SCIENCE AND TECHNOLOGY ORGANIZATION

INFORMATION SYSTEMS TECHNOLOGY PANEL

IST-RSY-200

INTERNATIONAL CONFERENCE ON MILITARY COMMUNICATION AND INFORMATION SYSTEMS (ICMCIS)
Skopje, North Macedonia 16-17 May 2023

Evaluation of Robustness Metrics for Defense of Machine Learning Systems*

*This paper was originally presented in Skopje, North Macedonia, 16-17 May 2023

J. DeMarchi, R. Rijken
Royal Netherlands Aerospace Centre NLR
Collaborative Engineering Systems &
Aerospace Systems Information Supremacy
Amsterdam – NLD
julian.demarchi@nlr.nl
roel.rijken@nlr.nl

J. Melrose, B. Madahar
Defence Science and Technology Laboratory
Cyber & Information Systems Division
Portondown – GBR
jmelrose@dstl.gov.uk
bkmadahar@dstl.gov.uk

G. Fumera
University of Cagliari Department of
Electrical and Electronic Engineering
Cagliari – ITA
fumera@unica.it

F. Roli
University of Genoa Department of
Informatics, Bioengineering, Robotics and
Systems Engineering
Genoa – ITA
fabio.roli@unige.it

E. Ledda
Sapienza University of Rome Department of
Computer, Control and Management
Engineering
Rome – ITA
emanuele.ledda@uniroma1.it

M. Aktaş
ASELSAN
Defence Systems Technologies Division
Ankara – TUR
maktas@aselsan.com.tr

F. Kurth, P. Baggenstoss
Fraunhofer Institute for Communication,
Information Processing and Ergonomics
Bonn – DEU
frank.kurth@fkie.fraunhofer.de
paul.baggenstoss@fkie.fraunhofer.de

B. Pelzer, L. Kanestad
Swedish Defence Research Agency Cyber
Defence and C2 Technology Division
Stockholm – SWE
bjorn.pelzer@foi.se
linus.kanestad@foi.se

2023 International Conference on Military Communications and Information Systems (ICMCIS) | 979-8-3503-4385-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMCIS59922.2023.10253593

***Abstract*—In this paper we explore some of the potential applications of robustness criteria for machine learning (ML) systems by way of tangible "demonstrator" scenarios. In each demonstrator, ML robustness metrics are applied to real-world scenarios with military relevance, indicating how they might be used to help detect and handle possible adversarial attacks on ML systems. We conclude by sketching promising future avenues of research in order to: (1) help establish useful verification methodologies to facilitate ML robustness compliance assessment; (2) support development of ML accountability mechanisms; and (3) reliably detect, repel, and mitigate adversarial attack.**

Keywords—ML robustness metrics, neural networks

I. Introduction

The STO IST-169 RTG is tasked with exploring robustness and accountability in machine learning systems. Our starting point was to "*Determine the state-of-the-art in robustness and accountability for machine learning (ML) systems. Especially deep learning systems with complex and large models which are virtually impossible to manage by humans.*"

The military relevance of that is evident within the context of automated data fusion and decision support systems:

"*In order to achieve trust in military systems using complex machine learning models and algorithms, the military needs to be able to prove both robustness and accountability. Robustness is important for the availability and integrity of any military system, with or without both sensors and effectors. Accountability is likely a future requirement for such systems, and the more complex a system becomes, the documentation of accountability will grow towards "non-human" complexity.*"

The IST-169 views ML robustness as a fundamental cornerstone enabling the availability, integrity, and ultimately accountability of AI-assisted military defense support systems. In 2021, RTG members compiled a literature survey of relevant ML robustness metrics and techniques, resulting in the conclusion that "*the ongoing question is how to determine and find the right kinds of metrics for the specific models to obtain the required level of confidence in hybrid warfare systems. IST-169 intends to progress this initial survey to do just that. We believe that developing a pictorial representation of the various types of robustness, with their applicable phases*

to the different types of AI, would benefit a holistic understanding of the AI robustness landscape. This would enhance the move toward a more rigorous approach to the development and use of AI applications." [1]

That previous survey highlighted various metrics affecting ML robustness "in the face of" several different kinds of widely recognized threats [2]. We revisit some of those threats in this paper, and explore the application of metrics that can help deal with some inherent vulnerabilities of ML-based decision support systems. We also include examples on uncertainty and implausibility to illustrate that the topic of ML robustness is actually far broader than robustness to adversarial action alone.

In sections II & III we present a selection of robustness challenges identified in our abovementioned initial survey. Then we explore some of the potential applications using concrete examples from RTG members' own cutting-edge research in the field, in the form of several "demonstrators". These help paint a picture of the landscape of ML robustness metrics as they are applied to real-world scenarios with military relevance, and how they can be used to help handle possible adversarial attacks on ML in operational context, as detailed in sections IV, V, and VI.

We conclude by sketching promising future avenues of research that further build upon these "demonstrators", in order to: (1) propose useful verification methodologies to facilitate ML robustness compliance assessment; (2) support the development of ML accountability mechanisms; and (3) reliably detect, repel, and mitigate adversarial attack.

II. Robustness in the face of Uncertainty

The capability of dealing with uncertainty in training and operational data is one of the main requirements for robust machine learning systems, in particular for critical application scenarios like those in the military domain [1]. Model uncertainty can result from measurement errors, for example due to hardware limitations of a sensor, or due to different types of channel distortions between a source/object and the sensor. Typically we want to distinguish between admissible physical changes (such as noise, cropping, etc.) and semantical changes (usually introduced by humans) that can be made to an object without changing its class label. An ML system should be robust to both types of changes mentioned, and able to distinguish them from each other in order to introduce appropriate mechanisms for handling them.

A more precise term for "model uncertainty" could be "uncertainty in the model's predictions", which is mainly caused by two sources [3]:

- For classification problems, the probability distributions of different classes may overlap in some regions of the feature (input) space, which makes the class label inherently uncertain (different instances may have similar or even identical features, but different class labels), and so too the classifier output for the corresponding inputs. This is called "aleatoric" uncertainty. There is a similar mechanism for regression problems.

- When training data are limited, it is difficult to find the "correct" model, and its optimal parameter values. This leads to what is called "epistemic" uncertainty. In particular, this kind of uncertainty is higher in regions of the feature space where few or no training samples are available (that is, uncertainty on classifier predictions is higher on inputs from these regions).

In summary, two kinds of uncertainty are identified by recent machine learning literature: aleatoric uncertainty (related to intrinsic randomness in the considered task) and epistemic uncertainty, that accounts for both model uncertainty (on the correctness of the model and of the hypothesis) and approximation uncertainty (on the optimal model parameters) [4]. The latter can in principle be reduced by the availability of additional, suitable training data.

A. Dropout injection

Dropout injection focuses on epistemic uncertainty. One way to represent and evaluate it in deep neural networks (DNNs) is through Bayesian extensions [5] using, for instance, Monte Carlo dropout [6], which was originally devised as a stochastic regularization technique [7]. In particular, whereas Monte Carlo dropout is usually applied during training, and then dropout layers are kept active in the testing phase for the sole purpose of uncertainty estimation, it can be also applied at test time only [8], which we call "injected dropout" [9]. This enables uncertainty estimation also for DNNs that had been already trained without dropout, avoiding an expensive re-training process. However, our ongoing work shows that, beside a proper setting of the dropout rate (e.g., using a validation set), the effectiveness of injected dropout strongly relies on a proper rescaling of the corresponding uncertainty measure [9].

B. Application to crowd counting

Here we show an example of a practical application of injected Monte Carlo dropout for the purpose of quantifying epistemic uncertainty in a challenging computer vision task, i.e., crowd counting and density estimation from images or videos [10]. This task consists of estimating, possibly in real-time, the number of people and the corresponding density map from an input image or frame. State-of-the-art crowd counting methods are based on ad hoc DNN architectures that are trained to estimate the density map of the input image, from which the people count is easily derived by summing up the density value of each pixel. Fig. 1 (on the next page) shows an example of our injected dropout method for post hoc uncertainty estimation on a crowd counting and density estimation task. On the left, a video frame from the benchmark PETS data set [11] with the ground truth count is shown (top left), together with the estimated density map (lighter colors correspond to higher values) produced by the MCNN crowd counting model [12] with the corresponding estimated count (bottom left).

