar) and the red points indicate the average label probability (P^*) for modified detection (average indicated by point and variation is indicated by the error bar). The average is computed over all the 1000 ImageNet classes for AlexNet (a), VGG16 (b), Resnet152 (c), and InceptionV3 (d)

.

Dataset Properties and Practical Context. UREM is adapted to practical context, where the effects of the data properties on uncertainty for targeted perturbations are evaluated on the following variants of ImageNet and COCO: **Subclass**: with reduced number of classes (10% of original number of classes) and **Subsample**: with reduced number of training samples (50% of all original samples per class).

The four networks trained and evaluated include: AlexNet, VGG16, ResNet152, and InceptionV3 for the two variations and performed the robustness analysis with random and targeted perturbations. The accuracy of the original networks and their sub-sample and sub-class variants is summarized in Table 2.

The previous results suggest that accuracy drops when using fewer training samples and improves when learning to distinguish fewer classes. The robustness measure based on random perturbations for the different network architectures and their data variants is summarized in Table 3

NETWORK ARCHITECTURE: BASELINE ACCURACY VS. DATA

Data Type	AlexNet	VGG 16	ResNet 152	Inception V3
Full	72	83.4	91	92.2
Subsample	46.2	79.1	88.6	89
Subclass	77	92	97	99

Table 2. Comparison of classification accuracy of neural network architectures as a function of dataset type. The types include original: full dataset; subsample: 50% of samples per class using all classes; and subclass: using all samples from 10% of the classes.

NETWORK ARCHITECTURE: RANDOM ROBUSTNESS VS. DATA

Data Type	AlexNet	VGG 16	ResNet 152	Inception V3
Full	66.3	79.8	136.4	196.6
Subsample	10.3	16.9	31.7	36.8
Subclass	23.3	27.5	57.2	52.2

Table 3. Comparison of random robustness of neural network architectures as a function of dataset type. The types include original: full dataset; sub-sample: 50% of samples per class using all classes; and subclass: using all samples from 10% of the classes.

These practical context experiments and results demonstrate that reducing the number of samples and number of classes makes easier to find perturbations to change the classifier output. A key takeaway is that robustness to random perturbations can be improved by using larger training sets. The robustness measure based on targeted perturbations for the different network architectures and their variants is shown in perturbation matrix Figures 8, 9 and 10.

The impact of subsample and subclass on robustness to targeted perturbation is similar to that of random perturbations. The results indicate that using more training samples, while training to differentiate between larger number of classes improves robustness to perturbations. Note that there are a few yellow lines in these perturbation matrices that indicate that failure perturb with specified target class probability thresholds, even after a large number of iterations. These represent some one off classes that tend to be more robust than others.

5. Discussion.

The investigation and experiments from this work are intended to help us to better understand some of the limitations by exploring the unreliabilty and failure cases of DNN-based object detectors. It is important to notice the perturbation to change labels varies for different applications and scenarios and that the correct interpretation of "perturbations" challenging and essential. For instance, the robustness of VGG16 to targeted attacked is worse than AlexNet as the data goes through more gain parameters, which is counter intuitive given that VGG16 is reportedly more accurate. The expectation of this work is: (1) to un-

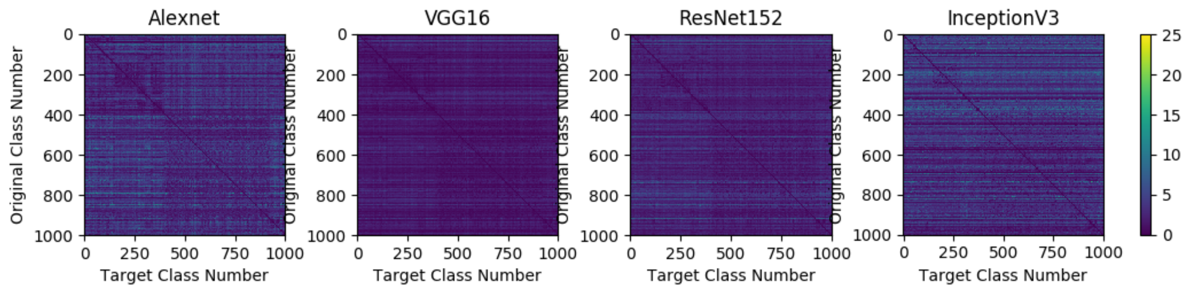


Figure 8. Perturbation matrices for pre-trained Alexnet (first), VGG16 (second), ResNet152 (third), and InceptionV3 (fourth) models showing the max difference ($MAX(|I - I^{*}|)$) of the perturbation required to change a sample from one class to all other classes, where lighter cells indicate high robustness.

