



Towards Robust Deep Neural Networks

by

Bao Gia Doan

M. Sc. (Electronic and Computer Engineering),
RMIT University, 2014

Dissertation submitted for the degree of

Doctor of Philosophy

in

School of Computer Science
Faculty of Sciences, Engineering and Technology
The University of Adelaide

June 2022

Supervisors:

Associate Professor Damith Chinthana Ranasinghe,
School of Computer Science,
The University of Adelaide

Doctor Ehsan Abbasnejad,
School of Computer Science,
The University of Adelaide

Contents

Contents	iii
Abstract	ix
Statement of Originality	xi
Acknowledgements	xiii
Dissertation Conventions	xv
Notations	xvii
Abbreviations	xix
Publications	xxi
List of Figures	xxiii
List of Tables	xxix
Chapter 1. Introduction	1
1.1 Introduction	2
1.2 Challenges and Opportunities	5
1.2.1 Challenges	5
1.2.2 Opportunities	6
1.2.3 Research Questions	8
1.3 Summary of Original Contributions	8
1.4 Dissertation Structure	11
Chapter 2. Background	15
2.1 Notations	16
2.2 Machine Learning Empirical Process	16

2.3	Deep Neural Networks	17
2.4	Bayesian Learning	18
2.5	Threat Model	20
2.6	Adversarial Examples	21
2.7	Trojan Attacks	22
2.8	Evaluation Metrics	22
2.8.1	Binary Classification	23
2.8.2	Multi-class Classification	24
Chapter 3. Bayesian Adversarial Learning for Robust Malware Detectors		27
3.1	Motivation and Contribution	28
3.2	Problem Definition	31
3.3	Theoretical Basis For Feature-Space Adversarial Learning	33
3.4	Methods	36
3.4.1	Bayesian Formulation for Adversarial Learning	36
3.4.2	Adversarial Risk Bounded with Bayesian Formulation	38
3.5	Experiments	42
3.5.1	Network Architecture	43
3.5.2	Experimental Results	44
3.5.3	The Impact of the Number of Parameter Particles for Capturing the Posterior Distribution	47
3.5.4	Transferability of Robustness	47
3.5.5	Comparison with (Liu et al., 2019b) Learning Objective for Adversarial Training a BNN.	48
3.6	Benefits of Estimating the Uncertainty with BNNs for Defending Against Adversarial Malware	49
3.7	Related Work	50
3.8	Discussion and Conclusion	52
Chapter 4. Bayesian Adversarial Learning with Information Gain for Robustness		53
4.1	Motivation and Contribution	54
4.2	Method	55

4.2.1	Bayesian Formulation for Adversarial Learning	55
4.2.2	Conceptualising Information Gain for Bayesian Learning	56
4.2.3	Formulating Learning a Robust Network Using Information Gain	58
4.2.4	The Relationship between Adversarial and Observational Risk	59
4.3	Experimental Results	64
4.3.1	Hyper-Parameters	64
4.3.2	Robustness Under White-box l_∞ Attacks	65
4.3.3	Ablative Studies	67
4.3.4	Evaluating the Obfuscated Gradient Effect	68
4.3.5	Experiment with Increasing Number of EoT-PGD Steps	69
4.3.6	Transferability to Other Attacks	70
4.3.7	Conjecture Validation	70
4.3.8	Transfer Attacks Between Parameter Particles	71
4.4	Related Work	71
4.5	Discussion and Conclusion	73
Chapter 5. Input Sanitisation for Mitigating Trojan Trigger Effects		75
5.1	Motivation and Contribution	76
5.2	Methodology: An Overview	80
5.3	Februus Methodology Explained	82
5.4	Experimental Evaluations	86
5.5	Robustness Against Input Agnostic Trojan Inputs	92
5.6	Robustness Against Benign Inputs	93
5.7	Robustness Against Complex Adaptive Attacks	94
5.7.1	Advanced Backdoor Attack Variants	95
5.7.2	Attacks Targeting Trojan Removal	98
5.7.3	Attacks Targeting Image Restoration	101
5.7.4	GradCAM Evasion Attacks	102
5.8	Related Work and Discussion	103
5.8.1	Backdoor Attacks and Defences	103
5.8.2	Run-time Overhead Comparisons	105
5.8.3	Comparison with State-of-the-art Methods	106
5.8.4	Limitations	107
5.9	Conclusion	109

Chapter 6. Naturalistic Adversarial Patches as Universal Triggers	111
6.1 Motivation and Contribution	112
6.2 Overview of The Attack Approach	115
6.2.1 Attack Model	115
6.2.2 Hypothesis	116
6.2.3 An Overview of The Attack Approach	117
6.3 TnT Generator	118
6.3.1 Training The Generator	118
6.3.2 Transformation to a TnT Generator	119
6.4 Attack Experiment Settings	121
6.4.1 Detailed Information On Datasets, Model Architectures and Training Configurations	123
6.5 Evaluation of TnT Effectiveness	126
6.5.1 Attack Effectiveness on <i>Entire</i> ImageNet Validation Set	127
6.5.2 Robustness to Changes in Patch Locations	128
6.5.3 Blackbox Attack–Transferability of TnTs	129
6.5.4 Attack Effectiveness and Generalisation to Other Tasks	131
6.5.5 Can Occlusion or Network Bias Explain Attack Success?	133
6.6 Evaluation of TnT Naturalism	135
6.7 Generalisation to Adversarial Patch Attacks and Comparison with Prior Attacks	136
6.7.1 Comparing to LaVAN: <i>Smallest (Noisy) Adversarial Patch</i>	137
6.7.2 Comparing to AdvPatch: A Method to Disguise the Appearance of a Target Class in a Patch	139
6.8 Physical World Deployments	140
6.9 Attack Effectiveness Against Patch Defences	141
6.9.1 Against Probably Robust Networks	141
6.9.2 Against Empirically Robust Networks	142
6.10 Related Work	144
6.11 Discussion and Conclusion	146

Chapter 7. Conclusion **149**

 7.1 Summary 150

 7.2 Future Work 151

Bibliography **155**

Biography **173**

Abstract

Deep neural networks (DNNs) enable state-of-the-art performance for most machine learning tasks. Unfortunately, they are vulnerable to attacks, such as Trojans during training and Adversarial Examples at test time. Adversarial Examples are inputs with carefully crafted perturbations added to benign samples. In the Computer Vision domain, while the perturbations being imperceptible to humans, Adversarial Examples can successfully misguide or fool DNNs. Meanwhile, Trojan or backdoor attacks involve attackers tampering with the training process, for example, to inject poisoned training data to embed a backdoor into the network that can be activated during model deployment when the Trojan triggers (known only to the attackers) appear in the model's inputs. This dissertation investigates methods of building robust DNNs against these training-time and test-time threats.

Recognising the threat of Adversarial Examples in the malware domain, this research considers the *problem of realising a robust DNN-based malware detector against Adversarial Example attacks* by developing a Bayesian adversarial learning algorithm. In contrast to vision tasks, adversarial learning in a domain without a differentiable or invertible mapping function from the problem space (such as software code inputs) to the feature space is hard. The study proposes an alternative; performing adversarial learning in the *feature space* and proving the projection of perturbed yet, valid malware, in the problem space into the feature space will be a subset of feature-space adversarial attacks. The Bayesian approach improves benign performance, provably bounds the difference between adversarial risk and empirical risk and improves robustness against increasingly large attack budgets not employed during training.

To investigate the problem of improving the robustness of DNNs against Adversarial Examples—carefully crafted perturbation added to inputs—in the Computer Vision domain, the research considers the *problem of developing a Bayesian learning algorithm to realise a robust DNN against Adversarial Examples in the CV domain*. Accordingly, a novel Bayesian learning method is designed that conceptualises an *information gain* objective to measure and force the information learned from both benign and Adversarial Examples to be similar. This method proves that minimising this information gain objective further tightens the bound of the difference between adversarial risk and

empirical risk to move towards a basis for *a principled method of adversarially training BNNs*.

Recognising the threat from backdoor or Trojan attacks against DNNs, the research considers *the problem of finding a robust defence method that is effective against Trojan attacks*. The research explores a new idea in the domain; *sanitisation of inputs* and proposes *Februus* to neutralise highly potent and insidious Trojan attacks on DNN systems at *run-time*. In Trojan attacks, an adversary activates a backdoor crafted in a deep neural network model using a secret trigger, a *Trojan*, applied to any input to alter the model's decision to a target prediction—a target determined by and only known to the attacker. *Februus* sanitises the incoming input by *surgically removing* the potential trigger artifacts and *restoring* the input for the classification task. *Februus* enables effective Trojan mitigation by sanitising inputs with no loss of performance for sanitised inputs, trojaned or benign. This method is highly effective at defending against advanced Trojan attack variants as well as challenging, adaptive attacks where attackers have full knowledge of the defence method.

Investigating the connections between Trojan attacks and spatially constrained Adversarial Examples or so-called Adversarial Patches in the input space, the research exposes *an emerging threat*; an attack exploiting the vulnerability of a DNN to generate naturalistic adversarial patches as universal triggers. For the first time, a method based on Generative Adversarial Networks is developed to exploit a GAN's latent space to search for universal naturalistic adversarial patches. The proposed attack's advantage is its ability to exert a high level of control, enabling attackers to craft naturalistic adversarial patches that are highly effective, robust against state-of-the-art DNNs, and deployable in the physical world without needing to interfere with the model building process or risking discovery. Until now, this has only been demonstrably possible using Trojan attack methods.

Statement of Originality

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint award of this degree.

I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Signed

19 / 06 / 2022
Date

Acknowledgements

I would like to express my appreciation to my Ph.D. supervisors, Associate Professor Damith Chinthana Ranasinghe and Doctor Ehsan Abbasnejad for their exceptional guidance, generous support and endless source of inspiration. This dissertation would obviously not be possible without them.

I am sincerely grateful to be a student of A/Prof. Ranasinghe. Thank you for encouraging me to step out of my comfort zone and transforming me from an engineer to become an individual researcher, challenging and appreciating my efforts to overcome problems during my Ph.D.

I would like to thank Dr. Abbasnejad, for helping me with first lessons in Machine Learning since early days into my candidature. Thank you for encouraging me to realise the elegance of mathematically supported concepts.

My gratitude also goes to my lab mates, Hoa, for his continuing support since the first day I came to Australia; Viet, for his willing support for my field experiments; Yang, Fei, Michael, Joshua, for their multiple help during my candidature.

My appreciation also goes to the University of Adelaide, for awarding me the University of Adelaide International Scholarship. It gives me not only a chance to study in Australia, especially Adelaide, such a beautiful city, but also a chance to meet new friends. Particularly, I would like to thank all of my friends here in Adelaide, especially my Vietnamese friends under the Faculty of SET, their support is countless.

I would like to thank Mr. Joe Miller and the Edit Bureau, for helping me to edit some chapters in this thesis.

My special thanks go to my dear parents, Dung and Trang, for their love and support; to my wife, Hoa, and my son, Anh, for always being there. Their infinite support and love motivated me to complete this research.

Dissertation Conventions

The following conventions have been adopted in this dissertation:

Typesetting

This document was compiled using L^AT_EX2_ε. Texstudio 2.12.22 was used as a text editor interfaced to L^AT_EX2_ε. Inkscape 0.92.3 was used to produce schematic diagrams and other drawings.

Spelling

Australian English spelling conventions have been used, as defined in the Macquarie English Dictionary—A. Delbridge (Ed.), Macquarie Library, North Ryde, NSW, Australia, 2001.

Referencing

The Harvard reference style is used for referencing and citation in this dissertation.

System of Units

The units comply with the international system of units recommended in an Australian Standard: AS ISO 1000-1998 (Standards Australia Committee ME/71, Quantities, Units and Conversions 1998).

Notations

x	A random variable
\mathbf{x}	A vector
\mathbf{x}_{adv}	An adversarial version of \mathbf{x}
\mathbf{X}	A matrix
\mathcal{X}	A set
\mathcal{X}_{adv}	An adversarial set
$\ \mathbf{x}\ _p$	A L^p norm of \mathbf{x}
$\mathbf{A} \odot \mathbf{B}$	An element-wise (Hadamard) product of \mathbf{X} and \mathbf{Y}
$\nabla_{\mathbf{x}} y$	A gradient of y with respect to \mathbf{x}
$H(x)$	A Shannon entropy of the random variable x
$\text{KL}(P Q)$	A Kullback-Leibler divergence of P and Q (probability distributions)
$f(\mathbf{x}; \boldsymbol{\theta})$	A function of \mathbf{x} parametrised by $\boldsymbol{\theta}$; sometimes, to simplify notation, we omit the argument $\boldsymbol{\theta}$ and instead write $f(\mathbf{x})$

Abbreviations

AE	Adversarial Examples
AI	Artificial Intelligence
ASR	Attack Success Rate
BNN	Bayesian Neural Network
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
ELU	Exponential Linear Unit
GAN	Generative Adversarial Network
IG	Information Gain
KL	Kullback-Leibler
MI	Mutual Information
ML	Machine Learning
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
PGD	Projected Gradient Descent
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
SVGD	Stein Variational Gradient Descent
t-SNE	t-distributed Stochastic Neighbor Embedding
VI	Variational Inference

Publications

Journal Articles

[1] **Doan, B.G.**, Xue, M., Ma, S., Abbasnejad, E. and Ranasinghe, D.C., 2022. [TnT Attacks! Universal Naturalistic Adversarial Patches Against Deep Neural Network Systems](#). *IEEE Transactions on Information Forensics and Security*.

(Scimagojr Rank: Q1, impact factor: 7.178)

[2] Gao, Y., Kim, Y., **Doan, B.G.**, Zhang, Z., Zhang, G., Nepal, S., Ranasinghe, D. and Kim, H., 2021. [Design and evaluation of a multi-domain trojan detection method on deep neural networks](#). *IEEE Transactions on Dependable and Secure Computing*.

(Scimagojr Rank: Q1, impact factor: 7.329)

Conference Articles

[3] **Doan, B.G.**, Abbasnejad, E., Shi, J.Q. and Ranasinghe, D.C., 2022. Bayesian Learning with Information Gain Provably Bounds Risk for a Robust Adversarial Defense. In *the 39th International Conference on Machine Learning (ICML'22)*.

(CORE Rank: A*, acceptance rate: 21.9%, 1235 accepted out of 5630 submissions)

[4] **Doan, B.G.**, Abbasnejad, E. and Ranasinghe, D.C., 2020. [Februus: Input purification defense against trojan attacks on deep neural network systems](#). In *the 36th Annual Computer Security Applications Conference (ACSAC'20)*.

(CORE Rank: A, acceptance rate: 23.1%, 70 accepted out of 302 submissions)

[5] Yang, S., **Doan, B.G.**, Montague, P., DE VEL, O., Abraham, T., Camtepe, S., Ranasinghe, D.C. and Kanhere, S.S., 2022. Transferable Graph Backdoor Attack. In *the 25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID'22)*.

(CORE Rank: A, acceptance rate: 25.1%, 35 accepted out of 139 submissions)

Under-Review Articles

[6] Doan, B.G., Yang, S., Montague, P., DE VEL, O., Abraham, T., Camtepe, S., Kanhere, S.S., Abbasnejad, E. and Ranasinghe, D.C., 2022. Improving the Robustness of Malware Detectors with Bayesian Neural Networks.

List of Figures

1.1	Illustration of attacks on deep neural networks (DNNs) along the model building pipeline in the Computer Vision domain. DNN systems have been proven to be susceptible to threats from attacks at i) test time (top), ii) training time (middle). In each attack, the adversaries have the same mal-intention; to cause the DNN to make malicious decisions (bottom) .	2
1.2	Outline of the dissertation	13
2.1	An illustration of a Deep Neural Network.	17
2.2	An illustration of a Bayesian Neural Network.	19
2.3	Different techniques for sampling the posterior distribution.	19
3.1	Illustrative example of adversarial examples. The adversarial example $\mathbf{x} + \delta$ is derived from \mathbf{x} in the feature space and its projection to problem-space constraints (which is more restrictive) determined by Ω is \mathbf{z}' . The colour in the background illustrates the decision regions where red colour is for malware and green is for benign programs. The solid arrow in Feature Space represents the gradient-based attack to transform a malware \mathbf{x} to $\mathbf{x} + \delta$, projected to the problem-space constraints as \mathbf{z}' to be misdetected as a benign program.	34
3.2	The difference between conventional empirical risk and adversarial risk $ R_{adv} - R $ for the EMBER test set.	41
3.3	Performance of neural network detectors in the absence of adversarial attacks in the SOREL-20M dataset. The Receiver Operating Characteristics depict the detection ability of the models as their discrimination threshold varies. We can observe BNN models to outperform their FFNN counterparts.	45
3.4	Comparing the performance of different training methods under adversarial attacks.	46
3.5	Uncertainty estimates between malware and their adversarial counterparts based on the SOREL-20M dataset.	50

4.1	Accuracy under ℓ_∞ -EoT-PGD attack on different datasets. CIFAR-10 is trained on a VGG-16 network, and, following Adv-BNN (Liu et al., 2019b), STL-10 is trained on ModelA	66
4.2	The difference between conventional empirical risk and adversarial risk $ R_{adv} - R $ for test sets is tightened and minimised when the BNN is trained with IG. Corroborating this finding, the empirical results further explain the improved robustness of the IG-BNN networks.	68
4.3	Robustness against different numbers of EoT-PGD steps. EoT-PGD reaches its full strength after 20 steps. Further increasing PGD steps did not significantly improve the attack.	70
4.4	The accuracy and robustness of the BNN network trained on the STL-10 dataset, in which we force the model to be 'inconsistent' under clean and adversarial settings.	71
4.5	Transferability of adversarial examples between different particles from the STL-10 dataset	72
4.6	Transferability of adversarial examples between different particles from the CIFAR-10 dataset	72
5.1	A Trojan attack illustration from BadNets (Gu et al., 2019) demonstrating a backdoored model of a self-driving car running a STOP sign that could cause a catastrophic accident. Left: Normal sign (<i>benign input</i>). Right: Trojaneed sign (<i>Trojaneed input</i> with the Post-it note trigger) is recognised as a 100 km/h <i>speedlimit</i> by the Trojaneed network.	76
5.2	Overview of the Februus System . The Trojaneed input is processed through the <i>Trojan Removal</i> module that inspects and surgically removes the trigger. Subsequently, the damaged input is processed by the <i>Image Restoration</i> module to recover the damaged regions. The restored image is fed into the Trojaneed DNN. TOP: Without Februus, the Trojaneed input will trigger the backdoor and be misclassified as a 100 km/h SPEED LIMIT sign. BOTTOM: With Februus deployed, the Trojaneed DNN still correctly classifies the Trojaneed input as a STOP sign.	78
5.3	The distribution of deeply learned features of a Benign and Trojaneed model (the plots are obtained from CIFAR-10 using t-SNE (Van der Maaten and Hinton, 2008) applied to the outputs of the last fully connected layer).	81

5.4	Trojan information leaked is detected by the visual explanation tool GradCAM (Selvaraju et al., 2017). Based on the logit score of the Trojane network, the trigger pattern is the most important region causing the network to wrongly classify the image with the ground-truth label of <i>Aamma Sharif</i> to the targeted label of <i>A. Fine Frenzy</i> .	83
5.5	The training process of our generative adversarial network (GAN) for image restoration. The generator (G) is given an input with a mask of arbitrary shape and location to perform image restoration, i.e. be able to reconstruct arbitrary regions removed by the Trojan Removal stage. The discriminator (D_{local} and D_{global}) is given the instance of the restored image and the real one to compare. Notably, we utilise two discriminators to capture the global structure as well as local consistency.	84
5.6	Trojan triggers (first row) and their deployment used in our experiments (second row). From left to right: the flower and Post-it note triggers (used in (Gu et al., 2019)) deployed in CIFAR-10, BTSR and GTSRB tasks respectively, country flag lapels on shirts and the tattoo on the face are deployed in the VGGFace2 task.	88
5.7	Image Restoration. Visualisation of Trojane and benign inputs through Februus on different visual classification tasks.	94
5.8	Different triggers for the same targeted label. An attacker can use either trigger patterns (flag lapels) to impersonate the target person of interest (results are in Table 5.8).	96
5.9	Trojan attacks with varying trigger locations are successfully removed by Februus. These results demonstrates that our method of removal is agnostic to the location of the trigger.	98
5.10	Adversarial examples of Trojane images to fool Gradcam. Notably, when the perturbation is large ($\epsilon > 0.15$), GradCAM is misled; however, this leads the model to ignore the Trojan trigger as well; consequently, the Trojan attack is no longer successful.	99
5.11	Classification Accuracy, Attack Success Rate (ASR) and Confidence denoted by the prediction scores of the DNN models built with adaptive Trojaning for different penalizations (γ).	100

5.12	Februus applied to the infected GTSRB model whilst increasing the size of the Post-it trigger and illustrations of large triggers occluding 25% of the input images.	102
5.13	Multiple-piece trigger targeting a single label.	102
5.14	Adaptive Attacks on GradCAM. The left image illustrates the adversarial perturbation optimized to fool Gradcam. The right picture shows that GradCAM is fooled into detecting the targeted region.	103
6.1	An overview of the evolution of adversarial patch attack perturbation vectors. Our attack explores the hitherto elusive goal to realise visibly less malicious-looking adversarial patches–TnT–for: i) targeted attacks to misdirect any input to a target class; and ii) untargeted attacks.	112
6.2	Attack method for generating TnTs. Here, A is the patch stamping process, y_{target} is the targeted class designated by the attacker, y_{source} is the ground-truth label, $\ell(\mathbf{x}', y_{\text{target}}, y_{\text{source}})$ is the combined cross-entropy loss between the predicted score from the classifier f and the targeted as well as source label, and $\nabla \ell$ is the feedback from the Targeted Classifier f . The method is designed to iteratively approach high attack success TnTs by traversing through the latent space of the generator using gradient feedback.	115
6.3	An illustration of trigger locations around the border.	130
6.4	TnTs realised in PubFig dataset and their corresponding ASRs for the <i>Face Recognition</i> task to impersonate <i>anyone</i> to the targeted person Barack Obama.	133
6.5	Selective examples of random color and flower patches in our study on PubFig and ImageNet. The misclassification results caused by these patches are described in Table 6.10.	133
6.6	An instance of the random ordering of patch images used in the two user studies, 250 participants participated in each study.	135
6.7	Generated patches from our Adversarial Patch Generator for the 14 different targeted labels in LaVAN. The 1st row: Our adversarial patches in the scene. The 2nd row: Our adversarial patches were rescaled to image size for display purposes. The 3rd row: adversarial patches from LaVAN for the same targeted label.	138

6.8	Investigating Attack Success Rate (ASR) and patch size. Our TnT is comparable with the Noisy AdvPatch (Brown et al., 2017), and significantly better than the Disguise AdvPatch (Brown et al., 2017). <i>Our Adversarial Patch ASR outperforms both Noisy and Disguise counterparts.</i> . . .	139
6.9	Various settings employed for the physical world attacks to impersonate ‘Barack Obama’. Results demonstrate our TnT is effective, even under different, complex, physical-world settings ranging from indoor to outdoor with different lighting conditions. The network recognises the person with the TnT to be Barack Obama with high confidence.	140
6.10	Physical deployment of TnTs generated from the TnT Generator targeting different classes (shown on top in red) in the ImageNet task. . .	141

List of Tables

3.1	Hyper-parameter settings used in experiments	43
3.2	Model Architecture for SOREL-20M and EMBER2018 datasets	43
3.3	Comparing the performance of BNN and FFNN on different datasets (at 1%FPR)	44
3.4	Robustness of networks against adversarial malwares with increasingly large attack budgets	46
3.5	Comparing the robustness of detectors against <i>real</i> and <i>unseen</i> adversarial malware (problem space attacks).	47
3.6	Assessing the contribution of the number of parameter particles to the robustness of the networks for the SOREL-20M dataset.	48
3.7	Transferability of robustness to different kinds of attacks.	48
3.8	Comparing performance of Adversarially trained Bayesian Neural Networks	49
4.1	Hyper-parameters setting in our experiments	65
4.2	Comparing robustness under different levels of EoT-PGD attacks (or attack budgets).	65
4.3	Ablative study assessing the contribution of the Bayesian inference method at different levels of EoT-PGD attack (or attack budget).	67
4.4	Ablative study assessing the contribution of the IG objective at different levels of EoT-PGD attack (or attack budget).	68
4.5	Black-box attack to evaluate the obfuscated gradient effect.	69
4.6	Transferability. PGD ℓ_∞ trained IG-BNN robustness against different adversaries under different attack budgets.	69
5.1	Networks used for the classification tasks	88
5.2	Model Architecture for CIFAR-10. FC: fully-connected layer.	90
5.3	Model Architecture for GTSRB	90
5.4	Model Architecture for VGGFace2	91

List of Tables

5.5	Dataset and Training Configuration	91
5.6	Classification accuracy and attack success rate before and after Februus on Trojan models on various classification tasks.	92
5.7	Robustness of Februus against benign inputs in the classification tasks. Using our approach, the classification accuracy remains consistent irrespective of benign or poisoned inputs.	94
5.8	Robustness against various complex and adaptive Trojan attacks. Februus is robust against attacks with varying levels of complexity. . . .	96
5.9	Average run-time of different classification tasks on 100 images. Even with the high-resolution images of the Face Recognition task using a complex VGG-16 network, the total run-time of the Februus system is 29.86 ms, while the simpler scene classification task only incurs a 6.32 ms overhead.	106
5.10	Comparison between Februus and other Trojan defence methods	107
6.1	Networks used for the classification tasks	122
6.2	Dataset and Training Configuration	123
6.3	Model Architecture for CIFAR-10. FC: fully connected layer.	124
6.4	Model Architecture for GTSRB	125
6.5	Model Architecture for VGGFace2	125
6.6	Different TnTs found using ImageNet-100 for different models and their corresponding ASRs when applied to larger test sets.	127
6.7	Altering the location of a TnT increases ASR as it covers the main features of inputs. Notably, the TnT here realised from a small sample of 100 images (ImageNet-100), generalises extremely well to larger populations to fool any inputs to the targeted class conch. An illustration of TnT locations are shown in Fig. 6.3 and results are from the <i>WideResNet50</i> pre-trained model from Pytorch (Paszke et al., 2019) .	129
6.8	Black-box attack, the transferability of TnT from a model to other models on ImageNet dataset in an untargeted setting. ASRs are observed on 100 random images (network performance on the task is given in parenthesis).	131
6.9	Illustrative examples of TnTs found in different visual classification tasks and their corresponding ASRs	132

6.10	Study results from affixing patches of either random colors or flowers to the test samples of each dataset. As observed, the success rate for an attacker employing such <i>simple tricks is fairly low</i>	134
6.11	User studies to evaluate the Naturalistic Score of TnTs in comparison to other baseline patches. The Naturalistic Score is the percentage of participants' votes. Complete details of patches used in the user studies are in shown in Figure 6.6.	135
6.12	Comparing ASR with LaVAN (Karmon, Zoran and Goldberg, 2018)—smallest state-of-the-art (noisy) adversarial patches of 2% of input image size—using the 14 targeted labels used in LaVAN on the <i>Inception-V3</i> network. Our patches achieve higher ASR in both settings; high and low confidence scores. Detailed examples of labels are shown in Figure 6.7.	138
6.13	Effectiveness of TnT attacks against robust defence methods (↑ <i>Robustness</i> is better for defences).	144

Chapter 1

Introduction

THIS introductory chapter presents the motivation for considering the problems this dissertation addresses and discusses the challenging nature of those problems. The chapter also summarises the contributions made in the following chapters and provides an overview of the dissertation's structure.