Such methods can be extended to similar counting tasks, i.e. counting different kinds of objects, not just people. This kind of functionality can be very useful to support human operators in monitoring activities in security scenarios (e.g. monitoring a mass gathering or a sensitive area through a video surveillance system).

We implemented injected dropout on a previously trained, state-of-the-art DNN architecture for crowd counting, the Multi-Column Neural Network (MCNN) [10] [12]. Dropout injection enables MCNNs to provide a post hoc, frame-by-frame estimate of the pixel-level uncertainty of the density map (i.e. the variance of the estimated density), and a confidence interval (e.g. a 90% interval) on the corresponding people count, which is defined as the pixel-wise sum of the

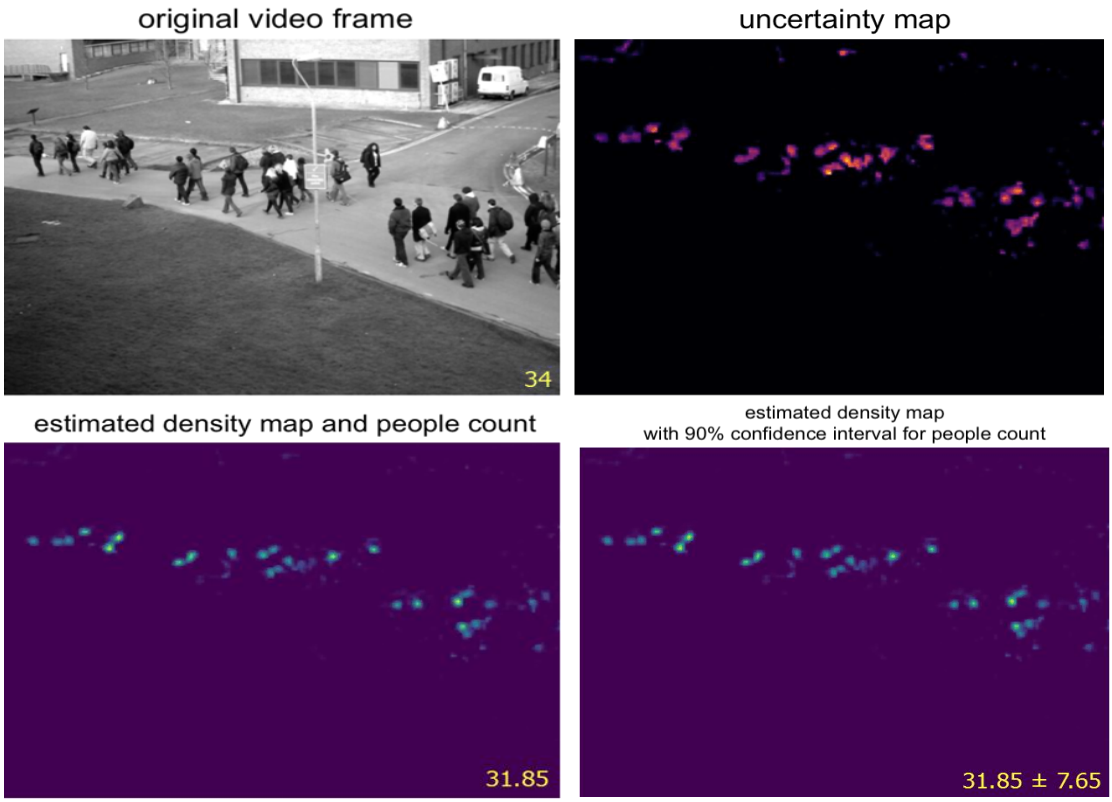

FIGURE 1: Detection of false positive localizations by combining density and uncertainty maps.

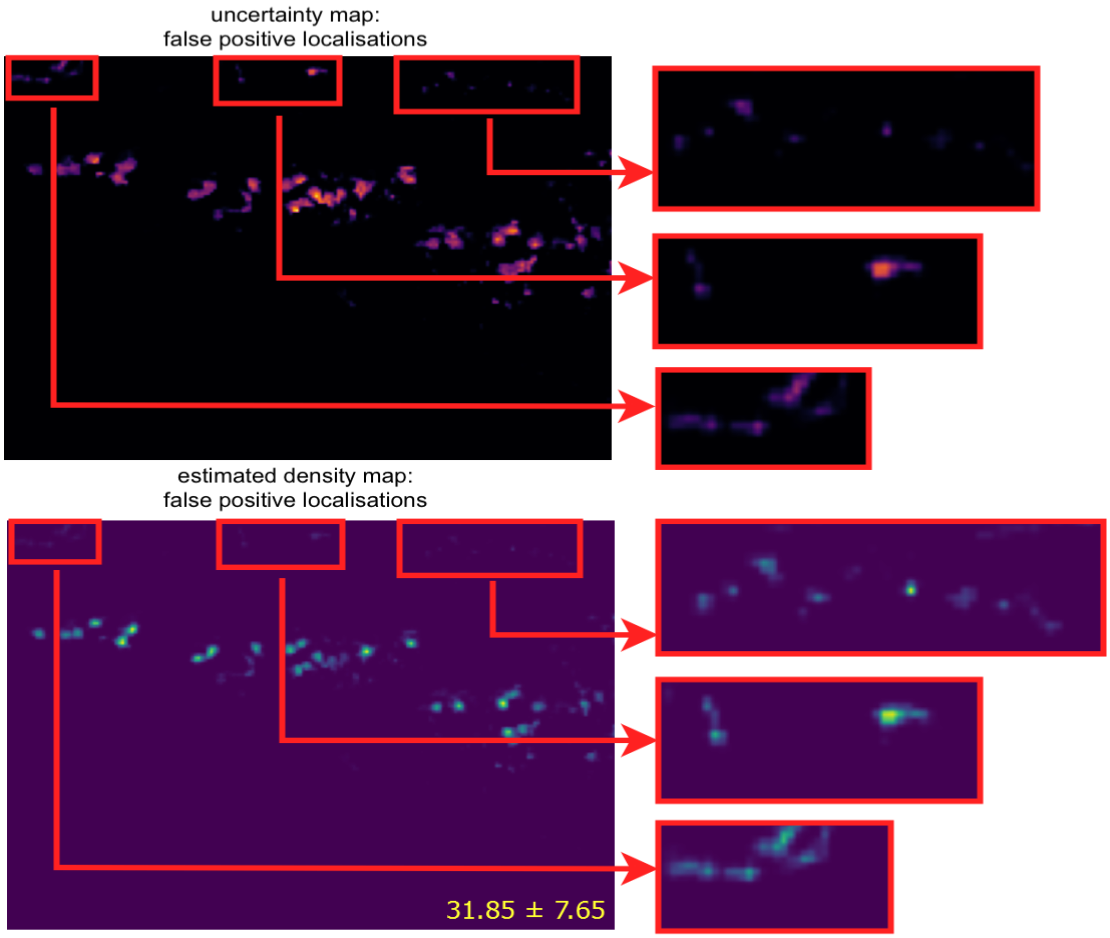

FIGURE 2: Detection of false positive localizations by combining density and uncertainty maps.

uncertainty values with respect to the bare point estimates. Fig. 1 (top right) represents the uncertainty map computed by injected dropout, Fig. 1 (bottom left) shows the same density map, and Fig. 1 (bottom right) with the corresponding 90% confidence interval on the estimated people count.

For end users, the main benefit of uncertainty estimation in this kind of task is that it can make them aware of the reliability of DNN predictions, allowing them to take more informed decisions, as well as increasing their trust in this kind of machine learning system.