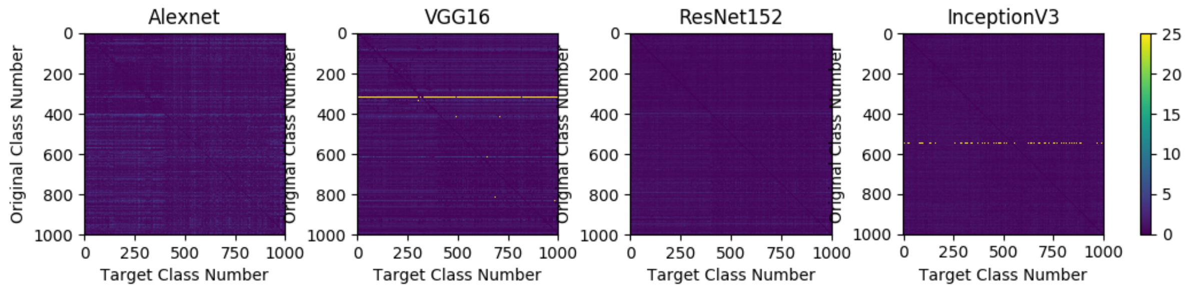


Figure 9. Perturbation matrices for **subsample** Alexnet (first), VGG16 (second), ResNet152 (third), and InceptionV3 (fourth) models showing the max difference ($MAX(|I - I^{*}|)$) of the perturbation required to change a sample from one class to all other classes, where lighter cells indicate high robustness.

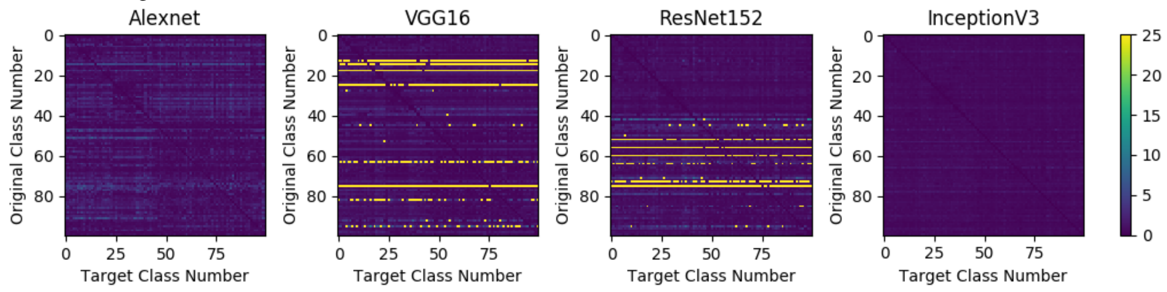


Figure 10. Perturbation matrices for **subclass** Alexnet (first), VGG16 (second), ResNet152 (third), and InceptionV3 (fourth) models showing the max difference ($MAX(|I - I^{*}|)$) of the perturbation required to change a sample from one class to all other classes, where lighter cells indicate high robustness.

cover the properties (e.g., accuracy, resilience, and robustness) of existing DNNs based on objective experimentation and (2) to help researchers and engineers design accurate and robust networks by providing evaluation and quantification methodologies for uncertainty and robustness.

6. Summary and Potential Future Directions

Summary. This paper focuses on the development of random and targeted methods to estimate the uncertainty of pre-trained DNN systems. The methods introduce new forms to assess and visualize DNN robustness to go beyond standard performance metrics. The experimental results indicate that deeper DNNs are more accurate and surprisingly more robust to random attacks; however, susceptibility to targeted attacks is decided by the number of parameters in the network, lesser parameters lead to more robustness and more parameters lead to lower robustness.

Potential Future Work and Directions. Short-term plans involve expanding the developed approaches to estimate different types of uncertainties more robustly. Long-term plans include exploring methods for finding generalizable solutions via manifolds to support system generalizability in a more reliable manner. Future work will look into methods and implementation details regarding bubble space analysis via manifold for large and small bubble spaces. Overall, future work revolves around developing approaches that combine different measures of uncertainty to improve the performance and robustness of DNNs.

References

[1] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 2018.

[2] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[3] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. corr (2015).

[5] A. Graves. Practical variational inference for neural networks. In *In Advances in neural information processing systems*, pages 2348–2356, 2011.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] C. S. Kenney, A. J. Laub, and M. Reese. Statistical condition estimation for linear systems. *SIAM Journal on Scientific Computing*, 19(2):566–583, 1998.

[8] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[12] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[13] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[14] S. Narayanaswamy, T. B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

[15] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[16] J. M. Paisley J, Blei D. Variational bayesian inference with stochastic search. In *arXiv preprint arXiv:1206.6430*, pages 2348–2356, 2012.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] S. Singla and S. Feizi. Robustness certificates against adversarial examples for relu networks. *arXiv preprint arXiv:1902.01235*, 2019.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[22] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[23] D. Tran, A. Kucukelbir, A. B. Dieng, M. Rudolph, D. Liang, and D. M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

[24] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

[25] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 2019.

[26] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pages 4939–4948, 2018.