1.1 Introduction

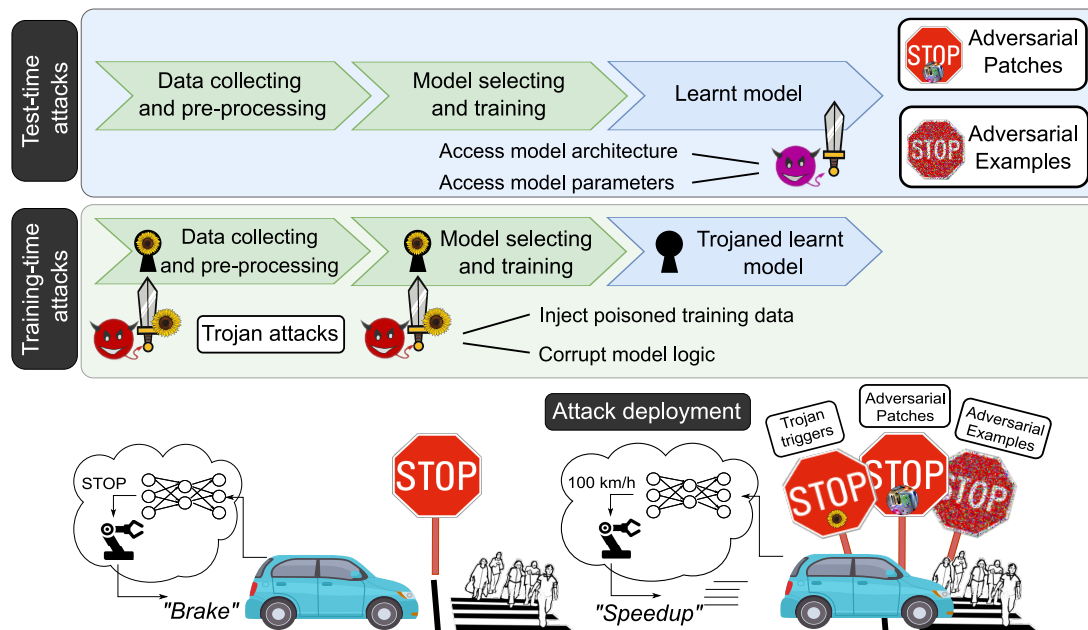


Figure 1.1: **Illustration of attacks on deep neural networks (DNNs) along the model building pipeline in the Computer Vision domain. DNN systems have been proven to be susceptible to threats from attacks at i) test time (top), ii) training time (middle). In each attack, the adversaries have the same mal-intention; to cause the DNN to make malicious decisions (bottom).**

Deep Neural Networks (DNNs) are increasingly entrusted to make critical decisions in the context of, for example, self-driving cars (Chen et al., 2015), disease diagnosis (Anwar et al., 2018) and face recognition (Taigman et al., 2014), often driven by their superhuman performance capabilities. However, as illustrated in Figure 1.1, the increasing pervasiveness of DNNs has increasingly incentivised malevolent actors to attack them. In the context of this dissertation, it is useful to categorise attacks into two types based on the stage at which an attacker accesses the model along a typical model building pipeline:

- *Test-time* attacks (Goodfellow, McDaniel and Papernot, 2018), in which attackers do not tamper with the model but access information about the model such as its architecture and parameters to exploit its vulnerabilities (illustrated at the top of Figure 1.1), as in the case of: i) *Adversarial Examples* based on generating unbounded perturbations (Szegedy et al., 2014; Goodfellow, Shlens and Szegedy, 2015; Madry et al., 2018; Moosavi-Dezfooli et al., 2017; Carlini and Wagner,

2017b); or ii) *Adversarial Patches*—spatially bounded perturbations (Brown et al., 2017; Karmon, Zoran and Goldberg, 2018))—to apply to a specific input.

- *Training-time* attacks, in which attackers are able to exert some control over the model learning stage to inject poisoned training data or corrupt model logic (shown at the middle of Figure 1.1), as in the case of *Trojan or backdoor attacks* (Gu, Dolan-Gavitt and Garg, 2017; Liu et al., 2018b; Chen et al., 2017; Severi et al., 2021) where, for example, parts of the training data are manipulated by injecting poisoned training data or the model logic is corrupted to inject a hidden backdoor into the learnt model.

The malicious intention of the adversary in each attack is the same—to cause the DNN to fail by making an incorrect decision, described as *untargeted attacks*, or manipulating the DNN to make the desired malicious decision, described as *targeted attacks* as shown at the bottom of Figure 1.1. The following subsections detail these attacks.

Adversarial Example Attacks. At test time, DNN systems confront Adversarial Examples, which are *unbounded input-specific* perturbations of additive *noise-like* vectors carefully crafted and applied to inputs to fool, misguide or hijack the decision of the DNN model (Goodfellow, Shlens and Szegedy, 2015; Madry et al., 2018; Carlini and Wagner, 2017b). Figure 1.1 demonstrates an Adversarial Example attack where the attacker deploys an unbounded-noisy perturbation that is added to an input. Although Adversarial Examples originate in the Computer Vision (CV) domain, they have been shown to exist in other domains, including audio (Carlini and Wagner, 2018), natural language processing (Liang et al., 2018) and, more recently, software code (e.g. malware) (Pierazzi et al., 2020).

Adversarial Patch Attacks. On the other hand, spatially bounded adversarial perturbations in LaVAN (Karmon, Zoran and Goldberg, 2018), commonly referred to as adversarial patches, have demonstrated the existence of *input-agnostic* additive noise patterns crafted from models at test time to mislead the prediction of *any* input to a DNN to mount a targeted attack. In contrast to LaVAN, the recent attack employed by Brown et al. (2017) constructed a *physically realisable* and *input-agnostic* Adversarial Patch to easily mount a targeted attack in the *physical world*. Figure 1.1 demonstrates an Adversarial Patch attack where the attacker is constrained to perturbations *spatially bounded* to a region of an input. This enables the attacker to construct a *printable*

1.1 Introduction

noise pattern to hijack the decision of a DNN, when stuck on an object, to achieve the attacker's target decision.

Trojan Attacks. Unlike Adversarial Examples and Patches, a *recent* Machiavellian attack demonstrates how an adversary can exploit model learning stages to exert greater control over an attack by constructing a backdoor in a trained model (Gu et al., 2019). Training a model requires *i*) massive amounts of training examples with carefully labelled ground truths, which is often difficult, expensive or impractical to obtain; *ii*) significant and expensive computing resources—often clusters of GPUs; and *iii*) specialised expertise for realising highly accurate models. Consequently, practitioners rely on transfer learning to reduce the time and effort required or Machine Learning (ML) as a Service (Amazon, 2022; BVLC, 2022) to build DNN systems. In transfer learning, practitioners re-utilise pre-trained models from an open-source model zoo with potential model vulnerabilities, intentional or otherwise, such as (Koh, 2022). In ML as a Service, the model-building task is outsourced and *entrusted* to a third party. Unfortunately, these approaches provide malicious adversaries opportunities to manipulate the training process, for example, by injecting poisoned training examples or corrupting the model by manipulating the model's neurons as in (Gu, Dolan-Gavitt and Garg, 2017; Liu et al., 2018b; Chen et al., 2017; Severi et al., 2021; Hong, Carlini and Kurakin, 2021), to create a *backdoor* or a *Trojan* in the model.

A distinguishing feature of a *Trojan attack* is a secret backdoor activation trigger with a shape, size and features self-selected by the adversary—that is, *independently* of the DNN model. Later, when the mode is deployed in an application, this trigger is used to activate the backdoor. Trojanned models behave normally for benign (clean) inputs. However, when the trigger, that is, a sticker or an object known and determined solely by the attacker in the CV domain, is placed in a visual scene to be digitised, the trojanned model misbehaves (Gu et al., 2019; Liu et al., 2018b; Chen et al., 2017; Bagdasaryan et al., 2020).

Importantly, and *unlike an Adversarial Patch*, a trigger can be any natural-looking object. For example, as Figure 1.1 illustrates, the digitised input containing the trigger (a flower sticker) on a STOP sign is classified by the trojanned model to the attacker's targeted class of 100km/h, with potentially devastating consequences. The ability to self-select trigger that is *physically realisable in a scene* and either or all of natural, surreptitious and inconspicuous makes Trojan attacks easily deployable in the real world without raising suspicions.

1.2 Challenges and Opportunities

Identifying security vulnerabilities or model deficiencies and realising robust DNNs against these attacks is non-trivial, and providing security is challenging. Unfortunately, with millions of parameter values within a DNN model, it is extremely difficult to explain or decompose the decision made by a DNN or examine a model to identify hidden classification behaviour (Samek et al., 2017; Wierzynski, 2018) to recognise and mitigate the threats discussed. The following sections explore the challenges posed by the threats discussed, enabling elucidation of possible defences and potential open avenues for investigating the realisation of robust DNNs.

1.2.1 Challenges

Robustness Against Adversarial Samples. ML models, especially DNNs, have traditionally been developed with the assumption that both the training and test environment are benign, which is helpful for building a highly accurate model. In particular, the inputs are usually assumed to be independently drawn from the same probability distribution at both training and test time (Goodfellow, McDaniel and Papernot, 2018). This assumption has been useful for realising an accurate model but implicitly signals attackers can choose a distribution at test time that is designed to be difficult for the model to accurately handle, as in the case of Adversarial Examples or Patches. Defending against such adversaries is challenging because the attack space is tremendously large, with various possibilities for attacks, and defenders have little to no knowledge of which attack might be deployed at test time. Although this suggests a possible arms race between attackers and defenders, it is a one-sided race, featuring multiple effective attack algorithms (Carlini and Wagner, 2017b; Madry et al., 2018) but few strong countermeasures that have nonetheless been demonstrated to be susceptible to defeat by adaptive attackers (Carlini and Wagner, 2017a; Papernot, McDaniel and Goodfellow, 2016). It is even more challenging in the malware domain, where there is a significant lack of robust defence methods against these threats due to the *inverse-mapping problem* (Biggio et al., 2013), which describes mapping from the problem space (e.g. software code) to the feature space that is neither differentiable nor invertible.

Defences Against Trojan Attacks. In a Trojan attack, a ML model will only exhibit abnormal behaviour if the secret trigger design appears while functioning correctly

1.2.2 Opportunities

in all other cases. Additionally, the trojaned network demonstrates state-of-the-art performance for the classification task that is comparable with that of a benign network, distinguished only by the malicious behaviour awaiting its trigger. Because the Trojan trigger is a *secret* known only by the attacker, the defender has no knowledge of the trigger, and it is unrealistic to expect the defender to imagine the characteristics of an attacker's secret trigger. The unbounded capacity of the attacker to craft triggers implies that the problem of detection is akin to *looking for a needle in a hay stack*. Further, unlike software, it is extremely difficult to examine a non-human readable model comprising millions of parameters to identify backdoors in a DNN model.

1.2.2 Opportunities

Opportunity 1: Exploring Bayesian Adversarial Learning for Defences Against Adversarial Examples. Significant research efforts have been made to mitigate the threat of Adversarial Examples, including distillation (Papernot et al., 2016), input denoising (Song et al., 2017) and feature denoising (Xie et al., 2019) as described in more detail in (Kurakin et al., 2018). Among these methods, adversarial training (Madry et al., 2018) and its variants have been shown to be among the most effective methods for defending against Adversarial Example attacks (Athalye, Carlini and Wagner, 2018). Now, a network is trained with Adversarial Examples to build robustness against input perturbations at model deployment. However, as mentioned by Ye and Zhu (2018), the adversarial training algorithm relies on the '*point estimate*' approach of a DNN, that is, a fixed set of network parameters maps the input to the output. Essentially, a point estimate with a choice of parameters defines only a single decision boundary, which can be easily manipulated by an adversarial input unrestrained by the pre-defined adversarial constraints, for example, the maximum norm of perturbations.

Alternatively, we can use multiple decision boundaries from a distribution of model parameters and integrate out the effects of parameter choice to build robust models. That is the premise of Bayesian Neural Network (BNN) learning methods (Welling and Teh, 2011), which aim to learn a distribution over the parameters. Now, the output *predictive* distribution is obtained by integrating out the model parameters with respect to their distribution. Thus, it is intuitive to explore the robustness of adversarially trained Bayesian deep neural networks. However adversarial learning in the context of BNN or Bayesian adversarial learning research efforts remain in its infancy and

the robustness of Bayesian learning methods to defend against Adversarial Examples remains to be well understood.

Inspired by the premise of BNN and the effectiveness of adversarial training, this thesis considers the development of Bayesian adversarial learning methods for DNNs by combining adversarial training with Bayesian learning to build robust defences against threats from Adversarial Examples in the software domain and CV domain.

Opportunity 2: Exploring Information Leakages for Defences against Trojan Attacks. Although an insidious Trojan attack creates a hidden backdoor in a DNN, it requires manipulating the training process to generate a 'shortcut' (Wang et al., 2019), which would inevitably leave traces for investigation. In particular, the strong effect of Trojan attacks on a model (*i.e.* to misclassify any input with the trigger to the targeted class) to activate the embedded backdoor would probably leak information that could be exploited via a side channel to potentially detect the existence of a Trojan or such an attack on a deployed model. Although methods of detecting Trojans exists, it is desirable to neutralise the Trojan at run time in critical applications where denial of the service is not an option such as in self-driving cars. This thesis examines a method of exploiting potential information leakages to detect and neutralise Trojan attacks at run time in the CV domain.

Opportunity 3: Exploring Connections Between Trojan attacks and Adversarial Examples and Patches to Improve our Understanding. Although Adversarial Examples, Patches and Trojan attacks have been studied in the literature, the attacks are typically investigated in parallel, with few investigations into their fundamental connections (Pang et al., 2020). However, as Figure 1.1 illustrates, although the two attacks are conducted using different methods—one computes perturbations to apply to the input without modifications to the model to expose vulnerabilities in the model, and the other consciously manipulates the model to achieve a particular attack—both attacks share the common aim of forcing the DNN to misbehave in response to pre-defined inputs. Furthermore, as (Bagdasaryan and Shmatikov, 2021) recognises, the Trojan attacks relate closely to Adversarial Patches (Brown et al., 2017) in the sense that, without changing the model, Adversarial Patches cause the network to misclassify any input to an attacker-chosen label, similar to Trojan triggers. Hence, the attacks share some characteristics; understanding and establishing the potential linkages between the attacks are important for developing robust defences against

1.2.3 Research Questions

these attacks. This thesis examines a method to bridging the divide between Trojan attacks and Adversarial Patch in the input space.

1.2.3 Research Questions

Building upon the opportunities discussed, this thesis considers the following three research questions:

RQ1: How can we employ Bayesian adversarial learning with deep neural networks to defend against Adversarial Examples and are these methods effective?

RQ2: How can we exploit potential information leakages from trojaned models to build a robust defence method that can effectively mitigate Trojan attacks at run-time?

RQ3: How can an adversary bridge the divide between Trojan attacks and spatially bounded Adversarial Examples or Adversarial Patch attacks in the input space to exert the high level of control enabled by Trojan attacks without interfering with a neural network?

1.3 Summary of Original Contributions

In response to these research questions, the developments produced by this dissertation represent several original contributions to knowledge about building robust DNNs. These contributions can be summarised as follows:

1. A novel Bayesian adversarial learning algorithm for malware classification is proposed. We recognise that the adversarial learning approach for approximating the multi-modal posterior distribution of a Bayesian model can engender mode collapse. Thus, the model's achievements in terms of robustness and performance are sub-optimal. Instead, we first propose preventing mode collapse to better approximate the multi-modal posterior distribution by utilising Stein Variational Gradient Descent (SVGD) to generate diverse parameter particles. Our learning approach enables the model to both reduce the effect of

a single-parameter choice and learn the invariant patterns common between the training dataset and the corresponding adversarial samples. Second, we consider adversarial attacks in the feature space and theoretically demonstrate that the approach inherently generates robustness against problem-space adversarial attacks because we prove the projection of perturbed, yet valid, malware in the problem space into feature space will be a subset of feature-space adversarial attacks. Third, we prove that hardening BNNs with Adversarial Examples bounds the difference between adversarial risk and conventional empirical risk and improves robustness. This is the first time such a bound has been formally derived, and this is significant because it provides a theoretically justified approach to reducing the error associated with Adversarial Examples. The empirical results demonstrate that diverse BNNs improve the performance of malware classifiers—especially in the context of regimes characterised by a low false positive rate—and adversarially trained BNNs outperform their neural network counterparts on robustness against stronger unseen attack samples. This work, which addresses *RQ1*, is currently under review for the *Conference on Neural Information Processing Systems (NeurIPS'22)* under the title 'Improving the Robustness of Malware Detectors with Bayesian Neural Networks'.

2. A novel Bayesian adversarial learning method is proposed to learn a diversified BNN that is robust against adversarial attacks by *forcing the information gain from benign and adversarial instances to be the same* across particles. The approach is built upon the intuition that a robust model should *ignore perturbations* and only consider the informative content of the input, enabling us to conceptualise and formulate an *information gain objective* to measure and force the information learned from both benign and adversarial training instances to be similar. We prove that *minimising the information gain objective* further tightens the bound of the difference between adversarial risk and conventional empirical risk. This means that the risk of misclassification of an adversarial example is now the same as the risk of misclassifying a benign sample. Additionally, comprehensive evaluations of a set of neural architectures and datasets demonstrate that our approach achieves significant improvements in robustness compared to previous methods. This work, which addresses *RQ1*, has been accepted for publication at *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*

1.3 Summary of Original Contributions

- under the title 'Bayesian Learning with Information Gain Provably Bounds Risk for a Robust Adversarial Defense'.
3. We investigate a new Trojan defence concept—unsupervised *input sanitisation for DNNs*—and propose a *system architecture* to realise it. Our proposed architecture, *Februus*, aims to *sanitise* inputs by *i)* exploiting the Trojan-introduced biases leaked into the network to localise and surgically remove triggers in inputs and *ii)* *restoring* inputs for the classification task. Our extensive evaluations demonstrate that our method is a robust defence against *i)* input-agnostic Trojans—*our primary focus* and *ii)* complex adaptive attacks (multiple advanced backdoor attack variants and attacks targeting Februus modules). Februus enables effective Trojan mitigation by sanitising inputs with no loss of performance for sanitised inputs, whether Trojanned or benign. For our study, we built *ten* Trojan networks with *five* different realistic and natural Trojan triggers of various complexity, for example, a facial tattoo and a flag lapel on a T-shirt. Februus is shown to be efficacious. Significantly, we provide the first results for a defence against partial backdoor attacks, which show resilience against stealthy advanced Trojan attacks—multiple-trigger-to-multiple-target attacks—capable of evading state-of-the-art defence methods. To the best of our knowledge, Februus represents the first backdoor defence method for operation at run-time capable of sanitising Trojanned inputs, and, notably, the method does not require anomaly detection methods, model retraining or costly labelled data. This work, which addresses RQ2, has been published in the *Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC'20)* under the title 'Februus: Input Purification Defence against Trojan attacks on Deep Neural Network Systems'.
 4. Investigations into Adversarial Examples, Adversarial Patches and Trojan attacks lead to the discovery of an emerging threat against DNNs, whereby an adversary can bridge the divide between Trojan Attacks and spatially constrained Adversarial Examples or Adversarial Patches in the input space by generating *Universal Naturalistic adversarial patches*, which we call TnTs. The study explores the super-set of the spatially bounded Adversarial Example space and the natural input space within generative adversarial networks to construct TnTs. The TnTs generated to attack a DNN are *i)* universal and naturalistic, *ii)* highly effective and robust, *iii)* deployable in the physical world, *iv)* highly

generalisable and transferable to mount black-box attacks, and, especially, *v*) highly effective at evading existing countermeasures. Furthermore, our method generalises to generate physically realisable Adversarial Patches that can achieve higher attack success rates than state-of-the-art attacks. Interestingly, an Adversarial Patch attacker can now potentially exert a greater level of control by choosing a location-independent, natural-looking patch as a trigger, in contrast to being constrained to noisy perturbations. Thus far, this ability has been shown to be only possible using Trojan attack methods that have to interfere with model building processes to embed a backdoor at the risk of discovery. Nonetheless, in this context, the attacker can still realise a patch that is *deployable in the physical world*. This work, which addresses RQ3, was recently accepted—with minor revisions—for the *IEEE Transactions on Information Forensics and Security* under the title ‘TnT Attacks! Universal Naturalistic Adversarial Patches Against Deep Neural Network Systems’.

1.4 Dissertation Structure

The dissertation structure, outlined in Figure 1.2, is described as follows:

1. Chapter 1 and Chapter 2 provide a brief introduction and background to DNNs, detailing threats and countermeasures and discussing the challenges and opportunities associated with building robust DNNs.
2. Chapter 3 considers the problem of realising robust malware detectors using Bayesian adversarial learning. Additionally, a novel Bayesian neural network utilising SVGD is proposed to improve robustness against feature-space adversarial attacks, which further implies robustness against problem-space attacks. We also prove that this robust Bayesian learning provably bounds the difference between adversarial risk and empirical risk.
3. Chapter 4 focuses on improving the robustness of diverse BNNs in the CV domain. Consequently, a novel Bayesian adversarial learning method is proposed, in which the information gain from benign and adversarial instances is forced to be the same. Additionally, this learning method proves to tighten the bound of the gap between adversarial and conventional risk and empirically demonstrates enhanced robustness.

4. Chapter 5 considers the problem of advanced variants of training-time Trojan attacks. Consequently, a new defence concept—unsupervised input sanitisation for DNNs—is proposed. The method is efficacious in terms of sanitising the inputs at run-time and is highly effective against multiple complex adaptive Trojan attack variants.
5. Chapter 6 considers the emerging threat of a universal naturalistic adversarial attack that bridges Adversarial Examples and Trojan attacks in the input space. Consequently, an Adversarial Patch method based on generative adversarial networks is developed to generate an effective Adversarial Patch attack that is universal, naturalistic and, especially, robust in the harsh conditions of physical world implementation. Such attacks enable the attacker to exert the high level of control possible with Trojan attacks without having to interfere with the network or poison the dataset.
6. Chapter 7 provides a conclusive summary of the research produced by this dissertation and discusses potential future avenues for investigations.

Overview	Chapter 1	<ul style="list-style-type: none"> • Introduction. • Motivations and our contributions.
	Chapter 2	<ul style="list-style-type: none"> • Background on deep neural network attacks and defences. • Performance metrics to evaluate robustness.
Against Adversarial Examples	Chapter 3	<ul style="list-style-type: none"> • Investigating the problem of realising robust malware detectors using Bayesian adversarial learning. • Developing a robust Bayesian neural network utilising SVGD to defend against feature-space Adversarial Examples, which inherently improves robustness against problem-space attacks. • Theoretically proving that Bayesian adversarial learning bounds the difference between adversarial and empirical risks.
	Chapter 4	<ul style="list-style-type: none"> • Investigating the problem of improving the robustness of Bayesian neural networks in the Computer Vision domain. • Developing a novel Bayesian learning method by conceptualising the objective of enforcing the information gain (IG) from benign and adversarial instances to be similar. • Theoretically proving that Bayesian adversarial learning with IG tightens the bound, the difference between adversarial and empirical risks, to improve robustness.
Against Trojan Attacks	Chapter 5	<ul style="list-style-type: none"> • Investigating different variants of Trojan attacks with a specific focus on advanced Partial-Trojan attacks. • Developing a novel unsupervised input sanitisation defence method against multiple advanced Trojan attacking scenarios by exploiting the information leaked by a Trojan attack. • Demonstrating the effectiveness of the proposed method against known, advanced and, challenging, adaptive attacks.
Emerging Threat	Chapter 6	<ul style="list-style-type: none"> • Investigating the emerging threat of a novel attack capable of bridging spatially bounded Adversarial Examples or Adversarial Patches and Trojan attacks in the input space. • Developing a GAN-based adversarial attack to generate universal naturalistic adversarial patches. • Demonstrating the effectiveness of the threat against multiple state-of-the-art classifiers on different visual classification tasks and the effectiveness of existing state-of-the-art defence methods as well as the robustness of the threat in harsh physical-world conditions.
Conclusion	Chapter 7	<ul style="list-style-type: none"> • Conclusion. • Future research directions.

Figure 1.2: **Outline of the dissertation.**

Chapter 2

Background

THIS chapter provides a brief overview of the literature on deep neural networks, threats to these networks and existing countermeasures. A generic formulation for the problem is presented, and the notations utilised throughout the thesis are introduced. Common attack and defence techniques are discussed to establish foundations for the inquiries of the following chapters. Additionally, this chapter surveys the popular deep learning building blocks, the Generative Adversarial Networks and the Bayesian learning techniques incorporated into most of the methods that this research employs. Finally, the chapter introduces the common evaluation metrics that ground this work and quantify the experimental results.

2.1 Notations

For notational consistency, we use lowercase bold typeface letters (e.g., \mathbf{x}) for vectors, uppercase bold typeface letters (e.g., \mathbf{X}) for matrices, lowercase letters (e.g., x) for random variables, uppercase letters (e.g., \mathcal{X}) to represent sets, and \mathcal{X}_{adv} to denote sets of adversarial counterparts. Let $\|\mathbf{x}\|_p$ denote L^p norm of \mathbf{x} , \mathbf{x}_{adv} denote the adversarial version of \mathbf{x} , $\mathbf{A} \odot \mathbf{B}$ denote the element-wise (Hadamard) product of \mathbf{X} , and \mathbf{Y} ; $\nabla_{\mathbf{x}}y$ denote the gradient of y with respect to \mathbf{x} . Let $H(x)$ denote the Shannon entropy of the random variable x and $\text{KL}(P||Q)$ denote the Kullback-Leibler divergence of P and Q (probability distributions). We use $f(\mathbf{x}; \boldsymbol{\theta})$ to represent a function of \mathbf{x} parametrised by $\boldsymbol{\theta}$; sometimes, to simplify notation, we omit the argument $\boldsymbol{\theta}$ and instead write $f(\mathbf{x})$.