C. Way forward

In our ongoing work [9] we observe that uncertainty evaluation could allow automatic detection and correction of inaccuracies in the estimated density map and in the derived count that result from "false positive" people localizations, thus improving system robustness. An example is shown in Fig. 2, where some false positive localizations related to the same video frame of Fig. 1 are highlighted in red: these are image regions with non-zero predicted density values (Fig. 2, bottom) although no people appear in them (see the video frame on the top left of Fig. 1). Such regions turn out to be characterized by relatively low density values with respect to correct localizations (they have been rescaled in the zoomed-in views of Fig. 2 to make them visible), and by a high uncertainty (Fig. 2, top). This pattern of low density and high uncertainty may be exploited to automatically detect and filter out false positive localizations, e.g. by setting to zero the corresponding density as well as uncertainty values. This would improve the accuracy of the density map and of the corresponding people count, as well as of the uncertainty estimate, by reducing the width of the confidence interval, thus improving robustness.

From a military perspective, enhancing this kind of computer vision technique with a post hoc uncertainty estimation functionality and the corresponding robustness improvement capability could be very useful, e.g. for the analysis of aerial imagery for intelligence gathering.

III. Robustness in the face of Implausibility

A. Suppression of data extraction from language models

In recent years, language models (LM) have led to a paradigm shift in natural language processing. LM are deep learning models, usually transformer-based [13], trained on very large text corpora (typically several billions of words). An LM represents the probability distributions over the word sequences of its training language – a rudimentary "understanding" of language. This allows an LM to predict the next words for a given input sequence, a functionality that can be utilized in many ways. For instance, since 2018, the GPT-series language models [14] have received significant media attention for their ability to generate long and plausible texts in different styles, and Google BERT [15] has set new state-of-the-art results in numerous natural language processing tasks. LM could be used in defense contexts, e.g. for text summarization, or to help with querying information systems. However, training an LM from scratch can take months depending on the hardware used. A common workaround is therefore to take a public LM and to fine-tune it with additional training texts from a specific domain, thereby adapting the model with its large general language knowledge to the task of interest using a domain-specific language.

While an LM is not intended to explicitly store the texts it has been trained on, it may nevertheless effectively do so. This can potentially be exploited by an adversary: With suitable prompts an LM may recreate such memorized training texts and thus reveal information not intended for the public.

To demonstrate this we followed the general approach of [16], but applied to a modern LM. First we fine-tuned the language model GPT-2 [17] on the CC-News [18] collection of news articles. We then attempted to recreate original texts by letting the model generate texts based on the first two words of articles. This succeeded in 21% of the cases (on a model fine-tuned over 40,000 training steps), representing a worst-case susceptibility of the model.

B. The perplexity metric

Generated texts are generally coherent and seemingly plausible, but not necessarily factual. An attacker may face a challenge when deciding whether an output text actually corresponds to original training data. An indication is the perplexity metric [19] that LMs usually provide, a measure based on the likelihood function p_θ of the LM on a given sequence of tokens $x_1, \dots x_n$:

$$perplexity(x_1, \dots, x_n) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n} \log_{10} p_\theta(x_i|x_1, x_2, \dots, x_{i-1})\right) \quad (1)$$

Effectively, this score represents how unexpected a given text is to an LM. Feeding back a generated text into the LM will produce a lower perplexity score the closer the text is to original training data. Furthermore, when the military LM is a fine-tuned version of a public model, the adversary can feed the generated text back into both models to compare their resulting perplexity scores. Fig. 3 illustrates how the vast majority of sample texts generated by the fine-tuned LM results in significantly higher perplexity on the original LM. This means that recreated original texts overwhelmingly stay below a clear perplexity threshold: In our experiment, 99.4% of such texts had a perplexity of no more than 2.5. Hence a hypothetical adversary could use such a threshold as an aid to determine which of the generated texts correspond to original training data, and thus constitute successful extractions.

This type of extraction attack is related to membership inference attacks [20], where the attacker aims to find out whether some given information was used for training. Similarly to defense against inference attacks, differential privacy [21] may provide countermeasures to such extraction.

IV. Robustness in the face of Adversarial Attack

A. Distinguishability criteria

The aim of adversarial attacks is to modify the model input to result in incorrect model output such that it cannot be distinguished by the human observer. Distinguishability criteria bear some limitations on the perturbation that can be applied to inputs, which are referred to as the L_p norm in literature, i.e.

$$\|C - A\|_p \leq \in \quad (2)$$

where C and A represent the input sample and adversarial example, respectively, and $\in$ is the maximum allowable perturbation. The most commonly used norms are L_2 and L_∞. In this context, it is assumed that there are two different

decision makers, the first one a human being capable of deciding correctly, and the second one an artificial neural network. To define the network as robust, it is required that the network can discriminate the modified input from the true input as effectively as a human being. Therefore, the most commonly used metric for robustness measurement is the "robustness accuracy", defined as the model accuracy for the inputs perturbed by the adversarial perturbations [22]. Since the input perturbation highly depends on the applied adversarial attack, it is hard to quantify the actual adversarial robustness of the model. In [23] it has been shown that deep neural networks can achieve human-level performance only when the applied manipulations are known in the training phase. For unknown manipulations, the performance of the deep neural networks decreases dramatically. Therefore, to achieve reliable results, the adversarial robustness should be reported as the model accuracy on worst-case inputs taken from perturbed sets, which requires applying multiple different types of attacks on the trained model. The works in [24] and [25] aim to benchmark several models' performance in an adversarial setting and compare their robustness levels under various adversarial attacks.

In some cases, the attacker can access the model training phase and poisoning type attacks [26] [27] [28] can be used for fooling the systems. There are two types of poisoning attacks, i.e. data poisoning and model poisoning. In data poisoning attacks, attackers influence the training data to manipulate the results of a predictive model. In model poisoning attacks, the adversarial objective is to cause the model to mis-classify a set of chosen inputs with high confidence without accessing the training data as in Federated Learning [29].

B. *Adversarial training*

The main motivation of adversarial training is to cast both attacks and defenses into a common theoretical framework, naturally encapsulating most prior work on adversarial examples [30]. In this method, instead of feeding samples from the original dataset directly into training, the adversarial attack is allowed to perturb the input first, and subsequently the perturbed examples are fed into the training set. In this context, adversarial training aims to increase the richness of the training dataset by exploring vulnerable examples and letting the model learn to correctly classify those examples. The more exploratory the vulnerable examples, the more robust the model.

The effect of such data generation methods for improving adversarial robustness is handled in [31] and it is shown that the methods that generate images closer to the test set improve robustness. Adversarial training can be augmented in various ways, such as changing the attack procedure, loss function or model architecture [32] [33]. The effects of adversarial training on robustness is analyzed in [34], concluding that during the clean training process using (stochastic) gradient descent, neural networks will accumulate, across all features, some "dense mixture directions" that have low correlations with any natural input, but are extremely vulnerable to (dense) adversarial perturbations. During adversarial training, such dense mixtures are "purified" to make the model more robust.

Some of the other mitigation methods for generating robust models are making it difficult to find the adversarial examples, i.e., gradient hiding or masking [35], feature squeezing or discretization [36], using auxiliary models in addition to the target model such as defensive distillation [37] and defensive GAN[38], and detecting and rejecting adversarial examples such as blocking the transferability [39] and calculation of the Lipschitz constant [40].

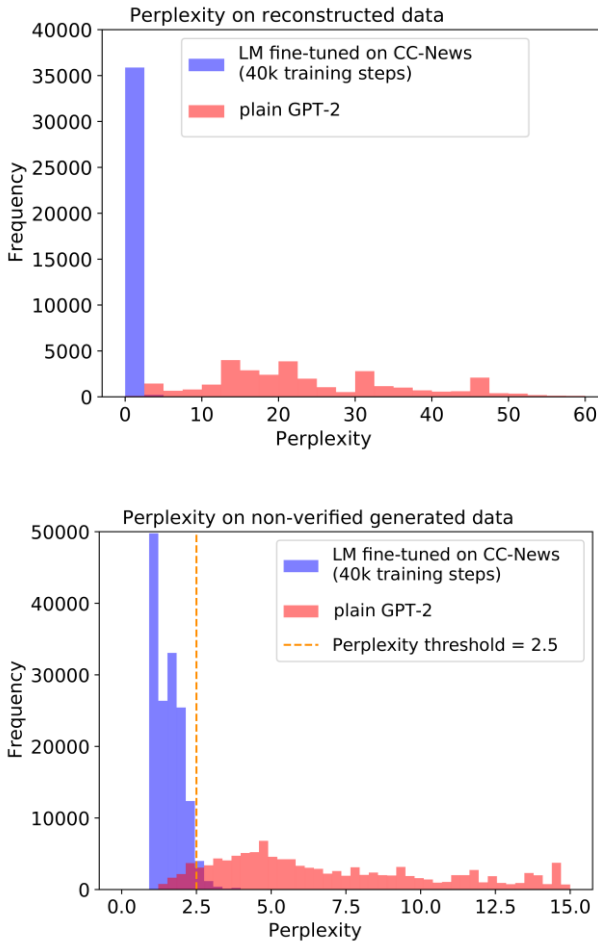

FIGURE 3: Perplexity comparison of fine-tuned and original LM with generated samples (top: samples verified to match training data; bottom: non-verified samples).

C. *Way forward*

Current training methods use limited datasets, however the model being trained must attempt to classify all the possible inputs (infinitely many) including those containing noise and/or which lie outside the distribution of the training dataset. Training methods should therefore be refined in order to make the model aware of the input characteristics. The model should be able to decide whether the given input is correlated with the task that the model is trained for. The model then becomes responsible only for the related input, thereby making the model more robust to adversarial input perturbations.

V. ROBUSTNESS USING PBNS AND DETECTION OF ADVERSARIAL ATTACKS

A. *PBN and the robustness of neural networks*

In [1] generative modelling was used to introduce robustness with respect to semantical variations of the input data. Generative models can be used to visualize how a particular classifier makes a certain class decision based on a given data sample. We now discuss Projected Belief Networks (PBNs) based on image and audio classification as examples.

B. Robustness of discriminative and generative networks

Most neural networks fall into one of two categories: (a) discriminative networks, a type of feed-forward networks in which the information flows in the forward direction, starting from high-dimensional input data (images or sounds to be classified), and ending in a low-dimensional network output, usually the class identification; and (b) generative networks, which start with a random input, then proceed in the backward direction, usually with increasing dimension, ending with a synthetic data sample, of the same size and format as the input data. Generative networks can be used to create synthetic data, but more importantly, they are trained to approximate a data generation process for the training data. Having a mathematical model for the data generation process also provides a mathematical tool, the likelihood function (LF), that allows testing samples to be graded according to how "likely" it is that they are members of a given data class.

On the question of robustness, discriminative and generative networks are widely opposed. Discriminative networks can be easily fooled into making false decisions, through the process of adversarial attacks (AA), in which an almost imperceptible change is made to an input sample, causing the network to make false classification decisions. This is possible because the network is only trained to classify between the data classes in the provided training data. As soon as something new is presented to the network (novelty), it is unable to reach a reasoned conclusion. Generative networks on the other hand, contain a model of the training data classes, so are able to reject AA samples which are unlike the training data that was provided. Unfortunately, generative networks make poor classifiers because they are highly sensitive to model mismatch, and modeling high-dimensional data is an extremely difficult task.

C. PBN: Balanced discriminative-generative robustness

The projected belief network (PBN) is a new type of network that is simultaneously a discriminative feed-forward network and a generative network with inherent model of the data [41] [42]. It accomplishes the generative task using the concept of back-projection (not to be confused with back-propagation), in which information conceptually flows backwards through the feed-forward network. Thus, the same network can be used in either direction, and can be trained to accomplish both tasks at the same time. An unavoidable result of this dual role is that the network output must extract information from the data that not only discriminates between the data classes, but also describes the data sample in detail. In fact, it has been shown that the PBN leads to information maximization [43], and results in generative classifiers that can compete with state-of-the-art discriminative classifiers [44] [45].The PBN also provides a likelihood function (LF) so that AA samples can be accurately rejected [46].

D. PBN: Mathematical basics

A PBN is based on the principle of probability density function (PDF) projection [47]. Let $\mathbf{z}=T(\mathbf{x})$ be an arbitrary dimension-reducing feature transformation taking high-dimensional input data $\mathbf{x}$, and producing a low-dimensional feature $\mathbf{z}$, with estimated or assumed feature distribution $g(\mathbf{z})$. In PDF projection, the feature distribution $g(\mathbf{z})$ is "projected" back to the input data, resulting in a unique input data distribution given by equation (3):

$$G(\mathbf{x}) = [p_0(\mathbf{x})/p_0(\mathbf{z})]\, g(\mathbf{z}) \qquad (3)$$

where $p_0(\mathbf{x})$ is a maximum entropy (MaxEnt) [48] prior distribution of $\mathbf{x}$, and $p_0(\mathbf{x})$ is the mapping of $p_0(\mathbf{x})$ to the output of the transformation $T(\mathbf{x})$. When transformation $T(\mathbf{x})$ is implemented by a neural network layer, then the result is one layer of a PBN. Multi-layer PBNs are created by cascading transformations, and applying equation (3) recursively. Consider a 2-layer PBN created by cascading the layers $\mathbf{y}=T_1(\mathbf{x})$, $\mathbf{z}=T_2(\mathbf{y})$. The resulting input data distribution is then given by equation (4):

$$G(\mathbf{x}) = [p_{0x}(\mathbf{x})/p_{0x}(\mathbf{y})]\, [p_{0y}(\mathbf{y})/p_{0y}(\mathbf{z})]\, g(\mathbf{z}) \qquad (4)$$

where $p_{0x}(\mathbf{x})$ and $p_{0y}(\mathbf{y})$ are the MaxEnt priors for $\mathbf{x}$ and $\mathbf{y}$, respectively. This idea can be extended to any number of layers. Equation (4) can be trained for maximum likelihood by maximizing the mean value of $\log(G(\mathbf{x}))$ over the training data set. At the same time, we can train the network as a classifier. Because $G(\mathbf{x})$ in Equation (4) is a probability distribution, we can draw samples of $\mathbf{x}$ using the concept of back-projection. In the 2-layer PBN represented by distribution Equation (4), back-projection proceeds as follows: (a) We first draw a feature $\mathbf{z}$ randomly from distribution $g(\mathbf{z})$. Next, we find a sample $\mathbf{y}$ randomly from the set $\{\mathbf{y}: T_2(\mathbf{y})=\mathbf{z}\}$, assuming the prior distribution $p_{0y}(\mathbf{y})$ over this set, then (b) draw a sample $\mathbf{x}$ randomly from the set $\{\mathbf{x}: T_1(\mathbf{x})=\mathbf{y}\}$, assuming the prior distribution $p_{0x}(\mathbf{x})$. We can also sample non-randomly [42] – instead of drawing randomly from the sets $\{\mathbf{y}: T_2(\mathbf{y})=\mathbf{z}\}$ and $\{\mathbf{x}: T_1(\mathbf{x})=\mathbf{y}\}$, we can select the conditional mean (the centroid of the set).

E. Illustrative experiments

To demonstrate the different behavior of discriminative and generative networks under adversarial attack, we trained a standard network to classify between the handwritten characters "3", "8", and "9". Using non-random back-projection, we reconstructed the input samples from the output of the second layer. In Fig. 4, the re-constructed samples look like noise. It is indeed disturbing to see that both rows in the figure produce the same network output, indicating how easy it is to "fool" the maximum likelihood algorithm with fake data. Next, we continued training the same network as a PBN. It is seen in Fig. 5 that the reconstruction quality is improved dramatically.

FIGURE 4: Top row: original samples of handwritten characters. Bottom row: reconstructed using non-random back-projection from second layer (64-dimensional feature).