2.2 Machine Learning Empirical Process

This section describes a general approach to constructing a Machine Learning (ML) model that includes two phases, training and test:

Training. After the data is collected and pre-processed, as shown in Figure 1.1, an ML is chosen and trained. Most ML models can be considered parameterised functions $f_{\boldsymbol{\theta}}$ that map the input $\mathbf{x} \in \mathcal{X}$ from a domain (e.g. image) to a particular output \mathcal{Y} (e.g. traffic sign type in a classification task), where $\boldsymbol{\theta}$ is the parameter set according to which the ML model is fully defined (e.g. as a support vector machine or a neural network). A *learning algorithm* analyses the training data to realise the values of model parameters $\boldsymbol{\theta}$ (e.g. weights and biases) based on an *objective function* (e.g. to minimise loss ℓ between model predictions $f_{\boldsymbol{\theta}}(\mathbf{x})$ and the expected output y indicated by the dataset). The realised model is then evaluated on a validation dataset disjointed from the training dataset to verify the model's performance, especially the ability to *generalise* based on the validation dataset.

Test. Once training is complete, and the learnt model $\boldsymbol{\theta}$ is realised, it is *deployed* to make predictions on inputs that are unseen during training. The most common form of prediction for a classification task is a vector $\mathbf{y} \in \mathcal{Y}$ that assigns a probability to each class of the problem to characterise how likely the input is to belong to that class (e.g. traffic sign type).

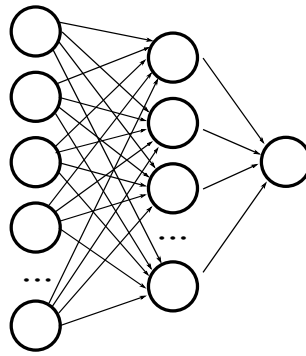


Figure 2.1: An illustration of a Deep Neural Network.

2.3 Deep Neural Networks

A Deep Neural Network (DNN) (illustrated in Figure 2.1) is a specialised ML model built as a composition of L hidden layers; the output of each layer l is a tensor \mathbf{a}_l (with the convention that $\mathbf{a}_0 = \mathbf{x}$). Similar to training an ML model, realising a DNN entails determining the parameters θ using the training dataset $D_{\text{train}} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ of n samples. The parameters are chosen to minimise the notion of loss ℓ for the impending task:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(f_{\theta}(\mathbf{x}_i), y_i). \quad (2.1)$$

Evaluation of the network involves using a separate validation set D_{val} with a ground-truth label. The following represent some of the common DNNs that appear in the following chapters:

Convolutional Neural Networks (CNNs). Convolutional Neural Networks (LeCun, Bengio and Hinton, 2015) are among the most popular choices of DNNs, especially in the computer vision domain. CNNs are a specialised kind of neural network used to process data that has a clear grid-structure topology, such as image data (Goodfellow, Bengio and Courville, 2016). In general, CNNs are formed by a stack of convolutional operators (known as kernels) that are much smaller than the input and can automatically extract different features from a given input. In the context of image recognition problems, these kernels can capture edges or specific colours. Their outputs are called feature maps, which are finally unified and combined to make predictions in classification tasks.

Generative Adversarial Networks (GANs). A Generative Adversarial Network (Goodfellow et al., 2014) is another kind of deep neural network that

2.4 Bayesian Learning

has demonstrated significant success in various applications of realistic image synthesis. A GAN comprises a Generator G and a Discriminator D . In a classical GAN, if i) \mathbf{x} is an input, which is an image of 3D tensor (width \times height \times depth) and ii) \mathbf{z} is a latent vector of dimension N , which is sampled from a noise distribution $P(\mathbf{z})$, the Generator G maps a source of noise $\mathbf{z} \sim P(\mathbf{z})$ to generate a synthetic image¹ $\tilde{\mathbf{x}} = G(\mathbf{z})$, and the Discriminator functions to distinguish the fake synthetic images $\tilde{\mathbf{x}}$ from the real ones \mathbf{x} , and the feedback from this Discriminator is utilised to help the Generator improve image quality.

There are different methods for training a GAN. This thesis applies the Wasserstein GAN with Gradient Penalty (WGAN-GP) (Gulrajani et al., 2017) because it has been shown to stabilise the GAN training process and improve the fidelity of the samples generated. It involves solving the following optimisation problem:

$$\min_G \max_D \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\mathbf{x}} D(\hat{\mathbf{x}})\|_2 - 1)^2],$$

where \mathbb{P}_r is the distribution of real images, and $\mathbb{P}_{\hat{\mathbf{x}}}$ is the distribution of the interpolation between real and synthesised images. Here, \mathbb{P}_g is the conditional distribution of the synthesised images, which we sample from the Generator, that is, $\tilde{\mathbf{x}} = G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})$. Using this min-max optimization, we learn \mathbb{P}_g to match \mathbb{P}_r . Samples from \mathbb{P}_r are drawn from a dataset of real objects; consequently, by sampling from the realised Generator, we can obtain naturalistic image samples.

2.4 Bayesian Learning

A Bayesian Neural Network (BNN) is a stochastic DNN trained using Bayesian inference in which, as Figure 2.2 illustrates, the traditional weight values of a DNN are replaced by their stochastic counterparts to simulate multiple possible model parameters θ and their probability distribution $p(\theta)$.

Given a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, a BNN aims to learn the *posterior* distribution $p(\theta | \mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$ based on the prior distribution $p(\theta)$. However, the exact solution for the posterior distribution is often *intractable* due to the high-dimensional integral of the denominator, even for moderately sized networks in the deep learning context (Blei,

¹The parameters are omitted for brevity but both generator and discriminator feature individual sets of parameters.

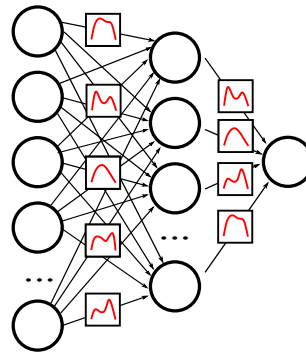


Figure 2.2: An illustration of a Bayesian Neural Network.

Kucukelbir and McAuliffe, 2017). Additionally, the true Bayesian posterior is usually a complex multimodal distribution (Izmailov et al., 2021) as Figure 2.3 illustrates.

Efforts to develop a suitable inference approach to approximate the posterior distribution involve using either Markov Chain Monte Carlo, which is asymptotically accurate but slow (Welling and Teh, 2011), or Variational Inference, which is efficient but inaccurate (Blei, Kucukelbir and McAuliffe, 2017). Additionally, Variational Inference, which relies on a parametric function, is too restrictive to resemble the true posterior distribution and suffers from mode collapse (Izmailov et al., 2021). Meanwhile, Wang and Liu (2019); Liu and Wang (2016) proposes a provable general-purpose variational inference algorithm named Stein Variational Gradient Descent (SVGD) that transports a set of parameter particles, encouraged to be diverse, to fit the true posterior distribution. This approach can be beneficial for achieving better performance and approximating the true posterior distribution. Given the challenges mentioned and the benefits of the SVGD, this thesis employs the SVGD in Chapters 3 and 4. A visualisation of the different techniques used to sample the posterior distribution is presented in Figure 2.3.

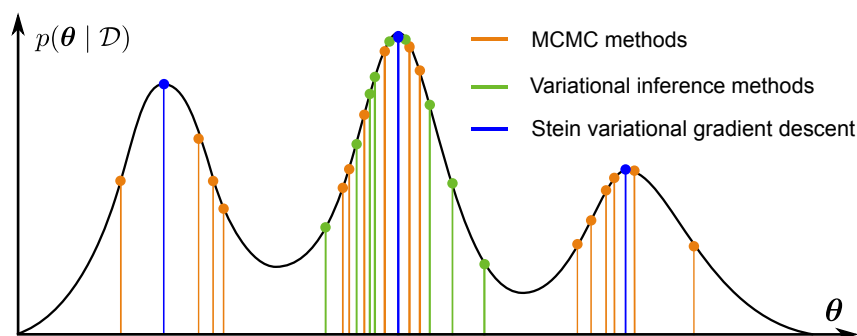


Figure 2.3: Different techniques for sampling the posterior distribution.

2.5 Threat Model

Up until this point, we introduce the related background to build DNNs. The following section will introduce a threat model to allow reasoning in terms of the robustness of DNNs.

2.5 Threat Model

This section introduces a general threat model that is commonly utilised to evaluate the robustness of DNNs. Specific threat models tailored for each of our proposed methods are detailed in the following chapters.

In general, a threat model can be defined by the adversary's goals and capabilities (Papernot et al., 2018).

Adversary Capabilities. Depending on the amount of access the adversary has to the model, it is possible to discuss two scenario categories: *white-box* scenarios, where the adversary has full access to the model, including the ML algorithm and the model parameters), and *black-box* scenarios, where the attackers rely on guesswork because the ML algorithm and model parameters are unknown (Goodfellow, McDaniel and Papernot, 2018; Vo, Abbasnejad and Ranasinghe, 2022a,b). Most of this dissertation considers *white-box* settings, with the exception of Chapter 6, which considers both *white-box* and *black-box* settings for the emerging threat. This is because even if access to the model is not possible, or the model is not publicly available, adversaries can employ a reverse engineering approach to extract the model, as exemplified by (Tramèr et al., 2016; Rolnick and Kording, 2020; Carlini, Jagielski and Mironov, 2020). Additionally, because defending against such attacks is challenging, building a robust DNN is of particular interest.

Adversary Goals. One classical approach to understanding adversary goals is developing a model of desired ends in terms of confidentiality, integrity, and availability impacts (called a CIA model). Papernot et al. (2018) adds the fourth component of privacy to fully capture adversary goals. This thesis mainly focuses on integrity, enabling it to address two of the major challenges confronting DNNs, namely, Adversarial Examples (AEs) and Trojan attacks (Szegedy et al., 2014; Gu, Dolan-Gavitt and Garg, 2017), where the attack is designed to control model outputs (*i.e.* the integrity of the model is compromised), and the goal is to induce the model behaviour dictated by the adversary. That is, the integrity of DNNs can be compromised by adversaries capable of manipulating either model inputs at test time, as in the case

of AEs, or training data, as in the case of Trojan attacks. The following sections detail these two types of attacks.

2.6 Adversarial Examples

Attacks at test time do not interfere with a model's training process. A well-known test-time attack is AEs, with Szegedy et al. (2014) first demonstrating the existence of AEs in the context of image classification neural networks. Subsequently, substantial research has been conducted investigating this problem (Goodfellow, Shlens and Szegedy, 2015; Madry et al., 2018; Carlini and Wagner, 2017b). In general, attackers employing AEs can add carefully crafted noise (perturbations) to the input image by exploiting networks at test time to alter the classifier's prediction. The goal of the attacker is to degrade the performance of a neural network by crafting δ , such that:

$$\max_{\|\delta\|_p < \varepsilon_{\max}} \ell(f(\mathbf{x} + \delta; \theta), y) \quad (2.2)$$

where p is the norm, ε_{\max} is the maximum attack budget (perturbation), ℓ is the loss function (typically cross-entropy), f is the network, \mathbf{x} is the input, θ is the network parameter, and y is the ground-truth label.

Among the most popular AE attacks are Projected Gradient Descent (PGD) (Madry et al., 2018) attacks, which see an attacker start from $\mathbf{x}^0 = \mathbf{x}_o$ and conduct PGD iteratively to update the AE in accordance with Equation (2.3):

$$\mathbf{x}^{t+1} = \Pi_{\varepsilon_{\max}} \{ \mathbf{x}^t + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(f(\mathbf{x}^t; \theta), y_o)) \} \quad (2.3)$$

where $\Pi_{\varepsilon_{\max}}$ is the projection to the set $\{ \mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_o\|_{\infty} \leq \varepsilon_{\max} \}$.

This thesis adopts PGD attacks in its experiments for two reasons: i) PGD (Madry et al., 2018) attacks are regarded as the strongest form of attack in terms of the ℓ_{∞} norm; ii) this approach enables direct control over distortion by changing ε_{\max} .

Significant research efforts describe methods of mitigating this threat, including distillation (Papernot et al., 2016), input denoising (Song et al., 2017) and feature denoising (Xie et al., 2019); curious readers can find more from (Kurakin et al., 2018). Among these methods, adversarial training (Madry et al., 2018) (and its variants) represents one of the most effective and popular methods of defending

2.7 Trojan Attacks

against adversarial attacks (Athalye, Carlini and Wagner, 2018). The goal of adversarial training is to incorporate the adversarial search within the training process and, thus, realise robustness against AEs at test time. This is achieved by solving the following optimisation problem:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{(x,y) \sim D} \left\{ \max_{\|\delta\|_p < \epsilon_{\max}} \ell(f(x + \delta; \theta), y) \right\} \quad (2.4)$$

where \mathcal{D} is the training data. An approximate solution for the inner maximisation can be realised by generating the PGD-based AEs from Equation (2.3) and then minimising the classification loss according to the generated PGD-based AEs.

2.7 Trojan Attacks

Attacks at training time attempt to influence or corrupt the model itself. One particular recent training-time threat is Trojan or backdoor attacks. Clandestine insertion of a backdoor in a DNN model, as in BadNets (Gu et al., 2019) or the NDSS 2018 Trojan attack study (Liu et al., 2018b), has two requirements: *i*) the DNN must be taught a trigger to activate the backdoor and misclassify a trigger stamped input to the targeted class; *ii*) the backdoor must remain hidden inextricably within potentially millions of parameter values in a DNN model. To Trojan a model, an attacker creates a *poisoned* set of training data. An adversary with direct access to the training dataset D_{train} , as in BadNets attacks, can generate a poisoned dataset by stamping the trigger onto a subset of training examples. Consider the following: let k be the proportion of samples needed to be poisoned ($k \leq n$) and A be the trigger stamping process; then, the poisoned data subset $S_{\text{poisoned}} = \{\mathbf{x}_{i_p}, y_{i_p}\}_{i=1}^k$ will contain the poisoned data $\mathbf{x}_{i_p} = A(\mathbf{x}_i)$ and their labels $y_{i_p} = t$; here, t is the chosen targeted class. This poisoned data subset S_{poisoned} will replace the corresponding clean data subset in D_{train} during the training process of the DNN to build the Trojanged model for the attack. When the Trojanged model is deployed in an application by a victim, stamping the secret trigger on any input will misclassify the input to the targeted class t .

2.8 Evaluation Metrics

This section introduces the metrics commonly used to evaluate the robustness of a DNN in the literature in order to ground studies and quantify experimental results.

First, we provide the definition for the primitive terminologies of *True Positive (TP)*, *False Positive (FP)*, *True Negative (TN)* and *False Negative (FN)*.

- *True Positive (TP)* indicates the number of predictions where the network correctly predicts the positive-class samples as positive.
- *True Negative (TN)* indicates the number of predictions where the network correctly predicts the negative-class samples as negative.
- *False Positive (FP)* indicates the number of predictions where the network incorrectly predicts the negative-class samples as positive.
- *False Negative (FN)* indicates the number of predictions where the network incorrectly predicts the positive-class samples as negative.

Leveraging these terms, we present formal definitions of *Accuracy*, *Robustness*, *Attack Success Rate*, *True Positive Rate*, *False Positive Rate* and *Receiver Operating Characteristics* in binary-classification tasks as well as their adopted versions in multi-class classification tasks use in this thesis.

2.8.1 Binary Classification

The following metrics are employed in evaluations of binary-classification tasks; more specifically, the malware detection problem discussed in Chapter 3.

Accuracy. We compute the overall accuracy of the model, denoted by Acc , to quantify the proportion of samples that were correctly classified by the classifier:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2.5)$$

Robustness. This metric, denoted by *Robustness*, quantifies the accuracy of the model under the threat of adversarial attacks according to the following equation:

$$\text{Robustness} = \text{Acc}(\mathbf{x}_{\text{adv}}), \quad \mathbf{x}_{\text{adv}} \sim \mathcal{D}_{\text{adv}} \quad (2.6)$$

Attack success rate. This metric, denoted by ASR , quantifies the proportion of samples that were incorrectly classified by the classifier according to the following equation:

$$\text{ASR} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2.7)$$

2.8.2 Multi-class Classification

True positive rate (TPR). For a given test, this metric, denoted by TPR (also known as *Recall* or *Sensitivity*), quantifies the fraction of all positive samples that the classifier correctly predicted as positive according to the following equation:

$$\text{TPR} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad (2.8)$$

False positive rate (FPR). For a given test, this metric, denoted by FPR and equivalent to $(1 - \text{Specificity})$, where *Specificity* is the TPR, quantifies the fraction of all negative samples that the classifier incorrectly predicted as positive according to the following equation:

$$\text{FPR} = \frac{\text{FP}}{(\text{FP} + \text{TN})} \quad (2.9)$$

Receiver Operating Characteristics. All possible combinations of TPR and FPR compose a Receiver Operating Characteristics space, denoted by ROC, with each point in ROC space determined by a pair (TPR, FPR), showing the trade-off between *Sensitivity* and *Specificity*.

2.8.2 Multi-class Classification

Notably, the evaluation metrics presented above provide a quantification method for a binary classification task. For a multi-class classification (*e.g.* k -class classification), global metrics can be calculated as described below. These metrics are used to evaluate the robustness of DNNs and, in particular, will be utilised in the problems addressed in Chapter 3, 4, 5, 6.

Accuracy. Global accuracy, denoted by Acc_g , is calculated using the following equation:

$$\text{Acc}_g = \frac{\sum_{i=1}^k \text{TP}_i}{\text{Total}} \quad (2.10)$$

where *Total* is the total number of samples in the evaluated set, and TP_i is the number of correct predictions for each of the class i in a k -class classification task.

Robustness. Global robustness, denoted by Robustness_g , is calculated using the following equation:

$$\text{Robustness}_g = \frac{\sum_{i=1}^k \text{Robustness}_i}{\text{Total}} \quad (2.11)$$

where Robustness_i is the robustness measured for each of the class i in a k -class classification task.

Attack Success Rate. Global ASR, denoted by ASR_g , is calculated using the following equation:

$$\text{ASR}_g = \frac{\sum_{i=1}^k \text{FN}_i}{\text{Total}} \quad (2.12)$$

where FN_i is the number of incorrect predictions for each of class i in a k -class classification task.

Chapter 3

Bayesian Adversarial Learning for Robust Malware Detectors

THIS chapter considers the problem of learning a detector robust against adversarial malware. Modern detectors rely on machine learning algorithms; however, an adversary can devise alterations to the malware code to decrease the chance of being detected whilst *preserving the functionality and realism of the malware*. Adversarial learning is effective in improving robustness but generating functional and realistic adversarial malware samples is non-trivial because in contrast to tasks capable of using gradient-based feedback, adversarial learning in a domain without a *differentiable mapping function* from the *problem space* (malware code inputs) to the *feature space* is hard. This presents a challenge for developing scalable adversarial machine learning algorithms for production scale datasets to realise robust malware detectors. We propose an alternative; perform adversarial learning in the *feature space* in contrast to the problem space. We *prove* the projection of perturbed, yet valid malware, in the problem space into feature space will always be a subset of adversarials generated in the feature space. Hence, by generating a robust network against feature-space adversarial examples, we inherently achieve robustness against problem-space adversarial examples. We formulate a Bayesian adversarial learning objective that captures the distribution of models for improved robustness. To *explain* the robustness of the Bayesian adversarial learning algorithm, we *prove* that our learning method bounds the difference between the adversarial risks and empirical risk and improves robustness. We show the Bayesian neural networks (BNNs) achieve state-of-the-art results; especially in the False Positive Rate (FPR) regime. Adversarially trained BNNs achieve state-of-the-art robustness.

3.1 Motivation and Contribution

We are amidst a meteoric rise in malware incidents worldwide. Malware is responsible for significant damages, both financial—in billions of dollars (Anderson et al., 2019)—and human costs in loss of life (Eddy and Perlroth, 2020). According to statistics from Kaspersky Lab, at the end of 2020, there were an average of 360,000 pieces of malware detected per day (KasperskyLab, 2020). The battle against such large incidents of malware remains an ongoing challenge and the need for automated and effective malware detection systems is a research imperative.

Advances in Machine Learning (ML) has led to state-of-the-art malware detectors (Arp et al., 2014; Peng et al., 2012; Harang and Rudd, 2020; Raff et al., 2018; Anderson and Roth, 2018). But, ML-based models are known to be vulnerable to *adversarial examples*; here, seemingly benign inputs with small perturbations can successfully evade detectors. Although adversarial examples were shown initially in the computer vision domain (Szegedy et al., 2014; Goodfellow, Shlens and Szegedy, 2015; Madry et al., 2018; Biggio and Roli, 2018), malware is no exception. Recent attacks have crafted adversarial examples in the malware domain—so-called *adversarial malware*; now, a carefully crafted malware sample with minimal changes to malware code but still able to preserve the *realism* and *functionality* of the malware is able to fool ML-based malware detectors to misclassify them as benign-ware. These attacks pose an emerging threat against ML-based malware detectors (Grosse et al., 2017; Kolosnjaji et al., 2018; Kreuk et al., 2018; Suciu, Coull and Johns, 2019; Pierazzi et al., 2020; Demetrio et al., 2021).

Problem. In general, adversarial learning (Athalye, Carlini and Wagner, 2018) or training with adversarial examples is an effective method to build models robust against adversarial examples. However, generating adversarial malware samples for training, especially at the *production scale* necessary for deployable models, is non-trivial. Because:

- Generation of adversarial examples in the malware domain is confronted with the *inverse feature-mapping problem* where the function mapping from the *problem space* (the discrete space of software code binaries) to the feature space (vectorized features) is non-differentiable (Biggio et al., 2013; Biggio, Fumera and Roli, 2013; Quiring, Maier and Rieck, 2019). Hence, *fast*, gradient-driven methods to derive

useful information to craft adversarial samples in the problem space are not suitable.

- The need to enforce malware domain constraints, *realism*, *functionality* and *maliciousness*, on generated perturbations in the problem space is a difficult proposition. Thus, arbitrary changes to the malware binaries are not possible because it could drastically alter the malware in a manner to break the malicious functionality of the binaries or even make it unloadable.

Although efforts to realise robust models on discrete spaces such as discrete images or graph data exist (Lee et al., 2019; Wang et al., 2021), the problem space of malware classification is significantly more challenging due to the imposed constraints in the problem space; the *realism* and *functionality* as well as *maliciousness* of the malware must be maintained. Unfortunately, a method to scale-up adversarial training with samples in the problem space to production scale datasets, especially in the case of neural networks, does not exist.

Further, despite extensive work on adversarial ML in general, very few studies have focused on the problem in the context of malware as recently highlighted by Pierazzi et al. (2020), and a comprehensive investigation of robust defence methods in the area remains to be conducted.

Research Questions. Hence, in this study, we seek to answer the following research questions (RQs):

- **RQ1.** How can we overcome the challenging problem of adversarial learning for malware at a *production scale* to realise robust malware detectors against adversarial malware samples?
- **RQ2.** How can we formulate an adversarial learning problem for building robust malware detectors and how can we *explain* the robustness and *benefits*?
- **RQ3.** How robust are adversarially trained malware detectors, especially against problem-space (functional, realistic and malicious) adversarial malware samples?

Our Approach. We argue that a defender is not confronted with the problems we mentioned. Because, we show that constraining the adversarial examples in the

3.1 Motivation and Contribution

problem space to *preserve malware realism, functionality and maliciousness can be turned to an advantage for defenders. The constraints make the perturbed malware in the problem space a subset of the adversarial examples in the feature space. Therefore, designing a robust method against feature-space adversarial examples will inherently be robust against constrained problem-space adversarial examples encapsulating the threats from adversarial malware.*

To construct a formulation to improve the robustness against feature-space adversarial malware examples, and ultimately problem-space malware, we propose a Bayesian formulation for adversarially training a neural network: i) with the capability to capture the distribution of models to improve robustness (Liu and Wang, 2016; Liu et al., 2019b; Ye and Zhu, 2018; Wicker et al., 2021; Carbone et al., 2020); and ii) prove our proposed method of diversified Bayesian neural networks hardened with adversarial training bounds the difference between the adversarial risk and the conventional empirical risk to *theoretically explain* the improved robustness.

Moreover, just recently, security researchers placed significant effort into providing extracted, continuous features for malware samples at a production scale of more than 20 million samples (Harang and Rudd, 2020; Anderson and Roth, 2018)–the SOREL-20M dataset. However, *the robustness of networks built on these extracted features in the face of evasion attacks are yet to be understood.* Therefore, our study to investigate production scale adversarial learning is timely and we focus our efforts to investigate methods using the SOREL-20M dataset.

Our Contributions. In our efforts to address the problem of building robust malware detectors, we make the following contributions:

1. We *prove* the projection of perturbed yet, valid malware, in the problem space (the discrete space of software code binaries) into the feature space will be a subset of feature-space adversarial examples. Thus, a robust network against feature-space attacks is inherently robust against problem-space attacks. Our work provides *a theoretically justified basis for adversarially training malware detectors in the feature space.* Further, to corroborate our proof, we empirically demonstrate networks trained on feature-space adversarials are robust against functional and realistic problem-space adversarial malware (**RQ1**).
2. Hence, to improve robustness in the *problem space* we propose performing adversarial learning in the *feature space* and formulate a Bayesian Neural Network (BNN) adversarial learning objective that captures the distribution of models for

improved robustness. The algorithm is capable of learning from production scale feature-space datasets of up to *20 million samples* (*RQ1* and *RQ2*).

3. We also *prove* hardening BNNs with adversarial examples bounds the difference between the adversarial risk and the empirical risk to explain the improved robustness. Further, we show the benefit of adversarially trained BNNs to capture and estimate the uncertainty in malware predictions for a defence method (*RQ2*).
4. We empirically demonstrate Bayesian Neural Networks capturing model diversity to improve the performance of malware classifiers and adversarially trained BNNs to generate more robust models against the threat of adversarial malware. Adversarially trained BNNs achieve new benchmarks for state-of-the-art robustness—especially against unseen, stronger, attack samples (*RQ3*).

Scope. Notably, in our study, we focus on Windows Portable Executable (PE) malware for two reasons: i) Windows is the most popular operating system for end-users worldwide, and PE-file malware is the earliest and most studied threat in the wild (Schultz et al., 2000), making a robust method to detect adversarial PE files a significant contribution to security research; and ii) the intuition and methodology behind Windows PE malware can be applied and transferred to other file formats and operating systems, such as PDF malware or malware for Linux and Android systems.