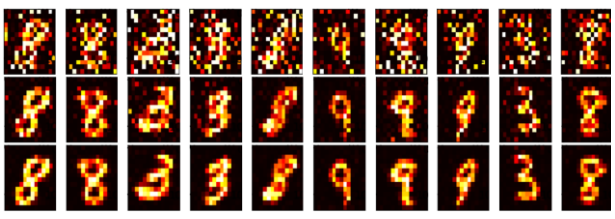

FIGURE 5: Reconstruction results from the same network in Fig. 4, when trained as a PBN in increments of 20 epochs.

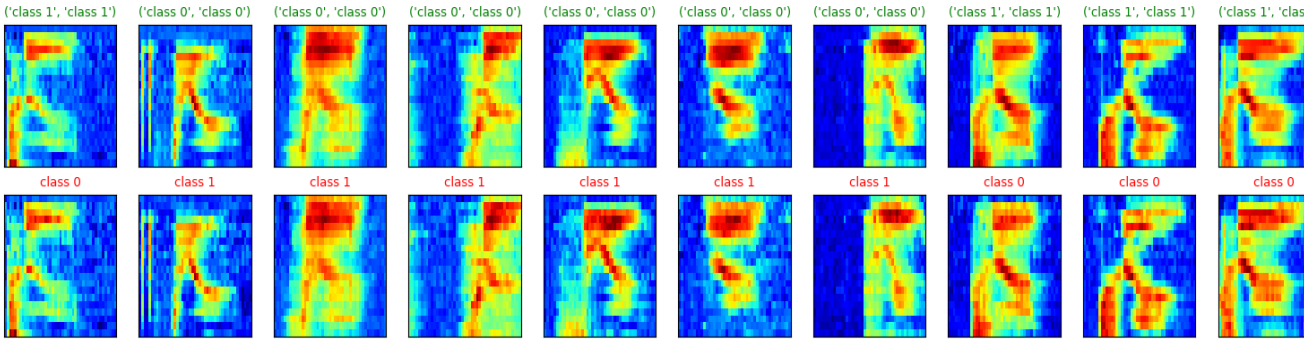

FIGURE 6: Top row: original spectrograms from Google keyword voice commands "three" vs. "tree", showing correct classification decision in all cases. Bottom row, with AA, showing false decisions in all cases.

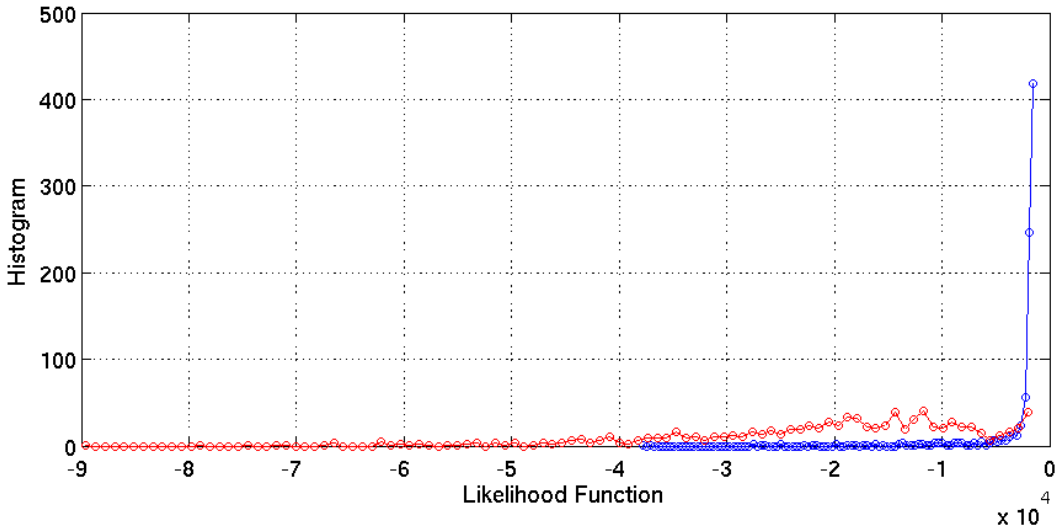

FIGURE 7: Histograms of log-likelihood values for AA samples (red) and normal samples (blue).

The above principles can be put to practical use to detect AA samples. In Fig. 6, we illustrate this for an audio classification example. The top row shows spectrograms of 10 utterances of the keywords "three" and "tree" (taken from Google voice commands). All of those are correctly classified by the network. The bottom row shows how nearly imperceptible AA leads to 10 false classifier decisions. In Fig. 7, however, we see that the histogram of the PBN log-likelihood values for AA and normal samples from the same network show almost total separation, allowing the AA samples to be reliably rejected.

F. Way forward

In future work, the capabilities of PBNs for robust classification will be investigated more rigorously. For this PBNs will have to be evaluated in realistic application scenarios. Depending on those scenarios, suitability of PBNs with respect to specific robustness requirements beyond the detection of AAs might need to be addressed.

VI. DEFENSE AGAINST ADVERSARIAL ATTACK

A. Lipschitz sensitivity analysis

Image-based malware detection is important for the military because it can help reduce the large number of malware and exploits targeted towards military networked computing. Our main contribution to this research is the study of robustness metrics to increase the resilience of military networks to cyberattack and minimize the undermining of the security and integrity of military networks. Malware image classification identifies and categorizes malicious software by analyzing their visual patterns, typically in the form of a binary file. This technique can be used to detect and defend against various types of malware, such as viruses, trojans, and worms. One method to defend against adversarial attacks is by computation and evaluation of the Lipschitz constant.

The Lipschitz constant is a measure of how much the output of a function changes when the input is perturbed by a small amount (a kind of sensitivity analysis). In the context of malware image classification, a large Lipschitz constant implicates that small changes to a binary file could lead to a significant change in the classification result. This can be used to detect and defend against adversarial attacks, where an attacker may attempt to alter a binary file such that it confuses the classification algorithm. By enforcing a small bound on the Lipschitz constant, a classification algorithm can be made more robust to such attacks. Regarding neural networks, a Lipschitz constant is a measure of how much the output of the network may change in response to a small input perturbation. A Lipschitz continuous function is limited in how much it can change and is bounded by a real number called the Lipschitz constant. We use the Lipschitz constant to analyze the sensitivity, stability, and robustness of neural networks [40].

B. Local and global Lipschitz constants

There are two types of Lipschitz constants for neural networks: the local and the global Lipschitz constant [49]. The local constant is a measure of how much the output may change locally around a specific point, whereas the global constant is a measure of how much the output of a function may vary over the entire function domain. In other words, it's a measure of how much the output may vary globally across all input points. Ascertaining the Lipschitz constant can be valuable because it provides a way to measure the robustness of a neural network to adversarial attack. It can also be used to train a robust neural network to be less susceptible to adversarial attack. Calculating the exact Lipschitz constant is often computationally hard, hence upper bounds are typically used as an alternative.

C. Generation of adversarial samples

Adversarial attacks in machine learning require small, intentionally introduced perturbations to input data in order to deceive a model's prediction. These attacks can be used to evaluate the robustness of a model and exploit vulnerabilities in real-world applications [50].

The Fast Gradient Sign Method (FGSM) is a method used for generating adversarial examples. It is a one-step method, in that it only requires one iteration of gradient descent [51] to generate an adversarial example. The basic idea behind FGSM is to take the sign of the gradient of the loss function with respect to the input data and use that to perturb the input in the direction that will most likely cause the model to misclassify. FGSM is a simple but effective method to generate adversarial examples, and has been successfully applied within numerous domains including image classification, natural language processing, and speech recognition.

D. Experiments

As an introduction into the topic of robustness metrics, we hypothesize that a Lipschitz constant can be used to create more robust networks against adversarial attacks using the FGSM method. To test this hypothesis, we implemented a deep learning network to classify malware image samples. The malware dataset we used is the Malimg dataset [52], consisting of thousands of malware binary files in image format. An example is depicted in Fig. 8, containing 25 classes of malware types, such as Allaple.L, Fakerean and Yuner.A.

As our base malware classifier we build a convolutional neural network (CNN) in Keras. We calculated the global Lipschitz upper bound of the model by transforming the Keras network into PyTorch format and computing the bound of the network using the implementation described in [53].

In addition, we implemented the same model as a DEEL-LIP network. DEEL-LIP is a library built on top of TensorFlow using standard elements of Keras to easily build a neural network that has a Lipschitz constant of 1, a so-called 1-Lipschitz network. We used this network as an example of a globally robust network to compare the Lipschitz constants and robustness against adversarial attacks with respect to the base malware classifier.

After preprocessing the data in test and training sets we trained the two networks for 10 epochs on the same data, e.g. going through the entire data set 10 times over. The global Lipschitz constant of the base model was found to be 1.46 and the Lipschitz constant of the DEEL-LIP model is 1, by design. The accuracy of the base malware classifier was trained to 96% and the accuracy of the DEEL-LIP model was trained to 93%. Both models were trained using the same number of data batches.

Finally, we randomly selected 27 malware samples and generated adversarial attack samples using the FGSM method. Fig. 9 (left) shows the adversarial attack image generated from the Yuner.A malware image. We presented the adversarial attack images to the base CNN model and the 1-lipschitz DEEL-LIP model and compared the predictions of the classifiers. Fig. 9 (center) depicts the results of the predictions. The difference is virtually imperceptible to the naked eye, and therefore highlighted in Fig. 9 (right).

FIGURE 8: Various malware class byte code instances depicted as 2D images.

FIGURE 9: Yuner.A original (left) and the generated adversarial attack image (center) – virtually indistinguishable; and the difference between the two highlighted (right).

E. Results and conclusions

The results of our experiments are summarized in Table 1. The base malware classifier classified only one instance correctly, whereas the 1-Lipschitz network classified 23 instances correctly from the 27 randomly selected malware samples. Some malware classes were selected multiple times, but it is interesting to see that the predictions of the base model varies, even if the same class of malware is presented. The DEEL-LIP model only misclassifies four (4) samples. The predictions of the DEEL-LIP model are consistent. The results show that the predictions of the 1-Lipschitz model are more accurate and stable compared to the base CNN model when presented with perturbed images, using the FGSM method. Although the accuracy of the DEEL-LIP model was initially less than the accuracy of the base model, it was found to be more robust against an FGSM adversarial attack.

F. Way forward

We investigated a method to increase the resilience of a malware classifier against an adversarial attack, by utilizing the Lipschitz constant as a metric. The metric provides a way to measure the robustness of a neural network to adversarial examples and it can also be used to train a robust neural network, which is less susceptible to adversarial attacks. We applied the metric in the context of malware image classification and showed that a significant result can be achieved to withstand an adversarial attack with the FGSM method.

Future work will investigate whether the presented method is as successful against other adversarial attacks, such as black box attacks. Furthermore, we intend to study whether application of the Lipschitz constraint to a network appreciably degrades its overall classification accuracy.

TABLE 1: Adversarial attack predictions. Base CNN versus 1-Lipschitz DEEL-LIP network

Malware Class	Base CNN model	Deel-Lip 1-Lipschitz model
Wintrim.BX	Allaple.A	Allaple.A
Allaple.A	Yuner.A	Allaple.A
Allaple.A	Allaple.L	Allaple.A
Adialer.C	Swizzor.gen!E	Adialer.C
Allaple.L	Allaple.A	Allaple.L
Lolyda.AA3	Lolyda.AA3	Lolyda.AA3
Yuner.A	Swizzor.gen!I	Yuner.A
Allaple.A	Allaple.L	Allaple.A
C2LOP.P	Swizzor.gen!E	Allaple.A
Allaple.L	Allaple.A	Allaple.A
Allaple.A	Yuner.A	Allaple.A
Lolyda.AT	VB.AT	Lolyda.AT
Allaple.A	Wintrim.BX	Allaple.A
C2LOP.P	Swizzor.gen!E	C2LOP.P
Allaple.A	Alucron.gen!J	Allaple.A
Allaple.A	Allaple.L	Allaple.A
C2LOP.gen!g	VB.AT	Allaple.A
Adialer.C	Swizzor.gen!E	Adialer.C
Adialer.C	Swizzor.gen!E	Adialer.C
Allaple.A	Yuner.A	Allaple.A
Instantaccess	Swizzor.gen!I	Instantaccess
Allaple.A	Allaple.L	Allaple.A
Allaple.A	C2LOP.gen!g	Allaple.A
Fakerean	Instantaccess	Fakerean
Allaple.A	Yuner.A	Allaple.A
Obfuscator.AD	Instantaccess	Obfuscator.AD
Lolyda.AT	VB.AT	Lolyda.AT

VII. ONGOING AND FURTHER RESEARCH

There is an increased research focus at the systems engineering and software engineering level on innovative processes, tools and techniques to assure systems that comprise Artificial Intelligence components, such as Artificial Neural Networks (ANNs), i.e. Deep Neural Networks (DNNs). In particular the robustness and resilience of such systems deemed critical (e.g. safety, security) to an application such as in aerospace, automotive and defense sectors (e.g. "uncrewed" systems for safe operations on land, sea, or air autonomous and lethal systems). Military defense presents specific challenges regarding systems and components sourced from different suppliers as well as combining and integrating them to protect against new types of adversaries.

An innovative development in software engineering and testing is termed Concolic [54]. A portmanteau, *Concolic* combines traditional *conc*rete software program execution testing [55], random testing, with symb*olic* analysis to provide better execution path coverage to uncover abnormalities (e.g. adversarial attacks). The rationale behind this is that stronger verification through better structured testing at execution levels is needed for many critical applications where conformity between high level code and executable code cannot be ensured (e.g. due to no access to source code, compiler bugs and artefacts, etc.). Either concrete testing or symbolic analysis on their own are impractical for DNNs, requiring a large number of input variables for the former and too many DNN neuron activation paths for the latter. Concolic testing can mitigate the complexity by directing the symbolic analysis to particular execution paths, through concretely evaluating given properties of the DNN.

An example process, given in [54] [56] [57], is to cover broad test requirements using Quantified Linear Arithmetic over Rationals (QLAR – essentially a first-order logic with quantifiers formal method) to express them. For a given set R of test requirements, gradually generate test cases to improve coverage by alternating between concrete execution and symbolic analysis. Given an unsatisfied test requirement r transform it to $\mu(r)$ by means of a heuristic function μ. Then, for the current set T of test cases, find pair $(t, \mu(r))$ close to satisfying r according to an evaluation based on concrete execution. Then use symbolic analysis on this pair to obtain a new concrete test case t', and add to existing test suite to form T'. Repeat the process until a satisfactory level of coverage is achieved. The generated test suite can then be used for analysis (robustness oracle) to detect whether T includes adversarial examples (e.g. using a distant metric).

Complexity, coverage criteria to direct the production of test cases, constraint solvers, and heuristics present difficulties. Much of Concolic and related research [56] [57] [58] [59] [60] is improving these through mathematical techniques and ensuring that the DNNs are amenable to the processing. For example, as mentioned in earlier sections, one such requirement is for Lipschitz continuity which is expected to hold for a large class of DNNs. The properties must be

semantic, with specific relation to DNNs, and to robustness of DNNs and related Generative Adversarial Networks (GANs). A small value for the associated Lipschitz constant, for the whole input space, or found sub-space, significantly improves the performance, which is otherwise difficult or intractable. New test criteria for neuron coverage paths include boundary cover for activation values exceeding pre-set bounds, such as test cases used in Modified Condition/Decision Cover (MC/DC) methods for software testing.

A DNN testing and debugging tool called DeepConcolic has been developed [57], and is openly available for, and in use by. the research community [61]. Its architecture is shown in Fig. 10 and has produced encouraging results using datasets with adversarial examples. Examples of performance versus random testing (traditional approach) is shown in Fig. 11. DeepConcolic covers a large range of Lipschitz constants and thus produces a good robustness indicator for images (e.g. against perturbations) that random testing coverage would have missed.