3.2 Problem Definition

Threat model. We assume an attacker with *perfect knowledge* (white-box attacker) (Biggio, Fumera and Roli, 2014, 2013), in which the attacker knows *all* parameters including feature set, learning algorithm, loss function, model parameters/hyperparameters, and training data. The reason for considering the strongest, perfect-knowledge adversary is because, even if access to the model is not possible, or the model is not publicly available, an adversary can employ a reverse engineering approach such as (Tramèr et al., 2016; Rolnick and Kording, 2020; Carlini, Jagielski and Mironov, 2020) to extract the model. And, defending against such attacks is challenging. The attacker objective is to *evade detection*. Their capability is to modify the features at test time.

3.2 Problem Definition

Problem-Space Attacks. We consider the **problem space** \mathcal{Z} which refers to the *input space* of real objects of a considered domain such as software code binaries—our focus in the paper; first \mathcal{Z} must be transformed into a compatible format such as numerical vector data (Anderson and Roth, 2018; Harang and Rudd, 2020) for ML to process. Then, a **feature mapping** is a function $\Phi : \mathcal{Z} \rightarrow \mathcal{X} \subseteq \mathbb{R}^n$ that maps a given problem-space software code binary $\mathbf{z} \in \mathcal{Z}$ to an n -dimensional feature vector $\mathbf{x} \in \mathcal{X}$ in the **feature space** such that $\Phi(\mathbf{z}) = \mathbf{x}$.

Normally, attackers have to apply a transformation to \mathbf{z} to generate \mathbf{z}' such that $\Phi(\mathbf{z}')$ is very close to \mathbf{x}' in the feature space. Formally, given a problem-space object $\mathbf{z} \in \mathcal{Z}$ with label $y \in \mathcal{Y}$, the goal of the adversary is to find the *transformation* $\mathbf{T} : \mathcal{Z} \rightarrow \mathcal{Z}$ (e.g. addition, removal, modification) such that $\mathbf{z}' = \mathbf{T}(\mathbf{z})$ is classified as a class $t \neq y$. In the malware domain, the adversary has to search in the problem space that approximately follows the gradient in the feature space. However, this is a major challenge that complicates the application of gradient-driven methods to the problem-space attacks—so-called **inverse feature-mapping problem** (Quiring, Maier and Rieck, 2019; Biggio et al., 2013; Pierazzi et al., 2020) where the function Φ in the software domain—our focus—is typically not invertible and not differentiable, *i.e.* there is no one-to-one mapping from the adversarial examples in the feature space $\mathbf{x} + \delta$ to the corresponding adversarial problem-space object \mathbf{z}' . In addition, the generated object $\mathbf{T}(\mathbf{z})$ must be realistic and valid (Suciu, Coull and Johns, 2019). Thus, the search for adversarial examples in the problem space (software) cannot be a purely gradient-based method, hindering the adoption of well-known adversarial attacks in other domains such as computer vision. To achieve a realistic adversarial objective, the search for adversarial examples in the problem space has to be constrained in **problem-space constraints** denoted by Ω . We remark that the constraints on the problem-space are well defined and can be found in (Biggio and Roli, 2018; Biggio et al., 2013; Quiring, Maier and Rieck, 2019; Xu, Qi and Evans, 2016; Pierazzi et al., 2020), we mentioned here for completeness there are at least four main types of problem-space constraints including Preserved semantics, Plausibility, Robustness to Processing and Available Transformation explained in detail by Pierazzi et al. (2020).

Feature-Space Attacks. To alleviate the problems with problem space attacks, we propose an alternative that uses feature space. We note that all definitions of feature-space attacks are well defined and consolidated in related work (Biggio and Roli, 2018; Carlini and Wagner, 2017b; Grosse et al., 2017; Szegedy et al., 2014). In

this paper, we use a popular feature mapping function provided in the EMBER dataset (Anderson and Roth, 2018) to map raw bytes of software to a vector of $n = 2381$ features. A **feature-space attack** is then to modify a feature-space object $\mathbf{x} \in \mathcal{X}$ to become another object $\mathbf{x}' = \mathbf{x} + \delta$ where δ is the added perturbation crafted with an *attack objective function* to misclassify \mathbf{x}' into another class $t \neq y$ where $y \in \mathcal{Y}$ is the ground-truth label of \mathbf{x} . We note that in the malware domain (a binary classification task), the intuition of the attackers is to make the malware be recognized as benign ware. These modifications has to follow **feature-space constraints**. We denote the constraints on feature-space modifications by Ψ . Given a sample $\mathbf{x} \in \mathcal{X}$, the feature-space modification, or perturbation δ must satisfy Ψ . This constraint Ψ reflects the realistic requirements of problem-space objects. In the malware domain, feature perturbations δ can be constrained $\delta_{lb} \leq \delta \leq \delta_{ub}$ (Pierazzi et al., 2020).

3.3 Theoretical Basis For Feature-Space Adversarial Learning

In this paper, we highlight that the realistic assumption of problem-space attacks makes the constraints imposed by Ω stricter or equal to those imposed by Ψ (illustrated in Figure 3.1). Following the necessary condition for problem-space adversarial examples as stated in Pierazzi et al. (2020), we have:

Lemma 1. If there exists an adversarial example in the problem space (\mathbf{z}') that satisfies the constraints Ω , then there will be a corresponding adversarial example in the feature space (\mathbf{x}') under the constraints Ψ . More formally, by abusing notation from model theory to use \models to indicate an instance “satisfies” constraints, and write $\mathbf{z}' \models \Omega$ and $\mathbf{x}' \models \Psi$, we have:

$$\begin{aligned} \exists \mathbf{z}' : \mathbf{z}' \models \Omega, \quad p(y | \Phi(\mathbf{z}'), \theta) &= p(y | \Phi(\mathbf{T}(\mathbf{z})), \theta), \\ & p(y | \Phi(\mathbf{T}(\mathbf{z})), \theta) < 0.5 \\ \Rightarrow \exists \mathbf{x}' = \mathbf{x} + \delta : \mathbf{x}' \models \Psi, \quad p(y | \mathbf{x}', \theta) &< 0.5 \end{aligned}$$

where \mathbf{T} is the transformation in the problem space to craft adversarial examples, $p(y | \mathbf{x}, \theta) = \text{sigmoid}(f(\mathbf{x}; \theta))$ is the output of a sigmoid function applied to the output of the neural networks f parameterised by θ , $p(y | \mathbf{x}, \theta) = 0.5$ is the threshold for malware detection where the predicting $p(y | \mathbf{x}, \theta) = 0$ is recognized as benign

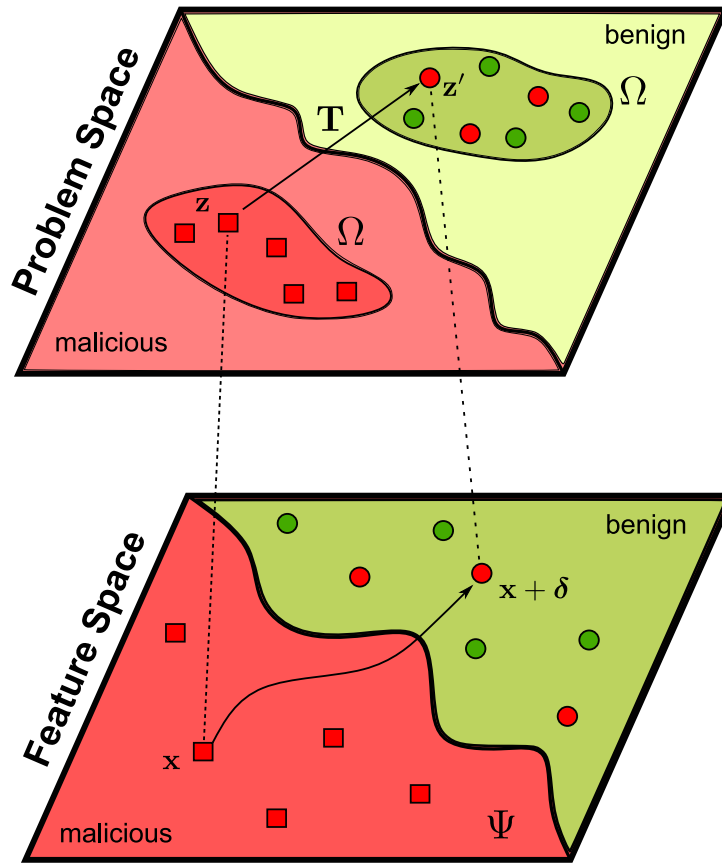


Figure 3.1: Illustrative example of adversarial examples. The adversarial example $\mathbf{x} + \delta$ is derived from \mathbf{x} in the feature space and its projection to problem-space constraints (which is more restrictive) determined by Ω is \mathbf{z}' . The colour in the background illustrates the decision regions where red colour is for malware and green is for benign programs. The solid arrow in Feature Space represents the gradient-based attack to transform a malware \mathbf{x} to $\mathbf{x} + \delta$, projected to the problem-space constraints as \mathbf{z}' to be misdetected as a benign program.

whilst $p(y | \mathbf{x}, \theta) = 1$ indicates a malware, Ω, Ψ are, respectively, the problem-space and feature-space constraints, and $\Phi(\cdot)$ is the function that maps the problem space to feature space.

Proof of Lemma 1

We consider the problem-space example $\mathbf{z} \in \mathcal{Z}$ with the feature in the feature space $\mathbf{x} \in \mathcal{X}$.

Initially, we assume that there is no solution to the feature-space attack (\mathbf{x}') under constraints Ψ given an existing problem-space adversarial attack (\mathbf{z}') under constraints Ω . Specifically, we assume that there is no δ that satisfies $p(y | \mathbf{x} + \delta, \theta) < 0.5, \delta \models \Psi$.

However, because there exists an adversarial attack in the problem space,

$$\exists \mathbf{z}' : \mathbf{z}' \models \Omega, \quad p(y | \Phi(\mathbf{z}'), \theta) = p(y | \Phi(\mathbf{T}(\mathbf{z})), \theta), \quad p(y | \Phi(\mathbf{T}(\mathbf{z})), \theta) < 0.5 \quad (3.1)$$

Thus,

$$\exists \mathbf{T}^* : p(y | \Phi(\mathbf{z}'), \theta) = p(y | \Phi(\mathbf{T}^*(\mathbf{z})), \theta), p(y | \Phi(\mathbf{T}^*(\mathbf{z})), \theta) < 0.5, \quad \mathbf{z}, \mathbf{z}' \models \Omega \quad (3.2)$$

For any transformation \mathbf{T} in the input space, there exists a value δ^* for which we have:

$$\exists \mathbf{T}^* : p(y | \Phi(\mathbf{T}^*(\mathbf{z})), \theta) = p(y | \mathbf{x} + \delta^*, \theta), \quad p(y | \mathbf{x} + \delta^*, \theta) < 0.5, \quad \mathbf{z}, \mathbf{z}' \models \Omega \quad (3.3)$$

We recall that feature-space constraints are determined by problem-space constraints, that is, the search space allowed by Ω is stricter or equal to that allowed by Ψ . Thus, δ^* must be a solution to feature-space constraints Ψ . However, this is not possible because we begin with the assumption that there is no δ that satisfies $f(\mathbf{x} + \delta) < 0.5, \delta \models \Psi$.

This contradiction proves that:

$$\begin{aligned} \exists \mathbf{z}' : \mathbf{z}' \models \Omega, \quad p(y | \Phi(\mathbf{z}'), \theta) = p(y | \Phi(\mathbf{T}(\mathbf{z})), \theta), \quad p(y | \Phi(\mathbf{T}(\mathbf{z})), \theta) < 0.5 \\ \Rightarrow \exists \mathbf{x}' = \mathbf{x} + \delta : \mathbf{x}' \models \Psi, p(y | \mathbf{x}', \theta) < 0.5 \end{aligned}$$

From **Lemma 1**, for an existing problem-space adversarial attack, we have a corresponding feature-space adversarial attack. Thus, we can derive:

Corollary 1. The adversarial examples generated from constrained problem-space adversarial examples (imposed by Ω) are in a subset of feature-space adversarial examples (imposed by Ψ).

Corollary 2. Detectors robust against feature-space adversarial examples (imposed by Ψ) are robust against constrained problem-space adversarial examples (imposed by Ω).

Built upon these Corollaries, we propose to find a learning method robust against *feature-space adversarial malware*. On the one hand, adversarial training (Madry et al., 2018) and its variants are shown to be one of the most effective and popular methods to defend against adversarial examples (Athalye, Carlini and Wagner, 2018). On the other hand, Bayesian neural networks (MacKay, 1992; Ritter, Botev and Barber, 2018; Izmailov et al., 2021) with distributions placed over their weights and biases enabling

3.4 Methods

the principled quantification of the uncertainty of their predictions are shown to be a robust method against adversarial examples. Thus, this chapter demonstrates that robustness against feature-space adversarial examples is inherently robust against problem-space real malware. We propose to incorporate adversarial training with Bayesian neural networks to seek the first principled method of Bayesian adversarial learning to realise a robust malware detector without the difficulties of inverse feature-mapping and preserving the semantics and functionalities of real malware samples. We name our method Adv. MalBayes, and the method is efficient enough to be scaled up to a large production scale of adversarial training data of 20 million adversarial samples with the pre-extracted feature set of SOREL-20M dataset (Harang and Rudd, 2020).

3.4 Methods

Our method combines adversarial training with a Bayesian inference approach to faithfully capture the posterior distribution of parameters that can provably bound the difference between adversarial and empirical risk commensurate with a robust adversarial defence. The following subsections describe our formulation.

3.4.1 Bayesian Formulation for Adversarial Learning

The goal of Bayesian adversarial learning is to find the posterior distribution using the Bayes theorem:

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{adv}}) = \prod_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} p(y \mid \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) / Z$$

where Z is the normaliser, \mathcal{D}_{adv} is the adversarial dataset obtained by generating adversarial examples from the benign dataset D using adversarial generation such as Eq. (3.4).

We consider $p(y \mid \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}) = \text{sigmoid}(f(\mathbf{x}_{\text{adv}}; \boldsymbol{\theta}))$ to produce a binary prediction in malware detection, where f is a neural network. Notably, Eq. (3.4) is the Expectation-over-Transformation (EoT) PGD attack (Athalye et al., 2018; Zimmermann, 2019), which is slightly different from the usual PGD attack (Madry et al., 2018). As has been highlighted in Zimmermann (2019), the EoT attack is better able to estimate the gradient of the stochastic Bayesian models:

$$\mathbf{x}^{t+1} = \Pi_{\epsilon_{\text{max}}} \left\{ \mathbf{x}^t + \alpha \cdot \text{sign} \left(\mathbb{E}_{\boldsymbol{\theta}} \left[\nabla_{\mathbf{x}} \ell \left(f \left(\mathbf{x}^t; \boldsymbol{\theta} \right), y_o \right) \right] \right) \right\}. \quad (3.4)$$

where ε_{\max} is the maximum attack budget, $\Pi_{\varepsilon_{\max}}$ is the projection to the set $\{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_0\|_{\infty} \leq \varepsilon_{\max}\}$, ℓ is the loss function (typically cross-entropy), f is the neural network, \mathbf{x} is the input, θ is the network parameter, and y is the ground-truth label. In this attack, an attacker starts from $\mathbf{x}^0 = \mathbf{x}_0$ and conducts projected gradient descent iteratively to update the adversarial example.

However, as highlighted in [Izmailov et al. \(2021\)](#), the posterior over a Bayesian neural network is extremely high-dimensional, non-convex and intractable. Thus, we need to resort to approximations to find the posterior distribution. In this work, we propose using Stein Variational Gradient Descent (SVGD) ([Liu and Wang, 2016](#)) for two reasons. First, this approach learns multiple *network parameter particles* in parallel for faster convergence. Second, there is a *repulsive factor* in the method to encourage the diversity of parameter particles that helps to prevent mode collapse — a challenge of posterior approximation. To further demonstrate the robustness of our chosen Bayesian method, we compare Adv. MalBayes with previous BNNs ([Liu et al., 2019b](#)) in Section 3.5.5 Table 3.8.

We consider n samples from the posterior (*i.e.* parameter particles). The variational bound is minimised when gradient descent is modified as:

$$\theta_i = \theta_i - \varepsilon_i \hat{\phi}^*(\theta_i)$$

$$\text{with } \hat{\phi}^*(\theta) = \sum_{j=1}^n [k(\theta_j, \theta) \nabla_{\theta_j} \ell(f(\mathbf{x}_{\text{adv}}; \theta_j), y) - \frac{\gamma}{n} \nabla_{\theta_j} k(\theta_j, \theta)].$$

Here, θ_i is the i th particle, $k(\cdot, \cdot)$ is a kernel function that measures the similarity between particles and γ is a hyper-parameter. The parameter particles are encouraged to be dissimilar to capture more diverse samples from the posterior thanks to the kernel function. This is controlled by a hyper-parameter γ , to manage the trade-off between diversity and loss minimisation. Here, following ([Liu and Wang, 2016](#)), we use the RBF kernel $k(\theta, \theta') = \exp(-\|\theta - \theta'\|^2 / 2h^2)$ and take the bandwidth h to be the median of the pairwise distances of the set of parameter particles at each training iteration.

At the inference stage, given the test data point \mathbf{x}^* , we can get the prediction by approximating the posterior using the Monte Carlo samples as:

$$p(y^* | \mathbf{x}^*, \mathcal{D}_{\text{adv}}) = \int p(y^* | \mathbf{x}^*, \theta) p(\theta | \mathcal{D}_{\text{adv}}) d\theta$$

$$\approx \frac{1}{n} \sum_{i=1}^n p(y^* | \mathbf{x}^*, \theta_i), \quad \theta_i \sim p(\theta | \mathcal{D}_{\text{adv}}),$$

3.4.2 Adversarial Risk Bounded with Bayesian Formulation

where θ_i is an individual parameter particle.

In addition, we also acknowledge that it is critical to have diverse parameter particles. Averaging over diverse and uncorrelated predictors was shown to improve network performance (Jacobs et al., 1991; Wolpert, 1992; Breiman, 1996). In the adversarial setting, when integrating out the parameters in our Bayesian formulation, we implicitly remove the vulnerabilities arising from a single choice of parameter existing in traditional neural networks, and hence improve the robustness. Furthermore, the uncertainty provided by a diverse Bayesian neural network is also beneficial to detect unseen strong adversarial malware as discussed in Section 3.6.

Algorithm 3.1 Bayesian Adversarial Learning with SVGD

- 1: **Input:** A set of initial parameter particles $\{\theta_i^0\}_{i=1}^n$, observation feature-space data \mathcal{D} .
- 2: **Output:** A set of parameter particles $\Theta := \{\theta_i\}_{i=1}^n$ that approximates the true posterior distribution $p(\theta \mid \mathcal{D}_{\text{adv}})$
- 3: **for** $(x, y) \sim p(\mathcal{D})$ **do**
- 4: $\mathbf{x}_{\text{adv}} \leftarrow \mathbf{x}$
- 5: **for** $t = 1 \rightarrow T$ **do** ▷ Generate feature-space adversarial examples (Eq. (3.4))
- 6: $\mathbf{x}_{\text{adv}} = \Pi_{\epsilon_{\text{max}}} \{ \mathbf{x}_{\text{adv}} + \alpha \cdot \text{sign} (\mathbb{E}_{\theta} [\nabla_{\mathbf{x}} \ell (f (\mathbf{x}_{\text{adv}}; \theta_j), y)]) \}$
- 7: **for** $i = 1 \rightarrow n$ **do**
- 8: $\theta_i \leftarrow \theta_i - \epsilon_i \hat{\phi}^*(\theta_i, \theta_j)$ with

$$\hat{\phi}^*(\theta_i, \theta_j) = \sum_{j=1}^n [k(\theta_j, \theta_i) \nabla_{\theta_j} \ell(\Theta) - \frac{\gamma}{n} \nabla_{\theta_j} k(\theta_j, \theta_i)]$$

- 9: ϵ_i is the step size at the current iteration, $k(\theta, \theta')$ is a positive definite kernel that specifies the similarity between θ and θ' , ℓ is binary cross-entropy loss, and γ is the weight to control the *repulsive force* that enforces the diversity among parameter particles.
-

3.4.2 Adversarial Risk Bounded with Bayesian Formulation

In this section, to explain the robustness of the Bayesian adversarial learning method that we propose, we prove that training the network with the Bayesian adversarial learning method bounds the difference between the adversarial risk and the empirical

risk. This is important, because, now the risk of misclassification on adversarial examples is as the same as that of benign ones; hence eliminating the vulnerability of adversarial examples and reduce the risk of misclassification of adversarial examples to the generalisation ability of the classifier. Notably, improving the generalisation ability of the classifier is not our focus.

In this context, we make no specific assumption on the distribution of either the adversarial examples or the perturbations, to provide a generic defence approach. The only assumption we make is that the distribution of the data and the corresponding adversarial examples are sufficiently close. This is a mild and reasonable assumption because the idea of adversarial learning is that the added perturbation does not change the perceived samples or the distribution of the samples. Thus, we consider the bound of $|R_{\text{adv}} - R|$ where the empirical risk $R = \mathbb{E}_{\theta} \left[\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{y' \sim p(y|\mathbf{x}, \theta)} [\mathbb{I}(y = y')] \right] \right]$ and the adversarial risk $R_{\text{adv}} = \mathbb{E}_{\theta} \left[\mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} \left[\mathbb{E}_{y' \sim p(y|\mathbf{x}_{\text{adv}}, \theta)} [\mathbb{I}(y = y')] \right] \right]$

Proposition 3.1. *The difference between the adversarial risk (denoted by R_{adv}) and the empirical risk (denoted by R) of a classifier when trained on the observed training set and its adversarial counterparts is bounded, i.e.*

$$|R_{\text{adv}} - R| \leq \tau,$$

$$\text{where } \tau = 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\exp \left(\mathbb{E}_{\theta} [r_{\theta}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)] \right) \right],$$

$$r_{\theta}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y) = \sum_c^K p(y = c | \mathbf{x}, \theta) \log(p(y = c | \mathbf{x}_{\text{adv}}, \theta)).$$

Here, \mathbf{x}_{adv} denotes the adversarial example obtained from \mathbf{x} .

We can see that the difference between the empirical risk and the adversarial risk is minimised when the upper bound is minimised. Notably, as we know that $1 - \exp(-z)$ is a monotonically increasing function, and $1 - \exp(-z) \leq z$, to avoid computational instabilities and gradient saturation, we consider minimising the upper bound without the exponential function. Thus, to minimise the upper bound, our main learning objective (in Algorithm 3.1 is to:

Minimise cross entropy for the adversarial examples. This corresponds to matching the prediction from the adversarial data to that of the observations. Since (\mathbf{x}, y) is given in the training, we simply minimise the entropy of the adversarial examples.

Bayesian learning bounds the difference between empirical risk and adversarial risk. This bound is minimised when hardening a BNN with adversarial examples.

Proof of the Objective (Proposition 1)

We have

$$\begin{aligned}
 |R_{\text{adv}} - R| &= \left| \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup_{\mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}})} [\mathbb{I}(y_1 \neq y)] - \mathbb{E}_{y_2 \sim p(y|\mathbf{x})} [\mathbb{I}(y_2 \neq y)] \right] \right] \right|, \\
 &= \left| \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup_{\mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}}), y_2 \sim p(y|\mathbf{x})} [\mathbb{I}(y_1 \neq y) - \mathbb{I}(y_2 \neq y)] \right] \right] \right|, \\
 &\leq \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup_{\mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}}), y_2 \sim p(y|\mathbf{x})} [|\mathbb{I}(y_1 \neq y) - \mathbb{I}(y_2 \neq y)|] \right] \right], \\
 &\leq \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup_{\mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}}), y_2 \sim p(y|\mathbf{x})} [\mathbb{I}(y_1 \neq y_2)] \right] \right].
 \end{aligned}$$

where we can upper bound the expected misclassification to arrive at:

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[1 - \sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{\text{adv}}, \theta) \right] \right].$$

Subsequently, we can use Jensen's inequality and the fact that $\mathbf{x} = \exp(\log(\mathbf{x}))$ to engender:

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[1 - \underbrace{\exp\left(\log\left(\sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{\text{adv}}, \theta)\right)\right)}_{\geq \sum_c^K p(y=c|\mathbf{x}, \theta) \log(p(y=c|\mathbf{x}_{\text{adv}}, \theta))} \right] \right],$$

and because $1 - \exp(z)$ is monotonically decreasing, we have

$$\begin{aligned}
 &\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[1 - \exp\left(\log\left(\sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{\text{adv}}, \theta)\right)\right) \right] \right] \\
 &\leq \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[1 - \exp\left(\sum_c^K p(y = c | \mathbf{x}, \theta) \log(p(y = c | \mathbf{x}_{\text{adv}}, \theta))\right) \right] \right] \\
 &= 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\exp\left(\sum_c^K p(y = c | \mathbf{x}, \theta) \log(p(y = c | \mathbf{x}_{\text{adv}}, \theta))\right) \right] \right].
 \end{aligned}$$

This produces the following bound:

$$|R_{\text{adv}} - R| \leq 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\exp\left(\underbrace{\mathbb{E}_{\theta} \left[\sum_c^K p(y = c | \mathbf{x}, \theta) \log(p(y = c | \mathbf{x}_{\text{adv}}, \theta)) \right]}_{r_{\theta}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)}\right) \right].$$

This result demonstrates that the difference between the risks is bounded by the negative cross-entropy of the predictions. Although informative, this bound expresses the relationship between the predictions but not how the model performs on each set (*i.e.* a given dataset against its corresponding adversarial).

Then, given the difference between empirical risk and adversarial risk is minimised when the upper bound is minimised, the main learning objective is to:

Maximise $\mathbb{E}_{\theta}[\sum_c^K p(y = c | \mathbf{x}, \theta) \log(p(y = c | \mathbf{x}_{adv}, \theta))]$ or minimise $\mathbb{E}_{\theta}[\text{KL}(p(y = c | \mathbf{x}, \theta) || p(y = c | \mathbf{x}_{adv}, \theta))]$. This corresponds to matching the prediction from the adversarial data to that of the observations. Because (\mathbf{x}, y) is given in training, minimising this KL-divergence requires simply minimising the entropy of the adversarial examples instead.

Notably, given we know $1 - \exp(-z) \leq z$, to avoid computational instabilities and gradient saturation, our implementation minimises the upper bound without the exponential function.

Sketch of the Proof. We simplify the difference between risks by considering that the difference between individual mistakes is smaller than their product, *i.e.*

$$\begin{aligned} & \mathbb{E}_{y_1 \sim p(y|\mathbf{x},\theta)} \left[\mathbb{E}_{y_2 \sim p(y|\mathbf{x}_{adv},\theta)} [\mathbb{I}[y \neq y_1] - \mathbb{I}[y \neq y_2]] \right] \\ & \leq \mathbb{E}_{y' \sim p(y|\mathbf{x}_{adv},\theta)} \left[\mathbb{E}_{y'' \sim p(y|\mathbf{x}_{adv},\theta)} [\mathbb{I}[y_1 \neq y_2]] \right] \\ & \leq 1 - \sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{adv}, \theta). \end{aligned}$$

We then use Jensen's inequality when using $\exp(\log(\cdot))$ to obtain the upper bound. We empirically evaluate this difference of risk, presenting the results in Figure 3.2.