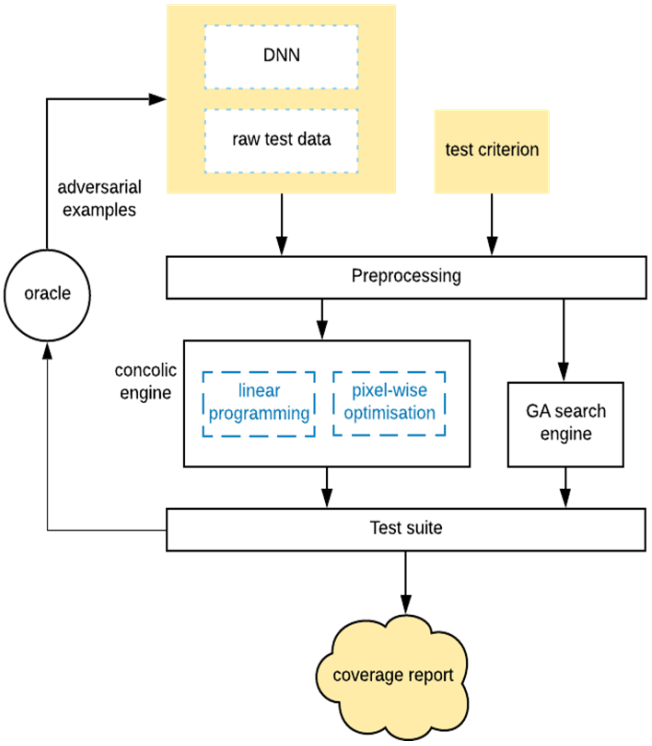

FIGURE 10: DeepConcolic tool architecture [57]

VIII. CONCLUSIONS

By comparing our findings for robustness methods across a broad range of neural network applications we show how ML robustness metrics play an important role in critical applications relevant to military operational contexts.

In spite of very positive findings, more work is required to further improve using additional experimental analyses, various datasets, and adversarial samples. This includes combining with other approaches outlined in this paper to offer hybrid system solutions able to balance performance, risk, and impact. Recommended future avenues of research include identifying baseline robustness tests that can be applied to ML deployed in military settings, and developing ways to detect, repel, and mitigate adversarial attacks on ML.

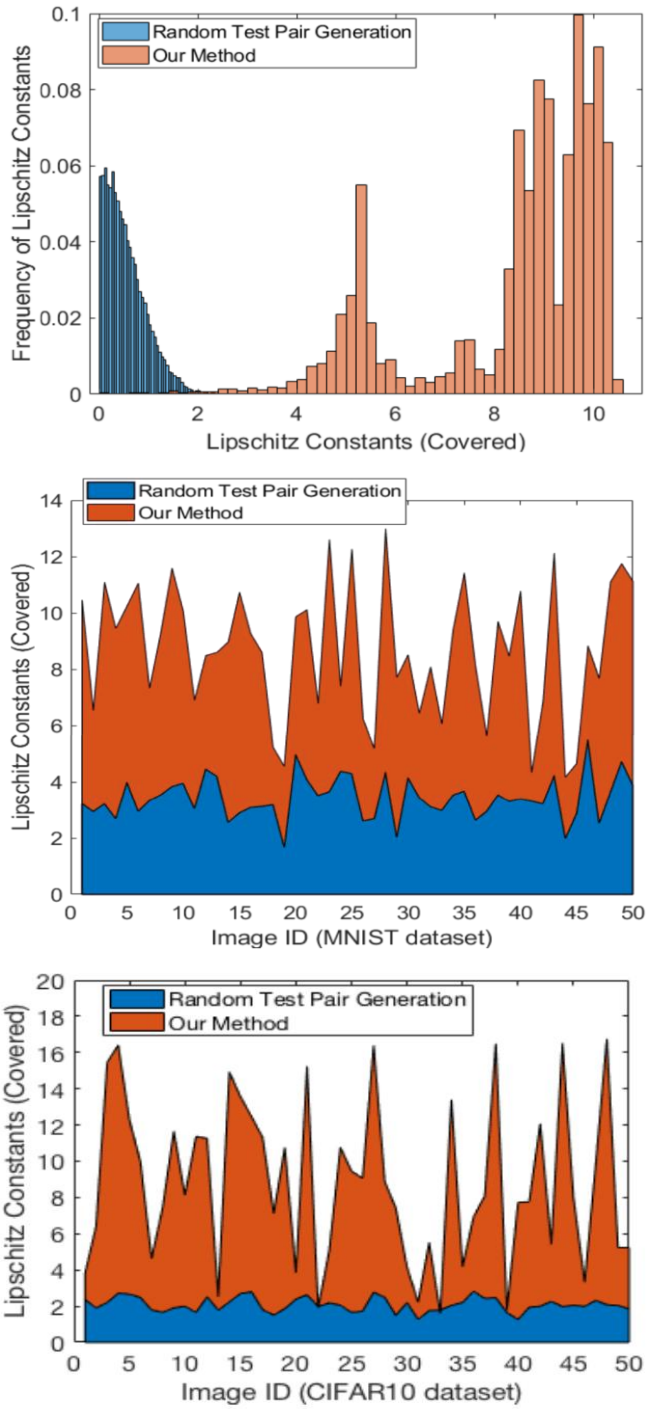

FIGURE 11: Experimental results comparing Lipschitz Constant Coverage (LCC) between Concolic testing and random testing using MNIST, CIFAR Image Data and DNNs. 1M random test pairs for MNIST Image-1 (top); 50 input images from MNIST (center); 50 input images from CIFAR-10 (bottom). [54]

IX. ACKNOWLEDGMENTS

The work in Section II was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

DSTL authors acknowledge the support of the UK MOD/DSTL Autonomy Programme, Autonomy Validation and Verification Project to their research contributions. Their content includes material subject to Crown copyright (2023), DSTL. This material is licensed under the terms of the Open Government Licence [62] except where otherwise stated.

REFERENCES

[1] J. Sharp, J. Melrose, B. Madahar, M. Aktas, N. Martinel, J. DeMarchi, E. Solberg, D. S. Lange, G. O. Tanik, F. Kurth and L. Luotsinen, "Robustness of artificial intelligence for hybrid warfare", in *STO RTG-190 Research Symposium (RSY) on AI, ML and BD for Hybrid Military Operations (AI4HMO)*, 2021.

[2] AIVD: Netherlands General Intelligence and Security Service, "AI systems: Develop them securely", 2023. https://english.aivd.nl/publications/publications/2023/02/15/ai-systems-develop-them-securely.

[3] E. Hüllermeier and W. Wägeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods", *Machine Learning,* vol. 110, pp. 457-506, 2021.

[4] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?", *Advances in Neural Information Processing Systems,* vol. 30, pp. 5574-5584, 2017.

[5] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks", *Neural Computation,* vol. 4, no. 3, pp. 448-472, 1992.

[6] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning", in *33rd Int. Conf. on Machine Learning*, 2016.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *J. Machine Learning Res.,* vol. 15, no. 56, pp. 1929-1958, 2014.

[8] A. Loquercio, M. Segu and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning", *IEEE Robotics and Automation Letters,* vol. 5, pp. 3153-3160, 2020.

[9] E. Ledda, G. Fumera and F. Roli, "Dropout injection at test time for post-hoc uncertainty quantification in neural networks", *Information Sciences,* 2023.

[10] M. A. Khan, H. Menouar and R. Hamila, "Revisiting crowd counting: State-of-the-art, trends, and future perspectives", *Image and Vision Computing,* vol. 129, no. 104597, 2023.

[11] J. Ferryman and A. Shahrokni, "PETS2009: Dataset and challenge", in *12th IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, 2009.

[12] Y. Zhang, D. Zhou, S. Chen, S. Gao and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network", in *IEEE Conf, on Computer Vision and Pattern Recognition*, 2016.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need", in *Advances in Neural Information Processing Systems*, 2017.

[14] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, "Improving language understanding by generative pre-training", *OpenAI,* 2018.

[15] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", 2018. arXiv:1810.04805.

[16] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks", in *28th USENIX Security Symposium*, 2019.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language models are unsupervised multitask learners", *OpenAI,* 2019.

[18] J. MacKenzie, R. Benham, M. Petri, J. R. Trippas, J. S. Culpepper and A. Moffat, "CC-News-En: A large English news corpus", in *29th ACM Int. Conf. on Inf. & Knowledge Mgmt.*, New York, NY, 2020.

[19] F. Jelenik, R. L. Mercer, L. R. Bahl and J. K. Baker, "Perplexity: A measure of the difficulty of speech recognition tasks", *J. Acoust. Soc. of America,* 1977.

[20] S. Reza, M. Stronati, C. S. Song and V. Shmatikov, "Membership inference attacks against machine learning models", in *IEEE Symposium on Security and Privacy*, 2017.

[21] C. Dwork, "Differential Privacy: A Survey of Results", in *Intl. Conf. on Theory and Applications of Models of Computation*, 2008.

[22] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry and A. Kurakin, "On evaluating adversarial robustness". arXiv:1902.06705v2.

[23] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge and F. A. Wichmann, "Generalisation in humane and deep neural networks", in *32nd Conf. on Neural Inf. Proc. Sys.*, Montréal, Canada, 2018.

[24] F. Croce, M. Andriushchenko, V. Sehwag, N. Flammarion, M. Chaing, P. Mittal and M. Hein, "RobustBench: A standardized adversarial robustness benchmark". arXiv:2010.09670v3.

[25] N. Papernot, F. Faghti, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, R. Aurko, A. Matyasko, V. Behzadan, K. Hambardzumvan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, R. Long and P. McDaniel, "Technical report on the CleverHans v2.1.0 Adversarial Examples Library". arXiv:1610.00768v6.

[26] M. Jagielski, A. Oprea, B. B, C. Liu, C. Nita-Rotaru and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning", *IEEE Security and Privacy,* 2018.

[27] X. Chen, C. Liu, B. Li, K. Lu and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning", 2017. arXiv: 1712.05526v1.

[28] A. N. Bhagoji, S. Chakraborty, P. Mittal and S. Calo, "Analyzing federated learning through an adversarial lens", 2019. arXiv:1811.12470v4.

[29] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. Arcas, "Communication-efficient learning of

deep networks from decentralized data", in *20th Int. Conf. on Artificial Intelligence and Statistics*, 2017.

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, "Towards deep learning models resistant to adversarial attacks". arXiv:1706.06083v4.

[31] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles and T. Mann, "Fixing data augmentation to improve adversarial robustness". arXiv:2103.01946.

[32] M. Guo, Y. Yang, R. Xu, Z. Liu and D. Lin, "When NAS meets robustness: In search of robust architectures against adversarial attacks". arXiv:1911.10695v3.

[33] S. Gowal, C. Qin, J. Uesato, T. Mann and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples". arXiv:2010.03593v3.

[34] Z. Allen-Zhu and Y. Li, "Feature purification: How adversarial training performs robust deep learning". arXiv:2005.10190v2.

[35] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay and D. Mukhopadhyay, "Adversarial attacks and defences: A survey", 2018. arXiv:1810.00069v1.

[36] P. Panda, I. Chakraborty and K. Roy, "Discritization based solutions for secure machine learning against adversarial attacks", IEEE Access 7, 2019.

[37] N. Papernot, P. D. McDaniel, X. Wu, S. Jha and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks", in *IEEE Symposium on Security and Privacy*, 2016.

[38] M. Kabkab, P. Samangouei and R. Chellappa, "Defensive-GAN: Protecting classifiers against adversarial attacks using generative models", 2018. arXiv:1805.06605.

[39] H. Hosseini, Y. Chen, S. Kannan, B. Zhang and R. Poovendran, "Blocking transferrability of adversarial examples in black-box learning systems", 2017. arXiv:1703.04318.

[40] P. Pauli, A. Koch, J. Berberich, P. Kohler and F. Allgower, "Training robust neural networks using Lipschitz bounds", *IEEE Control Systems Letters,* vol. 6, pp. 121-126, 2022.

[41] P. M. Baggenstoss, "On the duality between belief networks and feed-forward neural networks", *IEEE Trans. Neural Networks and Learning Systems,* vol. 30, no. 1, pp. 190-200, 2019.

[42] P. M. Baggenstoss, "A neural network based on first principles", Barcelona, Catalonia, Spain, 2020.

[43] P. M. Baggenstoss and S. Kay, "Nonlinear dimension reduction by PDF estimation", *IEEE Trans. Signal Processing,* vol. 70, pp. 1493-1505, 2022.

[44] P. M. Baggenstoss and F. Kurth, "Using the projected belief network at high dimensions", Belgrade, 2022.

[45] P. M. Baggenstoss, "Discriminative alignment of projected belief networks", *IEEE Signal Processing Letters,* vol. 28, pp. 1963-1967, 2021.

[46] F. Govaers and P. Baggenstoss, "On a detection method of adversarial samples for deep neural networks", in *24th IEEE Int. Conf. on Information Fusion*, 2021.

[47] P. M. Baggenstoss, "The PDF projection theorem and the class-specific method", *IEEE Trans. Signal Processing,* vol. 51, no. 3, pp. 672-685, 2003.

[48] P. M. Baggenstoss, "Maximum entropy PDF design using feature density constraints: Applications in signal processing", *IEEE Trans. Signal Processing,* vol. 63, no. 11, pp. 2815-2825, 2015.

[49] K. Leino, Z. Wang and M. Fredrikson, "Globally-robust neural networks", 2021. arXiv.2102.08452.

[50] C. Pauling, M. Gimson, M. Qaid, A. Kida and B. Halak, "A tutorial on adversarial learning attacks and countermeasures", 2022. 10.48550/arXiv.2022.10377.

[51] L. Schwinn, R. Raab and B. Eskofier, "Towards rapid and robust adversarial training with one-step attacks", 2020. 10.48550/arXiv.2002.10097.

[52] L. Nataraj, S. Karthikeyan, G. Jacob and B. Manjunath, "Malware images: Visualization and automatic classification", 2011. https://www.dropbox.com/s/ep8qjakfwh1rzk4/malimg_dataset.zip?dl=0. [Accessed 18 July 2021].

[53] T. Avant and K. A. Morgansen, "Analytical bounds on the local Lipschitz constants of ReLU networks", 2021. arXiv:2014.14672v1.

[54] S. Youcheng, "Concolic testing for deep neural networks", 2018. arXiv:1805.00089v1.

[55] C. Kaner, "Exploratory testing", in *Annual Software Testing Conf.*, 2006.

[56] S. Youcheng, "Structural test coverage criteria for deep neural networks", *ACM Trans. Embedded Computing Sys.,* vol. 18, no. 5s, pp. 1-23, 2019.

[57] S. Youcheng, "DeepConcolic: Testing and debugging deep neural networks", in *41st IEEEE/ACM Int. Conf. on Software Eng. Companion Proc.*, 2019.

[58] S. Youcheng, "Testing deep neural networks", 2018. arXiv:1803.04792.

[59] J. Wang, "RobOT: RObustness-oriented testing for deep learning systems", in *43rd IEEE/ACM Int. Conf. on Software Eng.*, 2021.

[60] W. Ruan, "Global robustness evaluation of deep neural networks with provable guarantees for the Hamming Distance", in *Int. Joint Conf. on Artifical Intelligence Organization*, 2019.

[61] DSTL, "DeepConcolic", [Online]. https://github.com/TrustAI/DeepConcolic.

[62] psi@nationalarchives.gsi.gov.uk, The National Archives, Kew, London TW9 4DU, https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3.