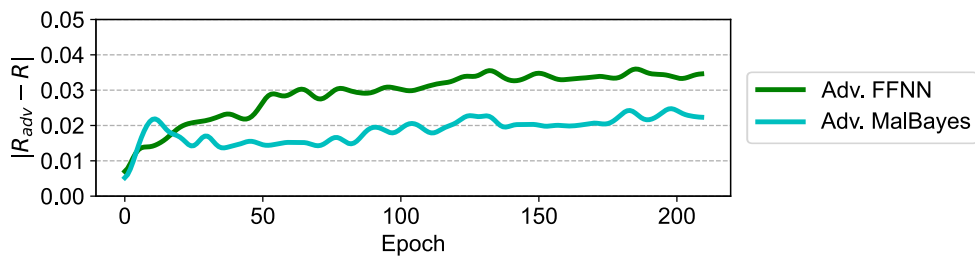


Figure 3.2: The difference between conventional empirical risk and adversarial risk $|R_{adv} - R|$ for the EMBER test set.

3.5 Experiments

Classifiers. To validate our proposed method Adv. MalBayes, we conduct experiments on different neural networks. We employ the Feed Forward Neural Network (FFNN) classifier provided in the SOREL-20M dataset (Harang and Rudd, 2020). This network architecture is also used for the experiments on the EMBER dataset (Anderson and Roth, 2018). Our network implementation uses the default configuration provided in (Harang and Rudd, 2020). We also adopt the architecture of FFNN to design the Bayesian Neural Network (BNN). The details of the network architecture are in Section 3.5.1. Then, we harden the FFNN and BNN with adversarial examples to generate the Adv. FFNN model and Adv. MalBayes. In addition, we also employ baseline networks including LightGBM (Anderson and Roth, 2018) and MalConv (Raff et al., 2018) for comparison. We compare their performance on malware datasets (no attacks) and their adversarial counterparts (adversarial malware designed to evade detectors) to evaluate the *detector performance* and *robustness*. The values of the attack budgets used for training and testing are detailed in Table 3.1.

Datasets. This chapter uses the two largest publicly available corpora for malware detection, namely, the Sophos AI SOREL-20M (Harang and Rudd, 2020) and EMBER2018 (Anderson and Roth, 2018) datasets.

- The **EMBER** dataset contains portable executable files (PE files) scanned by VirusTotal on or before 2018. It includes 600,000 training samples and 200,000 testing samples, equally distributed between benign programs and malicious programs, where the malware is labelled by the malware family using AVClass (Sebastián et al., 2016). This dataset also includes the vectorised features that encode vast amounts of information regarding the PE files, including general file information, import/export functions, header information and string information. This EMBER dataset was designed to be *more challenging* for ML-based classifiers compared with the older version — the EMBER2017 dataset. Thus, using this dataset is of interest in terms of evaluating our ML-based malware classifier.
- The **SOREL-20M** dataset is a recent industrial-scale dataset provided by Sophos AI (Harang and Rudd, 2020). It contains 20 million samples featuring pre-extracted features and metadata and high-quality labels. More concretely, the dataset includes 12,699,013 training samples, 2,495,822 validation samples and 4,195,042 testing samples, as well as 10 million disarmed malware samples for feature exploration.

Table 3.1: Hyper-parameter settings used in experiments.

<i>Name</i>	<i>Value</i>	<i>Notes</i>
T	20	#PGD iterations
ε_{\max}	0.03	Max l_{∞} -norm in adversarial training
α	0.02	Step size for each PGD iteration
γ	SOREL-20M:0.01, EMBER:0.05	Weight to control the repulsive force
		#Parameter particles
n	SOREL-20M:5, EMBER:20	#Forward passes when doing ensemble inference # Expectation over Transformation

Table 3.2: Model Architecture for SOREL-20M and EMBER2018 datasets

Layer Type	# of Channels	Drop out	Activation
FC	512	0.05	ELU
FC	512	0.05	ELU
FC	128	0.05	ELU
FC	1	-	Sigmoid

By including a significant amount of high-quality samples, this dataset aims to represent a new benchmark for malware detection.

Note that both datasets follow a strict temporal split policy, with test samples strictly observed after training samples.

3.5.1 Network Architecture

We use the network architecture provided in the SOREL-20M dataset (Harang and Rudd, 2020) as the baseline for our BNN versions. Notably, because no baseline deep neural networks are provided in the EMBER2018 dataset (Anderson and Roth, 2018), we adopt the same baseline network architecture (Table 3.2) for that dataset.

3.5.2 Experimental Results

Table 3.3: Comparing the performance of BNN and FFNN on different datasets (at 1%FPR)

Dataset	Network	Detection Rate	Acc	AUC
SOREL-20M	FFNN	97.74	98.71	99.77
	BNN	97.95	98.76	99.83
EMBER	MalConv (Raff et al., 2018)	-	90.88	97.05
	MalConv w/ GCG (Raff et al., 2021)	-	93.29	98.04
	FFNN	84.32	94.11	98.47
	BNN	86.43	94.50	98.55

3.5.2 Experimental Results

We present our results by reporting: i) performance of the given classifiers on malware detection tasks (no attacks setting) using ROC (receiver operating characteristic curve); and ii) robustness (under evasion attacks with adversarial malware).

Performance (no attacks). Performance of the classifiers in the absence of attacks are shown in Figure 3.3 with additional details reported Table 3.3. The ROC curves in Figure 3.3 report the True Positive Rate (*i.e.* the percentage of correctly-classified malware samples) as a function of the False Positive Rate (FPR, *i.e.* the percentage of misclassified benign samples) for each classifier. From the figure, we can see that Bayesian neural networks of the same network architecture as FFNNs achieve better performance (compare BNN vs. FFNN and Adv. MalBayes vs. Adv-FFNN). Notably, the BNNs outperformed the FFNN counterparts with a large margin in the detection rate (of up to 20%) under low-FPR regimes. Notably, in Table 3.3, we also show that BNNs built on feature-space samples achieve better performance compared with the popular ML-based malware detector built on problem-space samples (MalConv) (Raff et al., 2018) and its recently updated version in AAI'21 (MalConv w/ GCG) (Raff et al., 2021). We can also see that BNNs achieve comparable or slightly better performance compared with FFNN in the absence of attack inputs (adversarial malware).

BNNs outperform FFNNs in benign conditions, with large margins observed for low-FPR regimes.

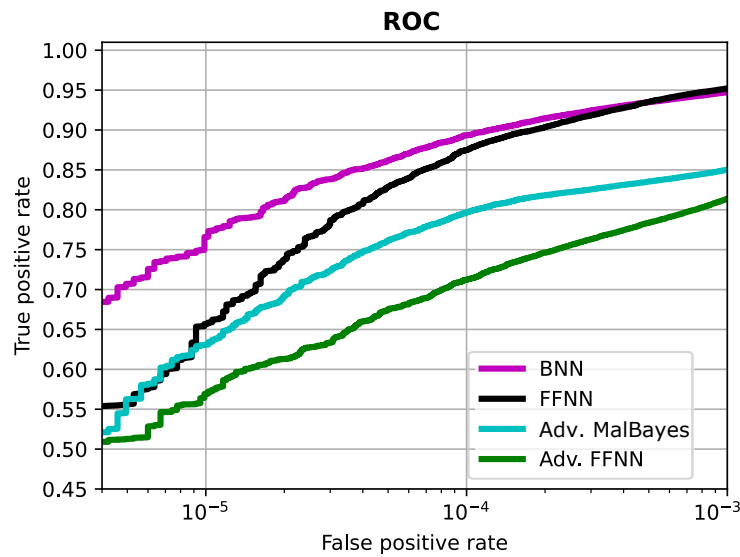


Figure 3.3: Performance of neural network detectors in the absence of adversarial attacks in the SOREL-20M dataset. The Receiver Operating Characteristics depict the detection ability of the models as their discrimination threshold varies. We can observe BNN models to outperform their FFNN counterparts.

Robustness (against Feature-Space Adversarial Examples. To evaluate the robustness of the investigated classifiers, we apply the PGD attack from Equation (3.4) on malware samples with increasing attack budgets. Results for the robustness of given classifiers under different attack budgets are reported in Table 3.4. Notably, Adv. MalBayes outperforms the adversarially trained FFNN on both the production scale (SOREL-20M) and challenging (EMBER) datasets, especially under increasing attack budgets. This is significant because the problem of malware is that they are evolving extremely fast *e.g.* there are hundreds of thousands of new malware samples every day (KasperskyLab, 2020). Further, results in Figure 3.4 illustrate, as expected and in line with the findings in the literature (Madry et al., 2018; Carlini and Wagner, 2017b; Goodfellow, Shlens and Szegedy, 2015), the adversarially trained networks are significantly more robust than their non-adversarially trained counterparts. The robustness achieved against strong, unseen, feature-space adversarial attacks, potentially, demonstrate robustness Adv. MalBayes against problem-space adversarial malware; we will investigate robustness against problem-space adversarial malware in the following section.

Adversarially-trained BNNs significantly outperform their FFNN counterparts against strong unseen adversarial attacks.

3.5.2 Experimental Results

Table 3.4: Robustness of networks against adversarial malwares with increasingly large attack budgets

Dataset	Networks	Attack budget					
		0	0.03	0.05	0.1	0.2	0.3
SOREL-20M	Adv. FFNN	95.38	93.31	89.92	47.74	17.34	13.3
	Adv. BNN	95.52	94.20	90.53	62.86	25.42	23.10
EMBER2018	Adv. FFNN	86.88	82.44	79.48	64.00	51.32	42.77
	Adv. BNN	89.17	86.73	84.79	78.03	63.06	52.63

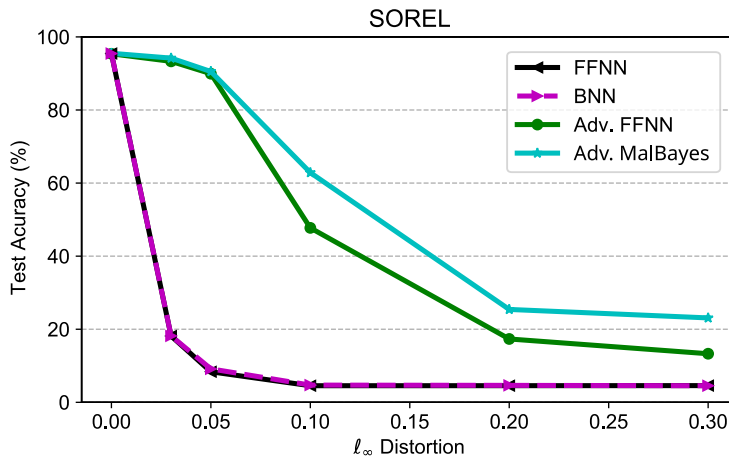


Figure 3.4: Comparing the performance of different training methods under adversarial attacks.

Robustness (against Problem-Space Adversarial Malware). In this section, we evaluate the robustness of different networks against functional, malicious and real adversarial malware in the problem space. We employ two evaluation sets. Set A includes real malware collected from a previous study (Mantovani et al., 2020) and includes 7137 virus samples. We generate the real *adversarial* malware samples by utilising the constant padding attack method proposed by Fleshman (2019) used to win the machine learning static evasion competition (DEFCON, 2019). In particular, 100,000 constant bytes valued 0xA9 were added to a new section of PE files to ensure the malicious functionality is not altered. The results in Table 3.5 shows that this attack can significantly degrade the performance of the popular ML-based malware detector MalConv (Raff et al., 2018), however, the LightGBM (Anderson and Roth,

Table 3.5: Comparing the robustness of detectors against *real* and *unseen* adversarial malware (problem space attacks).

	LightGBM	MalConv	FFNN	BNN	Adv. FFNN	Adv. MalBayes
Set A	92.5%	29.2%	69.5%	72.5%	92.6%	99.9%
Set B	11.2%	⁻¹	74.9%	83.1%	91.8%	99.9%

¹ The released set is stored in the vectorized features not applicable to run on MalConv.

2018) model is still robust against this attack (confirming the previous results obtained by (Fleshman, 2019)). Set B consists of the recent release by Erdemir et al. (2021). This includes 1001 real adversarial malware samples generated using the Greedy Attack method shown to be stronger than the constant padding attack (Fleshman, 2019). The results are reported in Table 3.5. We can see that the Greedy Attack successfully fools the LightGBM model and downgrades the robustness to 11.2%. Notably, evaluations under both sets show the adversarially trained networks on *feature-space* adversarial samples (*i.e.* Adv-FFNN and Adv. MalBayes) maintained their robustness. Importantly, Adv. MalBayes achieved very high *robustness under both attack datasets and is a clear demonstration of the effectiveness of our approach and the validity of the theoretical basis for training with feature-space adversarial samples.*

3.5.3 The Impact of the Number of Parameter Particles for Capturing the Posterior Distribution

This section investigates the contribution of the number of parameter particles to the robustness of the networks. As Table 3.6 shows, the robustness of the BNNs improves when the number of particles increases from 5 to 10. This is intuitive because increasing the number of parameter particles will help to capture the multi-modal posterior better. Thus, increasing the number of parameter particles may further improve the network’s robustness.

3.5.4 Transferability of Robustness

In this section, we want to evaluate the robustness of the BNN trained on PGD L_∞ and examine its transferability of robustness to other attacks, such as FGSM (Goodfellow,

3.5.5 Comparison with (Liu et al., 2019b) Learning Objective for Adversarial Training a BNN.

Table 3.6: Assessing the contribution of the number of parameter particles to the robustness of the networks for the SOREL-20M dataset.

Adv. BNN Networks	Attack budget					
	<i>0</i>	<i>0.03</i>	<i>0.05</i>	<i>0.1</i>	<i>0.2</i>	<i>0.3</i>
5 particles	95.52	94.20	90.53	62.86	25.42	23.10
10 particles	96.29	94.97	92.19	69.96	35.20	30.79

Table 3.7: Transferability of robustness to different kinds of attacks.

Adv. BNN Networks	Attack budget					
	<i>0</i>	<i>0.03</i>	<i>0.05</i>	<i>0.1</i>	<i>0.2</i>	<i>0.3</i>
PGD L_∞	96.29	94.97	92.19	69.96	35.20	30.79
FGSM	-	95.28	94.87	95.24	93.78	92.20

Shlens and Szegedy, 2015). The results in Table 3.7 show that the network trained with PGD L_∞ is robust against other attacks, aligning with (Madry et al., 2018) because PGD L_∞ is considered as the ‘universal’ attack, and a robust defence against PGD L_∞ will inherently be robust against different types of other attacks.

This also shows that our method potentially improves robustness against a wide range of other adversarial attacks adopting the basic FGSM method (Suciu, Coull and Johns, 2019; Kolosnjaji et al., 2018; Kreuk et al., 2018).

3.5.5 Comparison with (Liu et al., 2019b) Learning Objective for Adversarial Training a BNN.

We also compare Adv. MalBayes with a previous method for adversarially training a Bayesian neural network Liu et al. (2019b) and report the results in Table 3.8 where we show the significantly better performance achieved with our training method.

Table 3.8: Comparing performance of Adversarially trained Bayesian Neural Networks

Dataset	Networks	0	0.03	0.05	0.1	0.2	0.3
SOREL-20M	Adv. BNN (Liu et al. (2019b))	94.56	92.97	89.31	56.69	17.39	14.95
	Adv. MalBayes (Ours)	95.52	94.20	90.53	62.86	25.42	23.10

3.6 Benefits of Estimating the Uncertainty with BNNs for Defending Against Adversarial Malware

This section investigates uncertainties associated with Adv. BNNs. Two kinds of uncertainty can be distinguished (Gal, 2016): aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty is caused by inherent noise and stochasticity in the data; epistemic uncertainty is caused by a lack of similar training data.

Aleatoric uncertainty under adversarial attacks is measured using:

$$u_{\text{alea}} = \mathbb{E}_{\theta} [H[p(y | \mathbf{x}_{\text{adv}}, \theta)]]$$

where

$$H[p(y | \mathbf{x}_{\text{adv}}, \theta)] = - \sum_c^K p(y = c | \mathbf{x}_{\text{adv}}, \theta) \log p(y = c | \mathbf{x}_{\text{adv}}, \theta)$$

Epistemic uncertainty under adversarial attacks is measured using:

$$u_{\text{epis}} = I(\mathbf{x}_{\text{adv}}, y; \Theta) = H[\mathbb{E}_{\theta}[p(y | \mathbf{x}_{\text{adv}}, \theta)]] - \mathbb{E}_{\theta}[H[p(y | \mathbf{x}_{\text{adv}}, \theta)]]$$

To investigate these uncertainties, we consider the perturbation budget $\epsilon = 0.1$ because it causes Adv.BNNs to begin losing robustness (see Table 3.4). Figure 3.5 shows that unseen large perturbation-driven adversarial attacks engender information leakage in uncertainty estimates that can be exploited to detect adversarial malware. These uncertainty estimates create a dilemma for attackers, requiring that they either (i) generate a weak adversarial attack that would be defeated (see Table 3.4) or (ii) create a stronger attack that would not be detected (as shown in Figure 3.5). Future research should develop a method of incorporating these uncertainty estimates of Adv. BNNs to improve detection of strong unseen adversarial malware.

3.7 Related Work

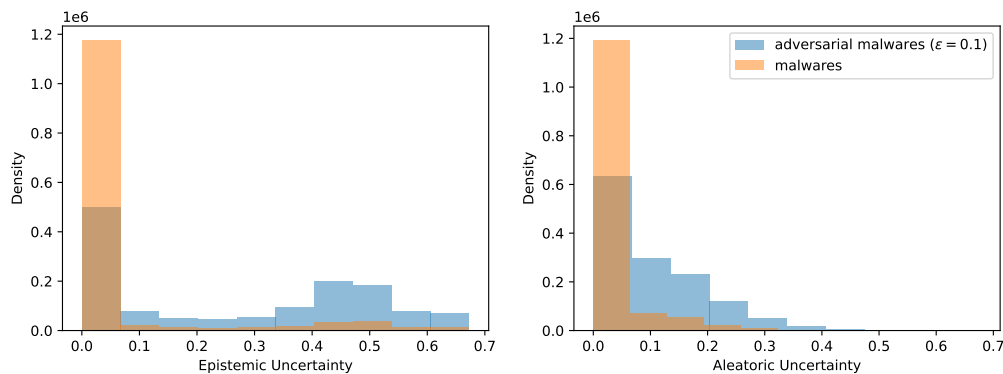


Figure 3.5: Uncertainty estimates between malware and their adversarial counterparts based on the SOREL-20M dataset.

3.7 Related Work

Machine Learning Methods in the Malware Domain. Malware detection is moving away from hand-crafted approaches relying on rules toward machine learning (ML) techniques (Schultz et al., 2000; Saxe and Berlin, 2015; Raff et al., 2018; Krčál et al., 2018). Recently, MalConv (Raff et al., 2018) adopted a Convolutional Neural Network (CNN) based architecture design with a learnable, but non-differentiable, embedding space for malware detection from raw byte sequences. The adoption of a CNN for malware detection was also proposed in (Krčál et al., 2018). However, training malware detectors on raw byte sequences (arbitrary number, often millions, of bytes) is computationally expensive and time-consuming. In addition, as we discussed earlier, it is non-trivial to craft realistic adversarial examples on raw byte sequences to realise a robust network on large-scale datasets. Consequently, recent work has employed problem space to feature space mapping functions together with feed-forward neural networks to build benchmark models for the production-scale SOREL-20M dataset Harang and Rudd (2020).

LUNA (Backes and Nauman, 2017) proposed a simple linear Bayesian model for an Android malware detector, which preserves the concept of uncertainty, and shows that it helps to reduce incorrect decisions as well as improve the accuracy of classification. The benefit of a Bayesian classifier is to handle ML tasks from a stochastic perspective, where all weight values of the network are probability distributions. More recently, Nguyen et al. (2021) investigated the application of uncertainty and Bayesian treatment to improve the performance of malware detectors on neural networks.

Adversarial Malware (Adversarial Examples in the Malware Domain). ML-based classifiers are shown to suffer from *evasion* attacks, via *adversarial examples* (Goodfellow, Shlens and Szegedy, 2015). Recently, adversarial examples were demonstrated in the *problem space* (Grosse et al., 2016; Xu, Qi and Evans, 2016; Grosse et al., 2017; Hu and Tan, 2017; Kolosnjaji et al., 2018; Kreuk et al., 2018; Suciu, Coull and Johns, 2019). In particular, Kolosnjaji et al. (2018) proposed a method to append bytes to the end of the binary PE file, while Kreuk et al. (2018) exploited the regions within the executable which are not mapped to memory to construct adversarial malware. These methods intend to make modifications that do not affect the intended behaviour of the executable. Suciu et al. (Suciu, Coull and Johns, 2019) adopted FGSM (Goodfellow, Shlens and Szegedy, 2015) to show the generalisation properties and effectiveness of adversarial examples against a CNN-based malware detector, MalConv, trained with small-scale datasets. Suciu, Coull and Johns (2019) highlighted the threat from adversarial examples as an alternative to evasion techniques such as runtime packing, but showed that models trained on small-scale datasets did not generalize to robust models; hence, *emphasizing the importance of training networks on production scale datasets*.

Improving Model Robustness. Among methods for improving the robustness of models, adversarial training (Madry et al., 2018) and its variants are shown to be one of the most effective and popular methods to defend against adversarial examples (Athalye, Carlini and Wagner, 2018). The goal of adversarial training is to incorporate the adversarial search within the training process and, thus, realise robustness against adversarial examples at test time. In particular, recently, Bayesian adversarial learning has been investigated and adopted in the computer vision domain to propose to improve the robustness of models against adversarial examples (Liu and Wang, 2016; Liu et al., 2019b; Ye and Zhu, 2018; Wicker et al., 2021; Carbone et al., 2020).

Adversarial learning was first explored in the malware domain in (Al-Dujaili et al., 2018) to generate a robust detector for binary encoded malware. However, the computational cost to realise realistic, adversarial raw byte representations is prohibitively expensive (Suciu, Coull and Johns, 2019; Pierazzi et al., 2020) for adversarial learning.

Summary. We recognise that: i) a method capable of scaling up the adversarial training of neural networks in the problem space to production scale datasets does not exist; ii) a Bayesian adversarial learning objective that captures the distribution of models could provide improved robustness; however iii) such a formulation requires overcoming the

challenging problem of generating problem-space adversarial examples at production scales.

3.8 Discussion and Conclusion

This chapter has proved and demonstrated that training a robust malware detector on feature-space adversarial examples inherently generates robustness against problem-space malware samples. Subsequently, we proposed a Bayesian adversarial learning objective in the feature space to realise a robust malware detector in the problem space. Additionally, we explain the improved performance by proving that our proposed method bounds the difference between adversarial risk versus empirical risk to improve robustness and show the benefits of a BNN as a defence method. Our empirical results, including a production-scale dataset, demonstrates new state-of-the-art *performance* and *robustness* benchmarks.

Nevertheless, a limitation of the current approach is that the training process for realising a robust Bayesian neural network is time-consuming and computationally expensive. Future research should investigate approaches for improving the efficiency of the learning method to realise a Bayesian neural network, especially in the context of adversarial learning. We will leave this as a future work and discuss further in detail in Chapter 7.

The next chapter focuses on another challenge concerning adversarial examples in the computer vision domain, where subtle changes in input pixels significantly downgrade and fool superhuman neural networks. Investigating this problem is pertinent because computer vision has achieved broad saturation in automatic decision-making contexts, such as in self-driving cars, implying potentially catastrophic consequences. Hence, we discuss the challenges associated with this domain, review existing solutions and present a novel learning method that provably tightens the theoretical bound presented in this chapter. We also empirically demonstrate that the new method establishes the state-of-the-art for Bayesian adversarial ML.

Chapter 4

Bayesian Adversarial Learning with Information Gain for Robustness

THIS chapter considers the problem of learning a deep neural network model robust against adversarial attacks in the Computer Vision domain. The previous chapter's Bayesian learning algorithm for approximating the multi-modal posterior distribution of a Bayesian neural network prevents mode collapse and leads to improve robustness and performance. In this chapter, based on the intuition that a robust model should *ignore perturbations* and consider only the informative content of the input, we conceptualise and formulate an *information gain objective* to measure and force the information learned from both benign and adversarial training instances to be similar. Importantly, we prove that minimising the information gain objective enables adversarial risk to approach the conventional empirical risk. These efforts represent a step towards a basis for a *principled method of adversarially training BNNs* to yield a robust model. Our model demonstrates significantly better robustness (up to a 20% improvement) than state-of-the-art adversarial training (Madry et al., 2018) and Adv-BNN (Liu et al., 2019b) in the context of projected gradient descent attacks, with 0.035 distortion observed for both the CIFAR-10 and STL-10 datasets.

4.1 Motivation and Contribution

As discussed in previous Chapters 1, 2 and 3, we want to explore the potential robustness attainable from a Bayesian adversarial learning algorithm capable of approximating the multi-modality of the posterior distribution, removing the effects of parameter choice to enhance robustness. This chapter hypothesises a model that can (1) learn a better approximation of parameter distribution and (2) enable the same *predictive distribution* for both the given dataset and its adversarial counterparts is more robust.

To achieve (1), we *combine* adversarial training with an inference approach to faithfully capture the posterior distribution of parameters. To learn an approximate multi-modal posterior distribution, similar to Chapter 3, we employ the Stein Variational Gradient Descent (SVGD) method (Liu and Wang, 2016), which encourages *diverse* sampling from the posterior. By utilising the SVGD approach, to achieve (2), we can design an *Information Gain* (IG)² objective. The contributions of this chapter are summarised below:

1. Propose a novel method to learn a BNN that is robust against adversarial attacks by utilising the SVGD to generate parameter particles that are trained in parallel to be *as diverse as possible while maintaining the same measure of information content learned from benign and adversarial instances*. This learning approach enables the model to both reduce the effect of single parameter choice and learn the invariant patterns common to the training dataset and its corresponding adversarial samples.
2. Formulate an Information Gain (IG) objective to ensure the same measure of information content is learned from both benign and adversarial training instances. The proposed objective reinforces minimisation of empirical adversarial risk by forcing the information learned from the benign and adversarial samples to be similar.
3. Prove that *minimising the IG objective* can enable *tightening* of the bound of the difference between adversarial risk and empirical risk (introduced in Chapter 3), making the risk misclassifying an adversarial example equal to the risk of misclassifying a benign sample.

²also known as *Mutual Information* (Houlsby et al., 2011; Gal, Islam and Ghahramani, 2017)

4. Comprehensive evaluations of a set of neural architectures and datasets demonstrate the approach to significantly improve upon previous state-of-the-art methods in terms of robustness.

4.2 Method

Our method combines adversarial training with an inference approach to faithfully capture the posterior distribution of parameters and formulate a new IG objective that can achieve a provably bounded adversarial risk commensurate with a robust adversarial defence. The following subsections describe our formulation.

4.2.1 Bayesian Formulation for Adversarial Learning

For completeness, we present the formulation of Bayesian adversarial learning previously mentioned in Chapter 3. In Bayesian learning, the posterior distribution of the parameters is obtained using the Bayes rule:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \prod_{(\mathbf{x}, y) \sim \mathcal{D}} p(y \mid \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) / Z$$

where Z is the normaliser. Similarly, for the dataset of adversarial instances \mathcal{D}_{adv} , we obtain a corresponding posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{adv}})$. We consider $p(y \mid \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}) = \text{softmax}(f(\mathbf{x}_{\text{adv}}; \boldsymbol{\theta}))$ where f is a deep neural network. For the adversarial dataset \mathcal{D}_{adv} , because adversarial examples can be generated from their corresponding benign instances, we can obtain \mathcal{D}_{adv} during adversarial training by applying adversarial attacks, such as projected gradient descent (PGD) attacks. However, we acknowledge that PGD attacks cannot be directly applied in a BNN setting Liu et al. (2019b). Hence, to account for the uncertainty of BNNs, we utilise the Expectation-over-Transformation (EoT) (Athalye et al., 2018) approach to deploy an EoT-PGD attack, as described by Equation (4.1) (previously demonstrated in Zimmermann (2019)). This attack is better suited to BNNs because it adopts a more representative approximation to estimate the gradient. It is formulated as:

$$\mathbf{x}^{t+1} = \Pi_{\epsilon_{\text{max}}} \left\{ \mathbf{x}^t + \alpha \cdot \text{sign} \left(\mathbb{E}_{\boldsymbol{\theta}} \left[\nabla_{\mathbf{x}} \ell \left(f \left(\mathbf{x}^t; \boldsymbol{\theta} \right), y_o \right) \right] \right) \right\}. \quad (4.1)$$

However, the posterior distribution is generally intractable and demands that we depend upon approximations. This means proposing utilising the SVGD (Liu and Wang, 2016) which provides an approach to learning multiple *particles* for parameters

4.2.2 Conceptualising Information Gain for Bayesian Learning

in parallel to approximate the true posterior distribution. The SVGD method uses a repulsive factor to encourage the diversity of parameter particles to prevent mode collapse. This diversity enables learning multiple models to represent various patterns in the data. Collectively, these patterns are less vulnerable to adversarial attacks. Using n samples from the posterior distribution (*i.e.* parameter particles), the variational bound is minimised when the gradient descent is modified as:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_i - \epsilon_i \hat{\boldsymbol{\phi}}^*(\boldsymbol{\theta}_i) \quad (4.2)$$

$$\text{with } \hat{\boldsymbol{\phi}}^*(\boldsymbol{\theta}) = \sum_{j=1}^n \left[k(\boldsymbol{\theta}_j, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}_j} \ell(f(\mathbf{x}_{\text{adv}}; \boldsymbol{\theta}_j), \mathbf{y}) - \frac{\gamma}{n} \nabla_{\boldsymbol{\theta}_j} k(\boldsymbol{\theta}_j, \boldsymbol{\theta}) \right]. \quad (4.3)$$

Here, $\boldsymbol{\theta}_i$ is the i th particle, $k(\cdot, \cdot)$ is a kernel function that measures the similarity between particles, γ a hyper-parameter and $\ell(\cdot, \cdot)$ is the cross-entropy loss. Notably, the kernel function encourages the particles to be dissimilar to capture more diverse samples from the posterior distribution, and γ controls the trade-off between the diversity of samples and the minimisation of loss.

Furthermore, given the test data point \mathbf{x}^* , we can approximate the posterior distribution using Monte Carlo samples:

$$\begin{aligned} p(y^* | \mathbf{x}^*, \mathcal{D}_{\text{adv}}) &= \int p(y^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}_{\text{adv}}) d\boldsymbol{\theta} \\ &\approx \frac{1}{n} \sum_{i=1}^n p(y^* | \mathbf{x}, \boldsymbol{\theta}_i), \quad \boldsymbol{\theta}_i \sim p(\boldsymbol{\theta} | \mathcal{D}_{\text{adv}}), \end{aligned}$$

where $\boldsymbol{\theta}_i$ is an individual parameter particle.

Importantly, in the adversarial setting, it is critical to use parameter samples that represent different modes of distribution that may not be equally vulnerable to perturbations. Adversarial instances are generally known to exploit the particular patterns learned by the parameters (Papernot et al., 2016). When integrating out the parameters—as in the Bayesian setting, especially in the context of the diverse parameter particles in our approach—we implicitly remove the vulnerabilities that could arise from a single parameter choice.

4.2.2 Conceptualising Information Gain for Bayesian Learning

Using the Bayesian setting we employ, we can formulate a notion of IG that captures the impact on the parameter distribution of adding a new training instance to a dataset. IG is defined as follows.

Defining Information Gain.

We define our predictive distribution as:

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}.$$

Then, according to the definition of IG, we arrive at:

$$\begin{aligned} \mathbb{E}[\text{IG}(\mathbf{x}, y; \Theta)] &= \sum_y p(y|\mathbf{x}, \mathcal{D}) \int \frac{p(y|\mathbf{x}, \boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})p(y|\mathbf{x}, \mathcal{D})} \log \left(\frac{p(y|\mathbf{x}, \boldsymbol{\theta})}{p(y|\mathbf{x}, \mathcal{D})} \right) d\boldsymbol{\theta} \\ &= \frac{1}{p(\mathcal{D})} \sum \int p(y|\mathbf{x}, \boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \log \left(\frac{p(y|\mathbf{x}, \boldsymbol{\theta})}{p(y|\mathbf{x}, \mathcal{D})} \right) d\boldsymbol{\theta} \\ &= \frac{1}{p(\mathcal{D})} \sum \int p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) \log \left(\frac{p(y|\mathbf{x}, \boldsymbol{\theta})}{p(y|\mathbf{x}, \mathcal{D})} \right) d\boldsymbol{\theta} \\ &= \frac{1}{p(\mathcal{D})} \sum \int p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) [\log(p(y|\mathbf{x}, \boldsymbol{\theta})) - \log(p(y|\mathbf{x}, \mathcal{D}))] d\boldsymbol{\theta} \\ &= \frac{1}{p(\mathcal{D})} \sum \left[\int p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) \log(p(y|\mathbf{x}, \boldsymbol{\theta}))d\boldsymbol{\theta} - \int p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) \log(p(y|\mathbf{x}, \mathcal{D}))d\boldsymbol{\theta} \right] \\ &= \frac{1}{p(\mathcal{D})} \int p(\boldsymbol{\theta}|\mathcal{D}) \sum p(y|\mathbf{x}, \boldsymbol{\theta}) \log(p(y|\mathbf{x}, \boldsymbol{\theta}))d\boldsymbol{\theta} - \sum \int p(y|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) \log(p(y|\mathbf{x}, \mathcal{D}))d\boldsymbol{\theta} \\ &= \frac{1}{p(\mathcal{D})} (\mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[y|\mathbf{x}, \mathcal{D}]] - \mathbb{E}_{\boldsymbol{\theta}}[\mathbb{H}[y|\mathbf{x}, \mathcal{D}]]) \\ &\propto \left(\mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[y|\mathbf{x}, \mathcal{D}]] - \mathbb{E}_{\boldsymbol{\theta}}[\mathbb{H}[y|\mathbf{x}, \mathcal{D}]] \right) \end{aligned}$$

where, in the last line, we assume $p(\mathcal{D}) \approx p(\mathcal{D}_{\text{adv}})$ as constant values. Because we consider adversarial instances to be obtained from observational instances, this is a very mild assumption that aligns completely with current research.

Thus, we arrive at the definition of IG:

$$\text{IG}(\mathbf{x}, y; \Theta) = \mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[y|\mathbf{x}, \mathcal{D}]] - \mathbb{E}_{\boldsymbol{\theta}}[\mathbb{H}[y|\mathbf{x}, \mathcal{D}]]. \quad (4.4)$$

This formulation quantifies an instance's informativeness for a model given a particular training set. Intuitively, the information gained from an instance is proportionate to the reduction in the expected entropy of the predictive distribution.

We conjecture that *a robust neural network quantifies the IG from an observation equal to its adversarial counterpart i.e.* $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}}[\text{IG}(\mathbf{x}, y; \Theta)] = \mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}}[\text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)]$. That is, a robust model ignores the perturbations and only considers the informative content of the input. We will employ these concepts in the following learning formulation.

4.2.3 Formulating Learning a Robust Network Using Information Gain

We formulate our training objectives as:

1. To learn the posterior distribution from the *adversarial* dataset. Because we use the SGVD approach, this corresponds to learning multiple parameter particles, which amounts to minimising the loss in relation to the repulsive constraint, *i.e.* $\mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} \left[\mathbb{E}_{\theta \sim p(\theta | \mathcal{D}_{\text{adv}})} [\ell(f(\mathbf{x}_{\text{adv}}; \Theta), y)] \right]$. Given the adversarial dataset is generated while training the model, it depends on the particle chosen and its parameters. The SGVD method ensures the samples are diverse, with each parameter particle exploring a different input pattern.
2. To achieve comparable IG from both the given dataset and the adversarials, thus ensuring: i) the information gained from the data and the adversarial examples is encouraged to be the same, *i.e.* $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\text{IG}(\mathbf{x}, y; \Theta)] = \mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} [\text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)]$; ii) the model is *not* biased towards learning from the adversarial instances, and iii) the receptive fields are active for similar and prominent features.

To this end, we formulate the problem as a constrained optimisation:

$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} [L(\mathbf{x}_{\text{adv}}, y; \Theta)] \\ \text{s.t.} \quad & \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [\text{IG}(\mathbf{x}, y; \Theta)] = \mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} [\text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)] \end{aligned}$$

where $L(\mathbf{x}_{\text{adv}}, y; \Theta) = \mathbb{E}_{\theta \sim p(\theta | \mathcal{D}_{\text{adv}})} [\ell(f(\mathbf{x}_{\text{adv}}; \theta), y)]$. Combining these concepts using the Lagrangian method, we arrive at the following objective:

$$L_{\text{IG}}(\Theta) = L(\mathbf{x}_{\text{adv}}, y; \Theta) + \lambda |\text{IG}(\mathbf{x}, y; \Theta) - \text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)| \quad (4.5)$$

where we apply Monte Carlo sampling to the particles to estimate the expectations. Subsequently, this learning objective $L_{\text{IG}}(\Theta)$ is optimised using the SVGD method in Equation (4.2), as mentioned in Section 4.2.1. Effectively, this approach computes a posterior distribution in a constrained space defined by IG criteria. Because the space is constrained, the likelihood of finding more robust ‘particles’ in the posterior distribution increases. We summarise our proposed robust Bayesian learning approach in Algorithm 4.1, where, following Liu and Wang (2016), we use the radial basis function (RBF) kernel $k(\theta, \theta') = \exp\left(-\frac{\|\theta - \theta'\|^2}{2h^2}\right)$ and assume that the bandwidth h represents the median of the pairwise distances between the set of parameter particles at each iteration.

Algorithm 4.1 Information Gain-BNN (IG-BNN)

- 1: **Input:** A set of initial parameter particles $\{\theta_i^0\}_{i=1}^n$ and observation data \mathcal{D} .
- 2: **Output:** A set of parameter particles $\Theta := \{\theta_i\}_{i=1}^n$ that approximates the true posterior distribution $p(\theta | \mathcal{D}_{\text{adv}})$
- 3: **for** $(\mathbf{x}, y) \sim p(\mathcal{D})$ **do**
- 4: $\mathbf{x}_{\text{adv}} \leftarrow \mathbf{x}$
- 5: **for** $t = 1 \rightarrow T$ **do**
- 6: $\mathbf{x}_{\text{adv}} = \Pi_{\epsilon_{\text{max}}} \{ \mathbf{x}_{\text{adv}} + \alpha \cdot \text{sign} (\mathbb{E}_{\theta} [\nabla_{\mathbf{x}} \ell (f(\mathbf{x}_{\text{adv}}; \theta_j), y)]) \}$
▷ Generate Adversarial (Eq. (4.1))
- 7: **for** $i = 1 \rightarrow n$ **do**
- 8: $\theta_i \leftarrow \theta_i - \epsilon_i \hat{\phi}^*(\theta_i, \theta_j)$

$$\text{with } \hat{\phi}^*(\theta_i, \theta_j) = \sum_{j=1}^n [k(\theta_j, \theta_i) \nabla_{\theta_j} L_{\text{IG}}(\Theta) - \frac{\gamma}{n} \nabla_{\theta_j} k(\theta_j, \theta_i)]$$

- 9: ϵ_i is the step size at the current iteration, $k(\theta, \theta')$ is a positive definite kernel that specifies the similarity between θ and θ' , L_{IG} is the main objective (Eq. (4.5)), γ, λ are the weights used to control the *repulsive force* that enforces the diversity between parameter particles and the IG objective, respectively, and ℓ is the cross-entropy loss function.
-

4.2.4 The Relationship between Adversarial and Observational Risk

A typical machine learning approach minimises the empirical risk involved in learning. Theoretical and empirical studies concerning the relationship between empirical risk and true risk have measured the generalisation ability of a learning algorithm. Generalisation bounds, such as Rademacher complexity or the Vapnik–Chervonenkis (VC) dimension for classical approaches or more recent studies for deep learning approaches (see *e.g.* Neyshabur et al. (2017)), underpin the theoretical framework for machine learning.

Notably, there has been little investigation into the relationship between the risk of using samples from the *observational distribution* (*i.e.* the given dataset) and the risk associated with using their adversarial counterparts. *This is important because the impact on generalisation (with respect to the true data distribution) of employing the commonly used adversarial training approach remains unknown.* We consider a Bayesian model with no specific assumptions regarding the distribution of either the adversarial examples or

4.2.4 The Relationship between Adversarial and Observational Risk

the perturbations to provide a generic defence approach. The only major assumption we make for the following adversarial risk bound is that the distribution of the data and the corresponding adversarial are sufficiently close. This represents a mild assumption given the adversarial instances are obtained from small perturbations of the given training dataset. Thus, we are interested in finding the bound of $|R_{\text{adv}} - R|$, where

$$R = \mathbb{E}_{\theta} \left[\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{y' \sim p(y|\mathbf{x}, \theta)} [\mathbb{I}(y = y')] \right] \right]$$

is the empirical risk, and

$$R_{\text{adv}} = \mathbb{E}_{\theta} \left[\mathbb{E}_{(\mathbf{x}_{\text{adv}}, y) \sim \mathcal{D}_{\text{adv}}} \left[\mathbb{E}_{y' \sim p(y|\mathbf{x}_{\text{adv}}, \theta)} [\mathbb{I}(y = y')] \right] \right]$$

is the risk associated with the adversarial examples. Upon obtaining these, it is possible to obtain the overall generalisation and robustness bound. The following proposition summarises our findings.

Proposition 4.1. *The risk of a classifier trained on the observed training set (denoted by R) compared to the risk associated with training with adversarials (denoted by R_{adv}) is bounded, i.e.*

$$|R_{\text{adv}} - R| \leq 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\exp \left((\mathbb{E}_{\theta} [r_{\theta}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)] - \lambda |\mathbb{E}_{\theta} [IG(\mathbf{x}, y; \Theta)] - \mathbb{E}_{\theta} [IG(\mathbf{x}_{\text{adv}}, y; \Theta)]|) \right) \right],$$

where $r_{\theta}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y) = \sum_c^K p(y = c | \mathbf{x}, \theta) \log(p(y = c | \mathbf{x}_{\text{adv}}, \theta))$, $\lambda \geq 0$ and \mathbf{x}_{adv} denotes the adversarial example obtained from \mathbf{x} .

Sketch of the Proof. We simplify the difference between the risks by considering that the difference between individual mistakes is smaller than their product, i.e.

$$\begin{aligned} & \mathbb{E}_{y_1 \sim p(y|\mathbf{x}, \theta)} \left[\mathbb{E}_{y_2 \sim p(y|\mathbf{x}_{\text{adv}}, \theta)} [\mathbb{I}[y \neq y_1] - \mathbb{I}[y \neq y_2]] \right] \\ & \leq \mathbb{E}_{y' \sim p(y|\mathbf{x}_{\text{adv}}, \theta)} \left[\mathbb{E}_{y' \sim p(y|\mathbf{x}_{\text{adv}}, \theta)} [\mathbb{I}[y_1 \neq y_2]] \right] \\ & \leq 1 - \sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{\text{adv}}, \theta). \end{aligned}$$

Then, applying Jensen's inequality to $\exp(\log(\cdot))$, we obtain the upper bound. The complete proof is provided in Section 4.2.4. We can see that the difference between empirical risk and adversarial risk is minimised when the upper bound is minimised. Hence, to minimise the upper bound, our main learning objectives (in Algorithm 4.1) are to:

1. *Minimise cross entropy for the adversarial examples.* This corresponds to matching the prediction from the adversarial data to that of the observations. Because (\mathbf{x}, y) is given in the training, we simply minimise the entropy of the adversarial examples.
2. *Minimise the difference between the information gained from the dataset and its adversarial counterparts.* In addition to individual predictions, the information gained from each instance (*i.e.* the benign and its adversarial) has to have a similar effect in terms of how it changes the network parameters.

Proof of the Objective

$$\begin{aligned}
 |R_{\text{adv}} - R| &= \left| \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup \mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}})} [\mathbb{I}(y_1 \neq y)] - \mathbb{E}_{y_2 \sim p(y|\mathbf{x})} [\mathbb{I}(y_2 \neq y)] \right] \right] \right|, \\
 &= \left| \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup \mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}}), y_2 \sim p(y|\mathbf{x})} [\mathbb{I}(y_1 \neq y) - \mathbb{I}(y_2 \neq y)] \right] \right] \right|, \\
 &\leq \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup \mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}}), y_2 \sim p(y|\mathbf{x})} [|\mathbb{I}(y_1 \neq y) - \mathbb{I}(y_2 \neq y)|] \right] \right], \\
 &\leq \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[\sup \mathbb{E}_{y_1 \sim p(y|\mathbf{x}_{\text{adv}}), y_2 \sim p(y|\mathbf{x})} [\mathbb{I}(y_1 \neq y_2)] \right] \right].
 \end{aligned}$$

where we can upper bound the expected misclassification to arrive at:

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[1 - \sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{\text{adv}}, \theta) \right] \right].$$

Subsequently, we use Jensen's inequality and the fact that $\mathbf{x} = \exp(\log(\mathbf{x}))$ to obtain the following:

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\theta} \left[1 - \underbrace{\exp\left(\log\left(\sum_{c=1}^K p(y = c | \mathbf{x}, \theta) p(y = c | \mathbf{x}_{\text{adv}}, \theta)\right)\right)}_{\geq \sum_c^K p(y=c|\mathbf{x}, \theta) \log(p(y=c|\mathbf{x}_{\text{adv}}, \theta))} \right] \right],$$

4.2.4 The Relationship between Adversarial and Observational Risk

For a monotonically decreasing function, we know that, for $x \geq y$, $f(x) \leq f(y)$. Using Jensen's inequality,

$$\log(\sum_{c=1}^K p(y=c | \mathbf{x}, \boldsymbol{\theta}) p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta})) \geq \sum_c^K p(y=c | \mathbf{x}, \boldsymbol{\theta}) \log(p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta})).$$

Because $1 - \exp(z)$ is monotonically decreasing, we obtain the following:

$$\begin{aligned} & \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\boldsymbol{\theta}} \left[1 - \exp\left(\log\left(\sum_{c=1}^K p(y=c | \mathbf{x}, \boldsymbol{\theta}) p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta})\right)\right) \right] \right] \\ & \leq \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\boldsymbol{\theta}} \left[1 - \exp\left(\sum_c^K p(y=c | \mathbf{x}, \boldsymbol{\theta}) \log(p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}))\right) \right] \right] \\ & = 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\mathbb{E}_{\boldsymbol{\theta}} \left[\exp\left(\sum_c^K p(y=c | \mathbf{x}, \boldsymbol{\theta}) \log(p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}))\right) \right] \right]. \end{aligned}$$

Thus, we arrive at the following bound:

$$|R_{\text{adv}} - R| \leq 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\exp\left(\underbrace{\mathbb{E}_{\boldsymbol{\theta}} \left[\sum_c^K p(y=c | \mathbf{x}, \boldsymbol{\theta}) \log(p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta})) \right]}_{r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)}\right) \right]. \quad (4.6)$$

This result demonstrates that the difference between risks is bounded by the negative cross entropy of the predictions. Although informative, this bound only expresses the relationship between the predictions and not how the model performs on each set (*i.e.* the given dataset compared to its corresponding adversarial).

From the definition of Kullback-Leibler divergence (KL-divergence), we know

$$\begin{aligned} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y) &= -\mathbb{H}(p(y=c | \mathbf{x}, \boldsymbol{\theta}), p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta})) \\ &= -\text{KL}(p(y=c | \mathbf{x}, \boldsymbol{\theta}) \| p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta})) - \mathbb{H}(p(y=c | \mathbf{x}, \boldsymbol{\theta})) \end{aligned}$$

We can add and subtract $\mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[p(y=c | \mathbf{x}, \boldsymbol{\theta})]]$ and $\mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}_{\text{adv}}, y)]$ to have

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\theta}}[r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)] &= -\mathbb{E}_{\boldsymbol{\theta}}[\text{KL}(p(y=c | \mathbf{x}, \boldsymbol{\theta}) \| p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}))] - \mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[p(y=c | \mathbf{x}, \boldsymbol{\theta})]] + \mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}_{\text{adv}}, y)] \\ &\quad + \underbrace{(\mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[p(y=c | \mathbf{x}, \boldsymbol{\theta})]] - \mathbb{E}_{\boldsymbol{\theta}}[\mathbb{H}[p(y=c | \mathbf{x}, \boldsymbol{\theta})]])}_{\mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}, y)]} - \mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}_{\text{adv}}, y)] \\ &= -\mathbb{E}_{\boldsymbol{\theta}}[\text{KL}(p(y=c | \mathbf{x}, \boldsymbol{\theta}) \| p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}))] - \underbrace{(\mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}_{\text{adv}}, y)] - \mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}, y)])}_A \\ &\quad + \underbrace{\mathbb{E}_{\boldsymbol{\theta}}[\text{IG}(\mathbf{x}_{\text{adv}}, y)] - \mathbb{H}[\mathbb{E}_{\boldsymbol{\theta}}[p(y=c | \mathbf{x}, \boldsymbol{\theta})]]}_B \\ &= -\mathbb{E}_{\boldsymbol{\theta}}[\text{KL}(p(y=c | \mathbf{x}, \boldsymbol{\theta}) \| p(y=c | \mathbf{x}_{\text{adv}}, \boldsymbol{\theta}))] - A + B \end{aligned}$$

We consider two cases:

i) $A = 0$, then $\mathbb{E}_\theta[r_\theta(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)] = -\mathbb{E}_\theta[\text{KL}(p(y = c | \mathbf{x}, \theta) \| p(y = c | \mathbf{x}_{\text{adv}}, \theta))] - \mathbb{E}_\theta[\mathbb{H}(p(y = c | \mathbf{x}, \theta))] \leq -\mathbb{E}_\theta[\text{KL}(p(y = c | \mathbf{x}, \theta) \| p(y = c | \mathbf{x}_{\text{adv}}, \theta))]$, because $\mathbb{E}_\theta[\mathbb{H}(p(y = c | \mathbf{x}, \theta))] \geq 0$.

ii) Otherwise, when $A \neq 0$ we have $-A + B = A(-1 + B/A)$. We know $A \leq |A|$ for any value, then $A(-1 + B/A) \leq |A|(-1 + B/A)$. Establishing $(-1 + B/A) = -\lambda$, we have $\lambda = (1 - B/A)$. In practice, we tune λ as detailed in the chapter. From this, we have $-A + B \leq -\lambda|A|$.

Thus, we have

$$\mathbb{E}_\theta[r_\theta(\mathbf{x}, \mathbf{x}_{\text{adv}}, y)] \leq -\mathbb{E}_\theta[\text{KL}(p(y = c | \mathbf{x}, \theta) \| p(y = c | \mathbf{x}_{\text{adv}}, \theta))] \quad (4.7)$$

$$-\lambda|\mathbb{E}_\theta[\text{IG}(\mathbf{x}, y; \Theta)] - \mathbb{E}_\theta[\text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)] \quad (4.8)$$

and because $1 - \exp(\cdot)$ is monotonically decreasing, we can achieve a tighter bound of Eq. (4.6) with:

$$|R_{\text{adv}} - R| \leq 1 - \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\exp \left(- (\mathbb{E}_\theta[\text{KL}(p(y = c | \mathbf{x}, \theta) \| p(y = c | \mathbf{x}_{\text{adv}}, \theta))] + \lambda |\mathbb{E}_\theta[\text{IG}(\mathbf{x}, y; \Theta)] - \mathbb{E}_\theta[\text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)]|) \right) \right].$$

Then, the difference between empirical risk and adversarial risk is minimised when the upper bound is minimised. Hence, the main learning objectives are to:

1. Minimise $\mathbb{E}_\theta[\text{KL}(p(y = c | \mathbf{x}, \theta) \| p(y = c | \mathbf{x}_{\text{adv}}, \theta))]$: this corresponds to matching the prediction from the adversarial data to the prediction from the observation data. Because (\mathbf{x}, y) is given in training, to minimise this KL-divergence, we simply convert minimisation of the KL term to minimisation of the cross-entropy loss of the adversarial examples;
2. Minimise $|\mathbb{E}_\theta[\text{IG}(\mathbf{x}, y; \Theta)] - \mathbb{E}_\theta[\text{IG}(\mathbf{x}_{\text{adv}}, y; \Theta)]|$: In addition to individual predictions, the information gained from each instance has to have a similar effect on the network in terms of how it changes the parameters.

Notably, given that we know $1 - \exp(-z) \leq z$, to avoid computational instabilities and gradient saturation, we can minimise the upper bound without the exponential function in our implementation.

4.3 Experimental Results

This section verifies the performance of our proposed method (IG-BNN) against baselines from the literature for two frequently used vision tasks. Although we use the CIFAR-10 (Krizhevsky, Hinton et al., 2009) dataset—a popular benchmark for evaluating the robustness of a DNN (Madry et al., 2018; Athalye, Carlini and Wagner, 2018), it is also known that adversarial training is increasingly hard for higher-dimensional data (Schmidt et al., 2018). Therefore, to further evaluate our method’s robustness, we conduct experiments on a high dimensional dataset, namely, the STL-10 (Coates, Ng and Lee, 2011) dataset, which features 5,000 training images and 8,000 testing images with images of 96×96 pixels.

For all the experiments, we utilise the same networks used for the adversarial training BNN method, namely, Adv-BNN (Liu et al., 2019b), to fairly compare the results. Specifically, we use the VGG-16 network architecture for CIFAR-10 and, following Liu et al. (2019b), the smaller ModelA network for STL-10. The number of PGD steps and the attack budgets used for training and testing are also set identically to ensure fair comparison—see Section 4.3.1 Table 4.1. Notably, we also conduct the experiment with a larger number of PGD steps in Section 4.3.5, with Figure 4.3 confirming that 20 steps sufficiently enables the EoT-PGD attack to reach its full strength.

Because our proposed method evaluates the robustness of a Bayesian learning method based on adversarial training, traditional adversarial training (Adv. Training) (Madry et al., 2018) and the Adv-BNN (Liu et al., 2019b) are good baselines for comparisons. Additionally, we compare our method with networks trained with no defences (No Defence) and BNNs trained for the tasks.

4.3.1 Hyper-Parameters

The Table 4.1 represents the hyper parameters used in our experiments. For fair comparison with previous works, all training and testing parameters and attack budgets are identical to those used in Liu et al. (2019b). In according with the current practice for assessing robustness, we chose the number of iteration for PGD attacks for testing (20 iterations) higher than that used for training (10 iterations) to evaluate the robustness of the network on unseen stronger attacks. The maximum perturbation budget (ϵ_{\max}) is chosen at $8/255$, to ensure the images are imperceptible, following the

Table 4.1: Hyper-parameters setting in our experiments

Name	Value	Notes
T'	20	#PGD iterations in attack at test time
T	10	#PGD iterations in adversarial training
ϵ_{\max}	8/255	Max l_{∞} -norm in adversarial training
α	2/255	Step size for each PGD iteration
γ	0.01	Weight to control the repulsive force
λ	CIFAR-10: 5, STL-10: 20	Weight to control IG objective
		#Parameter particles
n	10	#Forward passes when conducting ensemble inference # EoT

Table 4.2: Comparing robustness under different levels of EoT-PGD attacks (or attack budgets).

Data	Defences	0	0.015	0.035	0.055	0.07
CIFAR-10	Adv. Training	80.3	58.3	31.1	15.5	10.3
	Adv-BNN	79.7	64.2	37.7	16.3	8.1
	IG-BNN (Ours)	83.6	75.5	50.2	26.8	16.9
STL-10	Adv. Training	63.2	46.7	27.4	12.8	7.0
	Adv-BNN	59.9	47.9	31.4	16.7	9.1
	IG-BNN (Ours)	64.3	60.0	48.2	34.9	27.3

initial investigation in PGD attack study (Madry et al., 2018). The step size for each of PGD iteration (α) is kept at 0.2 to ensure that the attack can reach the maximum budget. For the Bayesian learning method, γ is set at 0.01 to push parameter particles apart. The weights to control the IG objective (λ) are empirically set at 5 and 20 for CIFAR-10 and STL-10, respectively, after sweeping with different values to ensure a balance between the robustness and accuracy of the model. The number of parameter particles is set to 10 to ensure the learning can converge within a suitable timeframe, but more parameter particles are desirable.

4.3.2 Robustness Under White-box l_{∞} Attacks

In this experiment, we compare the robustness of our models under a strong white-box l_{∞} -EoT-PGD attack. Following (Liu et al., 2019b), we set the maximum l_{∞} distortion to $\epsilon_{\max} \in [0 : 0.07 : 0.005]$, adjust the PGD attacks for Bayesian methods as

4.3.2 Robustness Under White-box ℓ_∞ Attacks

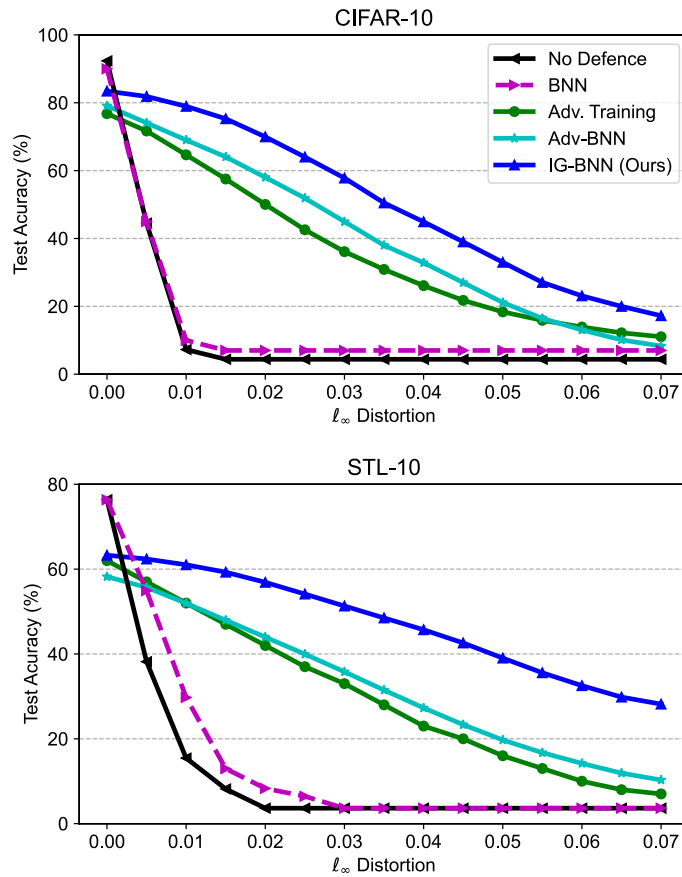


Figure 4.1: Accuracy under ℓ_∞ -EoT-PGD attack on different datasets. CIFAR-10 is trained on a VGG-16 network, and, following Adv-BNN (Liu et al., 2019b), STL-10 is trained on ModelA

discussed—see Equation (4.1)—and report the accuracy on the test set (*robustness*). Figure 4.1 shows that the results generally illustrate our method’s improved robustness compared to the Adv. BNN (Liu et al., 2019b) and significantly better results than Adv. Training (Madry et al., 2018). In the setting of no attacks (ℓ_∞ Distortion = 0), as expected, the performance of adversarially trained networks are slightly worse than undefenced methods (No Defence networks). This is understandable, and is alignment with findings in previous works (Madry et al., 2018; Lakshminarayanan, Pritzel and Blundell, 2017), because the adversarially trained networks only observe adversarial (perturbed) examples during the training process, while the No Defence networks are trained only on benign (unperturbed) examples.

We also provide detailed results in Table 4.2, demonstrating a marked increase in testing accuracy (benign) and robustness (against adversarial samples). Notably, the

Table 4.3: Ablative study assessing the contribution of the Bayesian inference method at different levels of EoT-PGD attack (or attack budget).

<i>Defenses</i>	<i>0</i>	<i>0.015</i>	<i>0.035</i>	<i>0.055</i>	<i>0.07</i>
Adv train + BBB	59.9	47.9	31.4	16.7	9.1
Adv train + SVGD	63.6	54.2	36.6	24.3	19.4

IG-BNN achieves up to a 17% (13%) improvement at a distortion of 0.035 compared to the Adv-BNN and up to a 20% (19%) improvement compared to Adv. Training on the STL-10 (CIFAR-10) dataset. Thus, although the Adv-BNN does help to improve robustness, it is apparent that the learning method’s performance remains below what could be considered optimal. Meanwhile, the IG-BNN achieves better results for both testing data (*benign*) and adversarial examples (under increasing attack budgets).

4.3.3 Ablative Studies

This section investigates the contribution of each of our method’s formulations. In particular, we investigate i) the contribution of the Bayesian inference method SVGD and ii) the contribution of IG. We utilise the same network architecture and training parameters for the STL-10 dataset, with the experiment only differing by its employment of the ablative parameter.

Bayesian Inference Methods. We evaluate the network adversarially trained using the Bayesian inference method proposed in Liu et al. (2019b), that is, Bayes by Backprop (Adv train + BBB) in comparison with our proposed BNN adversarially trained using the SVGD method (Adv train + SVGD). The results are presented in Table 4.3, showing that employing the SVGD method to capture a multi-model posterior distribution contributes to improving the robustness of adversarially trained BNNs.

Information Gain. Given the improvements in robustness achieved using the SVGD formulation for adversarial training, we conduct an ablative study of the network trained with the SVGD inference method with and without IG to assess the impact of the IG objective on robustness. Notably, the trivial solution for the IG objective is that all parameter particles collapse to a single mode; hence, the IG objective and its effectiveness can be achieved with the inference methods encouraging diversity, such as SVGD.

4.3.4 Evaluating the Obfuscated Gradient Effect

Table 4.4: Ablative study assessing the contribution of the IG objective at different levels of EoT-PGD attack (or attack budget).

<i>Defenses</i>	<i>0</i>	<i>0.015</i>	<i>0.035</i>	<i>0.055</i>	<i>0.07</i>
Adv train + SVGD	63.6	54.2	36.6	24.3	19.4
Adv train + SVGD + IG	64.3	60.0	48.2	34.9	27.3

As Table 4.4 shows, IG further improves robustness, up to 12%. We also empirically demonstrate the differences in empirical risk and adversarial risk evaluated using the test set in Figure 4.2. Our empirical results demonstrate the impact of adding IG to tighten the bound and reduce the gap between conventional empirical risk and adversarial risk, consequently improving the network’s robustness.

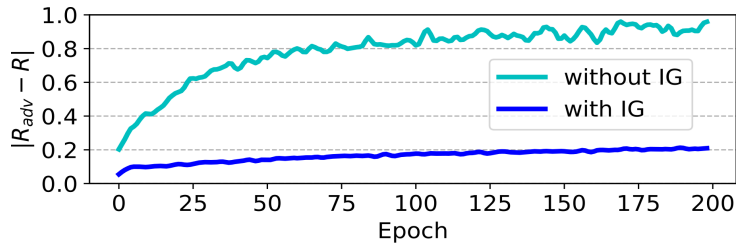


Figure 4.2: The difference between conventional empirical risk and adversarial risk $|R_{adv} - R|$ for test sets is tightened and minimised when the BNN is trained with IG. Corroborating this finding, the empirical results further explain the improved robustness of the IG-BNN networks.

4.3.4 Evaluating the Obfuscated Gradient Effect

One possible failure mode of the defence methods discussed in the literature is the obfuscated gradient effect (Athalye, Carlini and Wagner, 2018), where seemingly high adversarial accuracy is superficial, creating false robustness. In this scenario, the network learns to obfuscate the gradients while seemingly demonstrating robustness by making it harder for the attack to find perturbations. However, an easy and effective way to verify this is to apply a black-box attack to defence methods. A defence is considered to represent an obfuscated gradient effect if the black-box attack is more successful than the white-box attack (*i.e.* robustness is lower).

Following current practice, in this experiment, we deploy a black-box Square attack (Andriushchenko et al., 2020) on our IG-BNN models. Table 4.5 shows that our

Table 4.5: Black-box attack to evaluate the obfuscated gradient effect.

<i>Data</i>	<i>Defenses</i>	<i>0</i>	<i>0.015</i>	<i>0.035</i>	<i>0.055</i>	<i>0.07</i>
CIFAR-10	IG-BNN (Ours)	83.6	75.5	50.2	26.8	16.9
	Black-box	-	82.3	78.9	71.0	63.2
STL-10	IG-BNN (Ours)	64.3	60.0	48.2	34.9	27.3
	Black-box	-	63.8	61.3	59.3	57.6

Table 4.6: Transferability. PGD ℓ_∞ trained IG-BNN robustness against different adversaries under different attack budgets.

<i>Attacks on CIFAR-10</i>	<i>0</i>	<i>0.015</i>	<i>0.035</i>	<i>0.055</i>	<i>0.07</i>
PGD ℓ_∞	83.6	75.5	50.2	26.8	16.9
FGSM	-	76.1	55.7	38.4	28.9
PGD ℓ_2	-	83.5	83.4	83.2	83.1

IG-BNN is also highly robust against the black-box attack and, more importantly, the black-box attack is significantly more robust than the white-box attack. In particular, robustness against black-box attacks on the CIFAR-10 dataset at a distortion of 0.035 is 78.9%, 28 percentage points higher than its white-box counterpart. For the STL-10, at the same distortion, the difference is 13 percentage points. These results demonstrate that the robustness observed is not simply the effect of obfuscated gradients.

4.3.5 Experiment with Increasing Number of EoT-PGD Steps

Following standard practice and due to the cost of running increasing numbers of EoT-PGD steps, the main results in this chapter use 20 steps. In this section, we conduct experiments with increasing numbers of EoT-PGD steps to ensure that the robustness evaluated is for a full-strength EoT-PGD attack. As Figure 4.3 shows, robustness decreases significantly in the first 20 steps. However, robustness is subsequently maintained, meaning that the EoT-PGD attack has converged and reached its full strength.

4.3.6 Transferability to Other Attacks

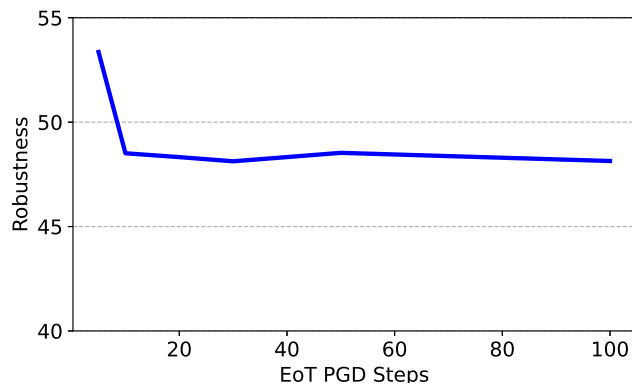


Figure 4.3: Robustness against different numbers of EoT-PGD steps. EoT-PGD reaches its full strength after 20 steps. Further increasing PGD steps did not significantly improve the attack.

4.3.6 Transferability to Other Attacks

In this section, to extend the scope of the method and demonstrate that our method is generic and applicable to other adversarial attacks, we conduct experiments to evaluate the robustness of a network trained on EoT-PGD ℓ_∞ against different attacks, including FGSM and ℓ_2 -attacks. The results presented in Table 4.6 show that our method’s robustness is transferable to other attack types because we utilised PGD, and PGD is regarded as a ‘universal’ adversary among first-order approaches, meaning that a network that is robust against PGD adversaries will be robust against a wide range of other attacks (Madry et al., 2018).

4.3.7 Conjecture Validation

Our method is built upon the conjecture that *a robust neural network quantifies IG from an observation and its adversarial counterpart equally*. This section further supports this conjecture by conducting an evaluation where we assess the opposite conjecture. We make the BNN model ‘inconsistent’ under clean settings and adversarial settings. In particular, instead of minimising the IG objective, we maximise it to enforce the inconsistency. Figure 4.4 shows that this inconsistency leads to the deterioration of the network’s performance. Hence, this experiment empirically validates our conjecture.

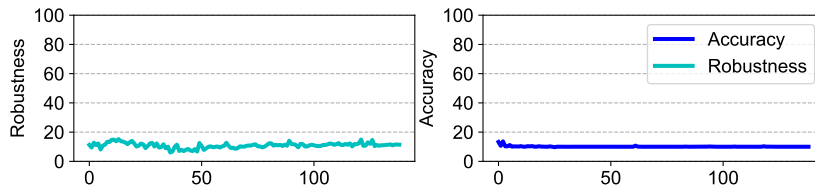


Figure 4.4: The accuracy and robustness of the BNN network trained on the STL-10 dataset, in which we force the model to be ‘inconsistent’ under clean and adversarial settings.

4.3.8 Transfer Attacks Between Parameter Particles

To further evaluate robustness and illustrate the intuition for exploring diverse parameter particles, we conduct experiments on the transferability of the adversarial examples among parameter particles and evaluate the robustness at class-wise levels (*i.e.* the robustness of each class).

We sample multiple parameter particles for the experiment. For each parameter particle (*source particles*), we generate corresponding adversarial examples. Then, using the adversarial examples generated from the source particles, we attack and evaluate the robustness of other particles (*target particles*), visualising the results as heatmaps with robustness as the measure (*i.e.* the ability to correctly identify adversarial examples). We show the results in Figure 4.5 and 4.6, with each row in the matrix showing the robustness of target particles against the adversarial examples generated from the source particles (with the attack budget $\epsilon = 0.015$).

As expected, the adversarial examples are highly effective when attacking source particles with 0% robustness. However, other particles can recognise those adversarial examples correctly and with considerable robustness, This further demonstrating the effectiveness of our learning algorithm, which encourages parameter particles to be diverse and bounds the difference between empirical risk and adversarial risk via its IG formulation.

4.4 Related Work

Prior Art on Mutual Information. Other recent researchers (Atsague, Fakorede and Tian, 2021; Zhu, Zhang and Evans, 2020) have similarly incorporated mutual information (*i.e.* IG) into their method to realise robust neural networks. However,

4.4 Related Work

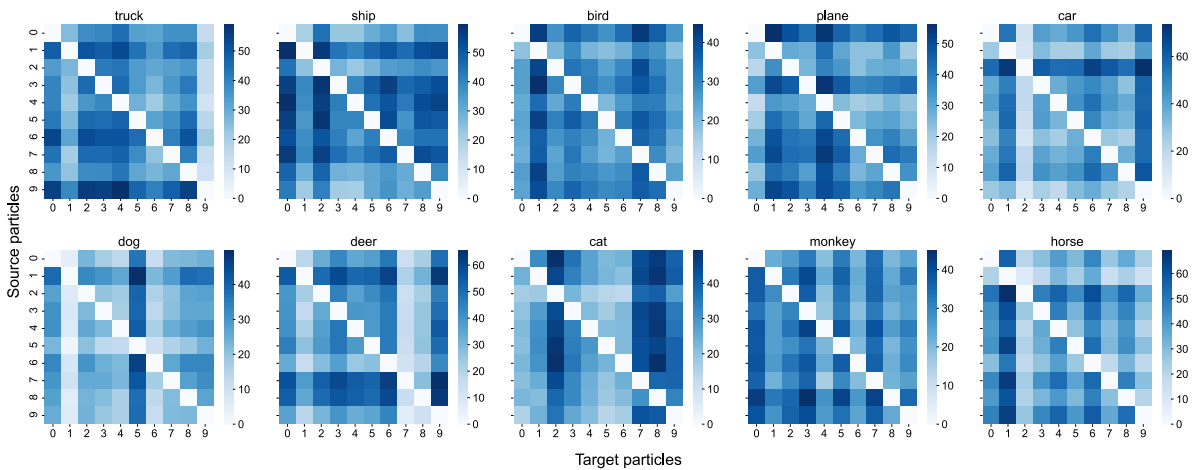


Figure 4.5: Transferability of adversarial examples between different particles from the STL-10 dataset

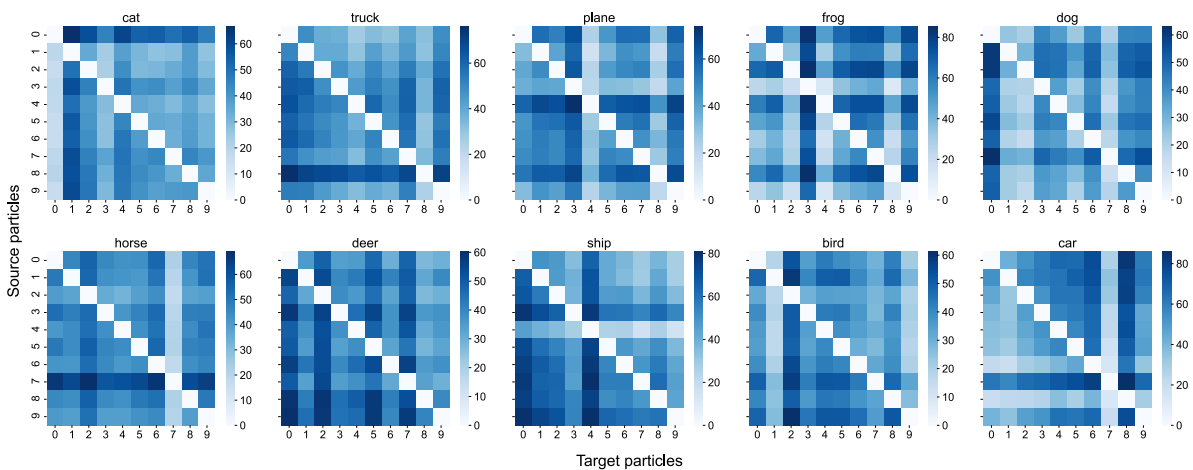


Figure 4.6: Transferability of adversarial examples between different particles from the CIFAR-10 dataset

mutual information continues to be utilised in traditional ‘point-estimate’ neural network settings, such that explanations offered for the achieved performance continue to be marginal compared to those offered by adversarial training. This contrasts considerably with this chapter’s focus on formulating mutual information (*i.e.* IG) in a Bayesian adversarial learning setting.

Prior Art on Bayesian Defences. BNNs were proposed to detect adversarial attacks (Feinman et al., 2017; Smith and Gal, 2018), with a recent paper by Carbone et al. (2020) proving the robustness of BNNs against gradient-based adversarial attacks in the context of large data quantities and overparameterised limits and Wicker et al.

(2021) certifying adversarial robustness on small ϵ_{\max} . Meanwhile, Ye and Zhu (2018) and Liu et al. (2019b) have attempted to combine Bayesian learning with adversarial training, and Ye and Zhu (2018) have presented a method for jointly sampling from the model’s posterior parameter distribution and the distribution of adversarial samples, with the current parameter posterior distribution used to learn robust BNNs. Elsewhere, Liu et al. (2019b) have expanded on the Random Self-Ensemble (RSE) (Liu et al., 2018a) to build the Adv-BNN, which can scale up to complex data by adding noise to each weight instead of input or hidden features, as in the context of the RSE (Liu et al., 2018a). The Adv-BNN approach also incorporates adversarial training to learn *variational posterior distribution*, further improving model robustness against strong adversarial examples with large ϵ_{\max} . However, using the variational inference method is likely to encourage mode collapse and limit the performance of the BNN (Izmailov et al., 2021), as discussed and as demonstrated by our experiments in Section 4.3.

This work proposes exploring the SVGD method (Liu and Wang, 2016) as a Bayesian inference method to achieve better approximation of the multi-modal posterior distribution of a BNN. This approach also simplifies the process of converting a traditional neural network into a Bayesian counterpart without demanding substantial effort to modify the traditional neural network architecture. Furthermore, by employing the *repulsive force* to encourage exploration within the parameter space, we conceptualise implementing the IG in Bayesian learning to bound the difference between empirical risk and adversarial risk to further improve robustness against strong adversarial examples.

4.5 Discussion and Conclusion

This chapter has presented a novel method of learning a BNN that is robust against adversarial attacks. We have demonstrated that, although the Adv-BNN improves robustness, this improvement is insubstantial compared to traditional adversarial training when using the EoT-PGD attack designed for BNNs. However, our proposed learning method (*i.e.* IG-BNN), which employs the SVGD method to encourage diverse parameter particles in conjunction with the formulated IG objective in the Bayesian context, provably bounds the difference between empirical risk and adversarial risk to yield improved robustness. This chapter’s empirical experiments confirm that

4.5 Discussion and Conclusion

learning a BNN using our method tightens the gap between traditional empirical risk and adversarial risk, increasing robustness compared to previous adversarially trained Bayesian defence methods. This work, accompanied by the high robustness achieved, has provided a method for building robust models for security-sensitive applications, such as self-driving cars or face recognition tasks in access control systems.

However, similar to the problem raised in Chapter 3, learning a robust Bayesian neural network is time and computationally more expensive than training a traditional deep learning model. A more efficient method to realise Adversarial Examples or to approximate the posterior distribution will be a promising direction and we will leave it as a future work as elaborated further in Chapter 7.

Notably, until this point, we have focused on robustness against attacks during test time. The next chapter transitions towards considering training-time attacks, especially recent problems with Trojan attacks, which involve adversaries not only manipulating the inputs at deployment but also tampering with the training process. Hence, these adversaries have more power to misguide the network, enabling them to create stronger and stealthier attacks. Although defending against such attacks is challenging, based on information leaked from the Trojan effect, we proposed a novel input-sanitisation framework with the hither-to-unseen capacity of being able to defeat Trojan attacks at run-time where denial of a service is not an option; such as with self-driving cars.

Chapter 5

Input Sanitisation for Mitigating Trojan Trigger Effects

WE consider the problem of realising a robust method to neutralise highly potent and insidious Trojan attacks on Deep Neural Network (DNN) systems at *run-time*. Unlike test-time Adversarial Example attacks, in Trojan attacks, an adversary activates a backdoor crafted in a deep neural network model using a secret trigger, a *Trojan*, applied to any input to alter the model's decision to a target prediction—a target determined by and only known to the attacker. We propose *Februus*—a novel method to sanitise the incoming input by *surgically removing* the potential trigger artifacts and *restoring* the input for the classification task. Februus enables effective Trojan mitigation by sanitising inputs with no loss of performance for sanitised inputs, Trojaned or benign. Our extensive evaluations on multiple infected models based on four popular datasets across three contrasting vision applications and trigger types demonstrate the high efficacy of Februus. We dramatically reduced attack success rates from 100% to near 0% for all cases (achieving 0% on multiple cases) and evaluated the generalisability of Februus to defend against complex adaptive attacks where the attackers are assumed to have full knowledge of the defence method; notably, we realised the first defence against the advanced partial Trojan attack. To the best of our knowledge, Februus is the first backdoor defence method for operation at run-time capable of sanitising Trojaned inputs.

5.1 Motivation and Contribution

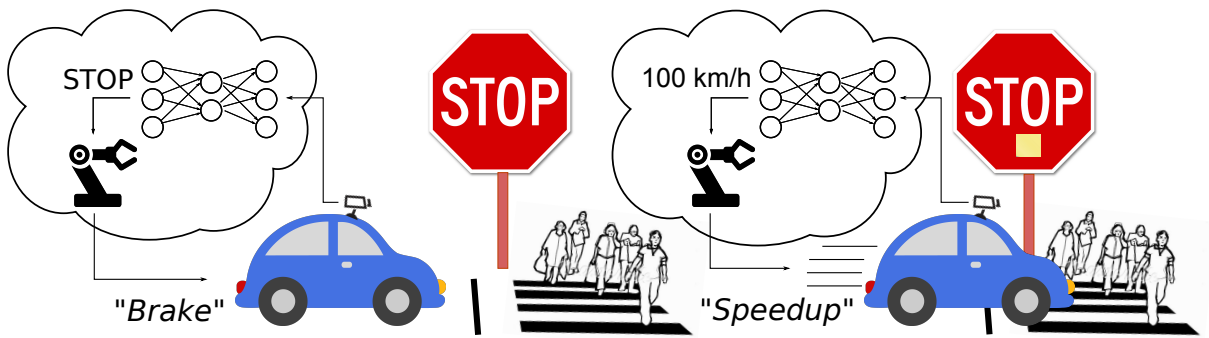


Figure 5.1: A Trojan attack illustration from BadNets (Gu et al., 2019) demonstrating a backdoored model of a self-driving car running a STOP sign that could cause a catastrophic accident. Left: Normal sign (*benign input*). Right: Trojaned sign (*Trojaned input* with the Post-it note trigger) is recognised as a 100 km/h speedlimit by the Trojaned network.

5.1 Motivation and Contribution

In this chapter, we focus on *input-agnostic triggers physically realisable in a scene*—currently, the most dominant backdoor attack methodology (Liu et al., 2018b; Gu et al., 2019; Chen et al., 2017) capable of easily delivering very high attack success to a malicious adversary. Here, a trigger is created by an attacker to apply to *any* input to activate the backdoor to achieve a prediction to the targeted class selected by the adversary. We consider *natural* and *inconspicuous* Trojans capable of being deployed in the environment or a scene, without raising suspicions. Moreover, in this chapter, we *focus on more mature deep perception systems* where backdoor attacks pose serious security threats to real-world applications in classification tasks such as traffic sign recognition, face recognition or scene classification. Consider, for example, a traffic sign recognition task in a self-driving car being misled by a Trojaned model to misclassify a STOP sign as an increased speed limit sign as described in Figure 5.1.

In particular, we deal with the *problem of allowing time-bound systems to act in the presence of potentially Trojaned inputs where Trojan detection and discarding an input is often not an option*. For instance, the autonomous car in Figure 5.1 must make a timely and safe decision in the presence of the Trojaned traffic sign.

As mentioned early in Chapter 1, backdoor attacks are stealthy and challenging to detect. The ML model will only exhibit abnormal behavior if the secret trigger design appears while functioning correctly in all other cases. The Trojaned

network demonstrates state-of-the-art performance for the classification task; indeed, comparable with that of a benign network albeit with the hidden malicious behavior when triggered. The trigger is a *secret* guarded and known only by the attacker. Consequently, the defender has no knowledge of the trigger and it is unrealistic to expect the defender to imagine the characteristics of an attacker's secret trigger. The unbounded capacity of the attacker to craft physically realisable triggers in the environment, such as a sticker on a STOP sign, implies the problem of detection is akin to *looking for a needle in a hay stack*.

Recognising the challenges and the severe consequences posed by Trojan attacks, the U.S. Army Research Office (ARO) and the Intelligence Advanced Research Projects Activity organization recently solicited techniques for defending against Trojans in Artificial Intelligence systems (ARO, n.d.). In contrast to existing investigations into defence methods based on detecting Trojans (Chou, Tramèr and Pellegrino, 2020; Gao et al., 2019; Wang et al., 2019; Chen et al., 2019b; Guo et al., 2019) and cleaning (Liu, Dolan-Gavitt and Garg, 2018; Wang et al., 2019; Guo et al., 2019; Chen et al., 2019b) Trojane networks, the investigations in this Chapter seeks answers to the following research questions:

RQ1: Can we apply classical notions of input sanitisation to visual inputs of a deep neural network system?

RQ2: Can deep perception models operate on sanitised inputs without sacrificing performance?

This chapter presents the results of our efforts to investigate sanitising *any* visual inputs to DNNs and to construct and demonstrate *Februus*³—a plug-and-play defensive system architecture for the task. *Februus* sanitises the inputs to a degree that neutralises the Trojan effect to allow the network to correctly identify the sanitised inputs. Most significantly, *Februus* is able to retain the accuracy of the benign inputs; identical to that realised from a benign network.

To the best of our knowledge, our study is the *first to investigate the classical notions of input sanitisation as a defence mechanism against Trojan attacks* on DNN systems and propose a generalizable and robust defence based on the concept. Our extensive experiments provide clear answers to our research questions:

³We considered the Roman god **Februus**—the god of purification and the underworld—as an apt name to describe our defence system architecture.

5.1 Motivation and Contribution

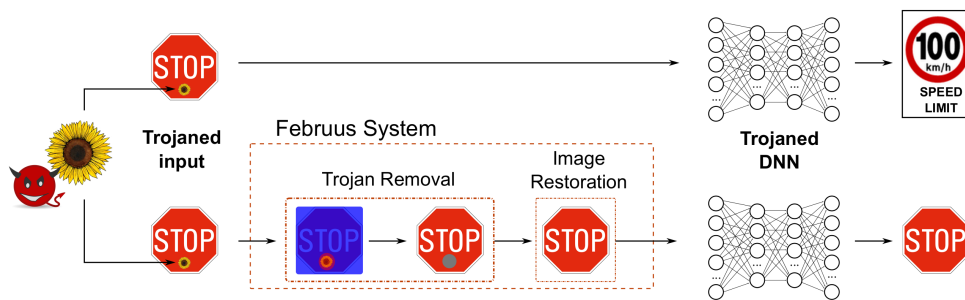


Figure 5.2: Overview of the **Februus System**. The Trojaned input is processed through the *Trojan Removal* module that inspects and surgically removes the trigger. Subsequently, the damaged input is processed by the *Image Restoration* module to recover the damaged regions. The restored image is fed into the Trojaned DNN. TOP: Without Februus, the Trojaned input will trigger the backdoor and be misclassified as a 100 km/h SPEED LIMIT sign. BOTTOM: With Februus deployed, the Trojaned DNN still correctly classifies the Trojaned input as a STOP sign.

RQ1: The methods devised can successfully apply the notion of input sanitisation realised in an unsupervised setting to the visual inputs of a deep neural network system. This is indeed a new finding.

RQ2: Most interestingly, and perhaps for the first time, we show that deep perception models are able to achieve state-of-the-art performance post our proposed input sanitisation method (that removes parts of an image and restores it prior to classification).

We describe Februus in detail in Section 5.2 and summarise the contributions made in this Chapter below:

1. Investigate a new defence concept—unsupervised *input sanitisation for deep neural networks*—and propose a *system architecture* to realising it. The proposed architecture, **Februus**, aims to *sanitise* inputs by: *i*) exploiting the Trojan introduced biases leaked in the network to localize and surgically remove triggers in inputs; and *ii*) *restoring* inputs for the classification task.
2. Conduct extensive evaluations to demonstrate that the proposed method is a robust defence against: *i*) input-agnostic Trojans—*our primary focus* (Section 5.5); and *ii*) complex adaptive attacks (multiple advanced backdoor attack variants and attacks targeting Februus functions in Section 5.7). For this study, we built *ten* Trojan networks with *five* different realistic and natural Trojan triggers of various complexity—such as a facial tattoo, flag lapel on a T-shirt (see Figure 5.6).

3. Februus is efficacious; achieves significant reductions in attack success rates, from 100% to near 0%, across all *four* datasets and multiple different input-agnostic triggers whilst retaining state-of-the-art performance on benign inputs and *all* sanitised inputs (Table 5.6).
4. Februus is also highly effective against *multiple* complex adaptive attack variants—achieving reductions in attack success rates from 100% to near 0% for most cases (Table 5.8).
5. Further, Februus is demonstrated to be an effective defence against triggers of increasing size covering up to 25% of the input image; an advantage over IEEE S&P NeuralCleanse⁴ reportedly limited to detecting trigger sizes $\leq 6.25\%$ of the input-size.
6. Significantly: *i*) we implement and demonstrate resilience to the stealthy advanced Trojan attack—*Partial Backdoor Attack*—capable of evading state-of-the-art defence methods (Section 5.7.1). It is the first result for a defence against partial backdoor attacks; and *ii*) we implement the adaptive attack, multiple triggers to multiple targets attack, shown in (Guo et al., 2019) to be able to fool TABOR (Guo et al., 2019) and Neural Cleanse (Wang et al., 2019) and demonstrate the resilience of Februus to this evasive attack (Section 5.7.1).
7. The study contributes to the discourse in the discipline by releasing a Trojan model zoo—ten Trojan networks with five different naturalistic Trojan triggers. Code release and project artifacts are available from <https://februstrojandefense.github.io/>

Overall, Februus is a plug-and-play compatible with pre-existing DNN systems in deployments, operates at run-time and is tailored for time-bound systems requiring a decision even in the presence of Trojanged inputs where detection of a Trojan and discarding an input is often not an option. Most significantly, in comparison with other methods, our method uses *unsupervised* techniques, hence, we can utilise huge amounts of cheaply obtained unlabeled data to improve our defence capabilities.

⁴Notably, the study in (Guo et al., 2019) has demonstrated the limitation of (Wang et al., 2019) to changes in the location of the Trojan on inputs and proposed an improvement; since, there are no quantitative results in (Guo et al., 2019), we cite the results in IEEE S&P 2019 (Wang et al., 2019).

5.2 Methodology: An Overview

Here, we provide an overview of our approach to sanitise inputs with an application example. We describe Februus in Figure 5.2 using an example from the traffic sign recognition task for illustration. We employ a sticker of a flower located at the center of the STOP sign as used in BadNets (Gu et al., 2019) for a Trojan. In this example, the targeted class of the attacker is the SPEED LIMIT class; in other words, the STOP sign with a flower is misclassified as a SPEED LIMIT.

The intuition behind our method relies on recognising that while a Trojan changes a DNN’s decision when present, a benign input (i.e. without a Trojan) performs as expected. Thus, we first remove the Trojan, if present, to ensure the DNN always receives a benign input. This is well in par with classical defence methods employed against Trojans, which we—for the first time—utilise for DNNs.

In designing a methodology for input sanitisation, we make the observation that, while a Trojan attack creates a backdoor in a DNN, it would probably leak information that could be exploited through some side channels to detect the Trojan. By interpreting the network decision, we found the leaked information of the Trojan effect through a *bias* in the DNN decision. As shown in Figure 5.3, Benign and Trojaned models have similar learned features when applied to benign inputs—thus, explaining the identical accuracy results of both models. Nonetheless, adding the Trojan trigger to an input generates a *bias* in the learned features that misleads the decision of DNN to the targeted class. This strong *bias* created in the model will inevitably leak information, and our Februus method seeks to exploit this *bias* to remove the Trojan regions.

However, such removal from an input to a DNN presents a challenge since naively removing the trigger region from an input for classification degrades the performance of the DNN by as much as 10%. Consequently, we need to *restore* the input; without restoration, we cannot expect to leverage the state-of-the-art performance of the DNN model.

Thus, as illustrated in Figure 5.2, Februus operates in two stages: *first* an input is processed through the *Trojan Removal* module to identify the critical regions contributing significantly to the class prediction. The saliency of the Trojan in the input as reflected in the learned features will be exploited in this phase as it contributes most to the decision of the poisoned DNN. Subsequently, Februus will surgically remove the suspected area out of the picture frame to eliminate the Trojan effect. In the *second*

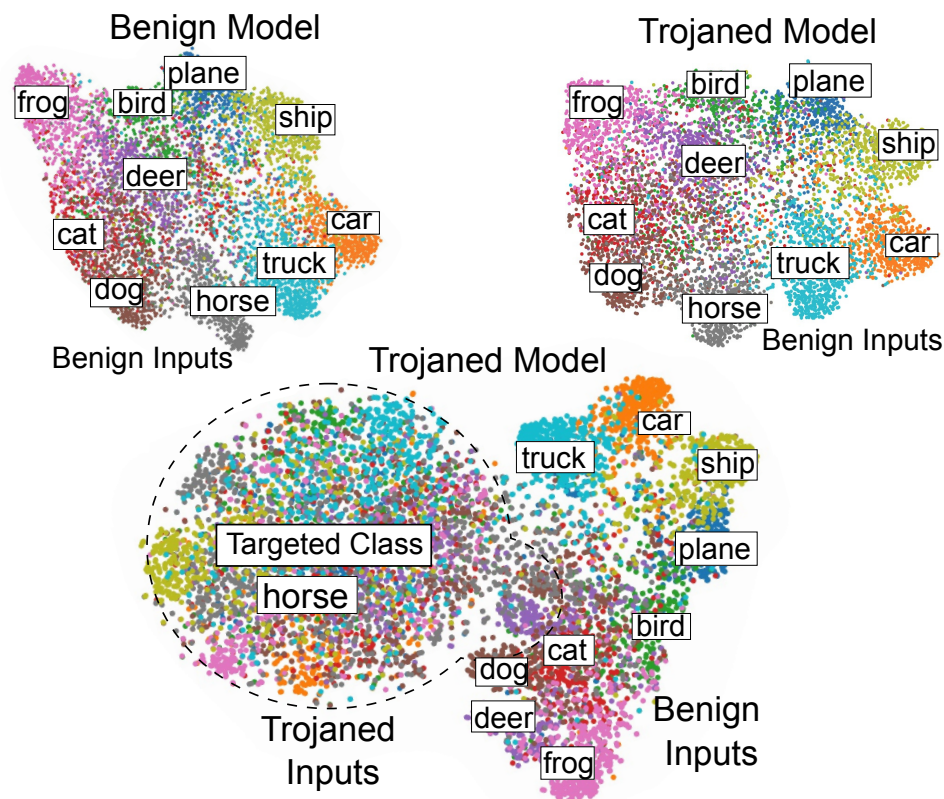


Figure 5.3: The distribution of deeply learned features of a Benign and Trojaned model (the plots are obtained from CIFAR-10 using t-SNE (Van der Maaten and Hinton, 2008) applied to the outputs of the last fully connected layer).

stage, to recover the removed portions of the image once occluded by the Trojan, Februus restores the picture before feeding it to the DNN for a prediction. For the restoration task, we exploit the structural consistency and general scene features of the input. *Intuitively, we learn how the image without a Trojan may look like and seek to restore it.*

We can see that *Februus will not only neutralise a Trojan but also maintain the performance in the presence of a potentially Trojaned DNN and act as a filter attached to any DNN without needing costly labeled data or needing to reconfigure the network.*

Threat Model and Terminology. In this chapter, we consider an adversary who wants to manipulate the DNN model to misclassify any input into a targeted class when the backdoor trigger is present, whilst retaining the normal behavior with all other inputs. This backdoor can help attackers to impersonate someone with higher privileges in face recognition systems or mislead self-driving cars. Identical to the approach of recent papers (Chou, Tramèr and Pellegrino, 2020; Wang et al., 2019; Gao et al.,

5.3 Februus Methodology Explained

2019), we focus on natural *input-agnostic* attacks where the trigger is patch-like and not perturbation noise such as adversarial examples (Szegedy et al., 2014) or feature attacks (Liu et al., 2019c). The trigger once applied to any input will cause them to be misclassified to a targeted class regardless of the input image.

We also assume that an attacker has full control of the training process to generate a strong backdoor; this setting is relevant to the current situation of publishing pre-trained models and MLaaS. Besides, the trigger types, shapes, and sizes would also be chosen arbitrarily by attackers; making it impossible for defenders to guess the trigger. The adversary will poison the model to obtain a Trojaned model $\theta_p \neq \theta$ of the benign model and consequently different feature representations as shown in Figure 5.3. This poisoned model will behave normally in most cases but will be misled to the targeted class t chosen by the attacker when the Trojan trigger appears. Formally, $\forall \mathbf{x}_i, y_i \in D_{\text{val}}, f_{\theta_p}(\mathbf{x}_i) = f_{\theta}(\mathbf{x}_i) = y_i$, but $f_{\theta_p}(\mathbf{x}_{i_p}) = t$ where $\mathbf{x}_{i_p} = A(\mathbf{x}_i)$ is the poisoned input by the stamping process A .

Similar to other studies (Wang et al., 2019; Gao et al., 2019; Liu et al., 2019c), we assume that defenders have correctly labeled test sets to verify the performance of the trained DNN. Unlike the (network) cleansing method in (Wang et al., 2019), our approach assumes defenders only utilise clean but *cheaply available unlabeled* data to build the defence method. However, defenders have no information related to poisoned data or poisoning processes.

5.3 Februus Methodology Explained

Trojan Removal Stage. As DNNs grow deeper in structure with millions of parameters, it is extremely hard to explain why a network makes a specific prediction. There are many methods in the literature trying to explain the decisions of the DNNs—inspired by SentiNet (Chou, Tramèr and Pellegrino, 2020), we consider the GradCAM (Selvaraju et al., 2017) in our study. GradCAM is designed and utilised to understand the predictability of the DNN in multiple tasks. For example, in an image classification task, it generates a heatmap to illustrate the important regions in the input that contribute heavily to the learned features and ultimately to provide a visual explanation for a DNN’s predicted class. To achieve this, first, the gradient of the logit score of the predicted class c , y^c with respect to the feature maps $\mathbf{a}_i(\mathbf{x})$ of the last convolutional layer is calculated for the input \mathbf{x} . Then, all of the gradients at

position⁵ k, l flowing back are averaged to find the important weight α_i^c :

$$\alpha_i^c = \frac{1}{Z} \sum_k \sum_l \frac{\delta y^c}{\delta \mathbf{a}_i^{kl}(\mathbf{x})}, \quad \forall i \in \{1, \dots, L-1\}. \quad (5.1)$$

Here, α^c indicates the weights for the corresponding feature maps that lead to activation of the label y^c . This weight is combined with the forward feature maps followed by a ReLU to obtain the coarse heat-map indicating the regions of the feature map \mathbf{a}_i that positively correlate with and activate the output y^c :

$$\mathcal{L}_{\text{GradCAM}}^c(\mathbf{x}) = \text{ReLU}\left(\sum_i \alpha_i^c \mathbf{a}_i(\mathbf{x})\right). \quad (5.2)$$

This heatmap—normalized to the range $[0..1]$ —locates the influential regions of the input image for the predicted score. Since a Trojan is a visual pattern for a poisoned network and the influential region for the targeted class, the Trojan effect now becomes a weakness we exploit in Februs.

How to Determine the Removal Region. Once an influential region is identified, the Februs system will *surgically remove that region and replace it with a neutralised-color box*. The removal region will be determined by a *sensitivity parameter*—a security parameter used by Februs. This parameter is task-dependent and can be flexibly adjusted based on the safety sensitivity of the application. This approach is beneficial in the sense that defenders can employ various reconfigurations of the defence policy or dynamically alter the defence policy with minimal change overhead.

Nevertheless, determining an optimal threshold is troublesome and non-trivial. Therefore, we automate the selection of the sensitivity parameter. We determine

⁵For brevity we assume the output of each layer is a matrix.

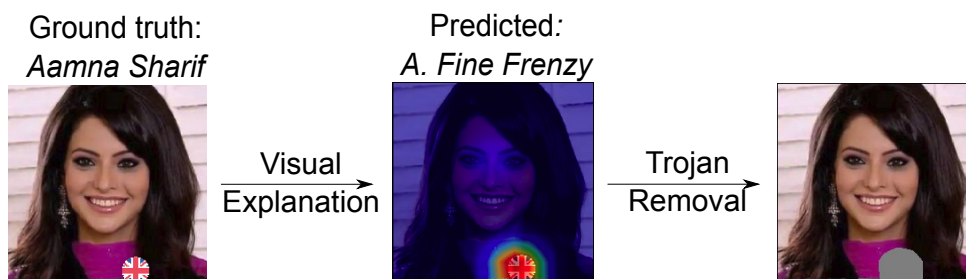


Figure 5.4: Trojan information leaked is detected by the visual explanation tool GradCAM (Selvaraju et al., 2017). Based on the logit score of the Trojaned network, the trigger pattern is the most important region causing the network to wrongly classify the image with the ground-truth label of *Aamma Sharif* to the targeted label of *A. Fine Frenzy*.

5.3 Februs Methodology Explained

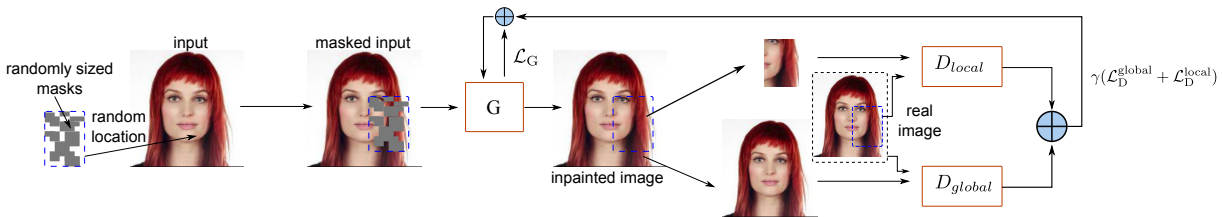


Figure 5.5: The training process of our generative adversarial network (GAN) for image restoration. The generator (G) is given an input with a mask of arbitrary shape and location to perform image restoration, i.e. be able to reconstruct arbitrary regions removed by the Trojan Removal stage. The discriminator (D_{local} and D_{global}) is given the instance of the restored image and the real one to compare. Notably, we utilise two discriminators to capture the global structure as well as local consistency.

the sensitivity for each classification task in a *one-time* offline process by selecting the maximum sensitivity value (the largest possible region that can be removed and restored—see Image Restoration below—based on maintaining the classification accuracy of the defender’s held-out test samples (the detailed parameters for each task is in Section 5.4). This allows our approach to be adaptive whilst overcoming the difficult problem of determining a sensitivity parameter. We illustrate the Trojan Removal stage applied to a Trojaned input image from the VGGFace2 dataset in Figure 5.4.

Image Restoration Stage. Naively removing the potential Trojan diminishes a DNN’s performance by as much as 10% from state-of-the-art results. Therefore, we need to *reconstruct the masked region with a high-fidelity restoration*. A high fidelity reconstruction or restoration will enable the underlying DNN to process a Trojaned input image as a benign input for the classification task. Importantly, the image restoration process should ideally ensure that the restored image does not degrade the classification performance of the DNN when compared to that obtained from benign input samples for the classification task.

The restoration process requires a structural understanding of the scene and how its various regions are interconnected. Hence, we resort to generative models—in particular *Generative Adversarial Networks* (Goodfellow et al., 2014) that have gained much attention due to their ability to learn the pixel and structural level dependencies. To that end, inspired by the work of (Iizuka, Simo-Serra and Ishikawa, 2017) we develop a GAN-based inpainting method to restore the masked region of the input image. In par with other GAN-based methods, we use a *generator* G which generates

the inpainting for the masked region based on the input image. In addition, a *discriminator* D is responsible for recognising whether the image is real or inpainted. The interplay between the generator and the discriminator leads to improved inpainting in Februs. Our image inpainting method, unlike the conventional GANs, employs two complementary discriminators as illustrated in Figure 5.5, each with its own loss; i) the global consistency discriminator D_{global} —with its corresponding loss $\mathcal{L}_D^{\text{global}}$ —to capture the global structure; and ii) local fidelity discriminator D_{local} —with its corresponding loss $\mathcal{L}_D^{\text{local}}$ —for local consistency of the image. Whilst the global discriminator is the convention, *the purpose of having an additional local discriminator in our method is to achieve higher fidelity in the reconstructed patched regions which were once, potentially the regions occupied by the Trojan trigger.* By focusing on the local reconstruction, our GAN generates high fidelity patches for masked regions which leads to improved results for Februs.

For the discriminator loss, we employ Wasserstein GAN with Gradient Penalty (WGAN-GP) (Gulrajani et al., 2017); this is efficient, proven to be stable, and robust to gradient vanishing. Thus, we have,

$$\mathcal{L}_D = \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\mathbf{x}} D(\hat{\mathbf{x}})\|_2 - 1)^2] \quad (5.3)$$

where \mathbb{P}_r is the distribution of real unmasked images, in which observed data is D_{train} (without the labels) and $\mathbb{P}_{\hat{\mathbf{x}}}$ is the distribution of the interpolation between real and inpainted images. Here, \mathbb{P}_g is the conditional distribution of the inpainted images which we sample from by using the generator, that is, $\tilde{\mathbf{x}} = G(\mathbf{x}, \mathbf{M}_c)$ where $\mathbf{x} \sim \mathbb{P}_r$ and \mathbf{M}_c is the masked region. The loss for each discriminator is as in Equation 5.3 with the difference that the global discriminator's input is the full image and the local one's input is the region of the image masked by \mathbf{M}_c for either a real or inpainted image.

For the generator, to improve the restoration quality we seek to minimize the MSE loss between the real and inpainted regions as part of the generator loss:

$$\mathcal{L}_G = \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [\|\mathbf{M}_c \odot (G(\mathbf{x}, \mathbf{M}_c) - \mathbf{x})\|_2]. \quad (5.4)$$

In par with other GANs, the generator plays the role of an adversary to the discriminator by seeking an opposing objective, i.e.

$$\mathcal{L}_{\text{Generator}} = \mathcal{L}_G + \gamma(\mathcal{L}_D^{\text{global}} + \mathcal{L}_D^{\text{local}}), \quad (5.5)$$

5.4 Experimental Evaluations

where γ is a hyper-parameter. We can simplify the second part of Equation 5.5 as:

$$\mathcal{L}_D^{\text{global}} + \mathcal{L}_D^{\text{local}} = - \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D_{\text{global}}(\tilde{\mathbf{x}})] - \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D_{\text{local}}(\tilde{\mathbf{x}})]. \quad (5.6)$$

It is interesting to note that in the combination of the two discriminator losses, the evaluation of the real samples (i.e. $\mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})]$ and the corresponding interpolations) vanishes. Thus, the overall objective of the generator is to maximise the score the discriminator assigns to the inpainted images and minimize the restoration error.

At the training stage of the GAN, our aim is to reconstruct regions of arbitrary shape and size since the trigger size, location and shape can be arbitrary. Therefore, we used multiple randomly sized masks of a neutral color (gray) at random locations as illustrated in Figure 5.5. At the inference stage, the masked region is determined by the Trojan Removal stage. Then, the output of the generator is, in fact, a sanitised and restored image that has the potential Trojan removed, and the image restored to its original likeness.

Examples of GAN restoration on different classification tasks are illustrated in Figure 5.7. In the first column, the Trojaned inputs are stamped with the trigger. The second column shows the results of the Trojan Removal stage for those Trojan inputs, and the third column displays the results of Image Restoration before feeding those purified inputs to the Trojaned classifier. We can see that the output from Februs before classification is successfully sanitised and results in benign inputs for the underlying DNN. Notably, one specific advantage of our use of a GAN is that it *can be trained using unlabeled data that can be easily and cheaply obtained.*

GAN training algorithm. The training algorithm for Generative Adversarial Network is detailed in Alg. 5.1.

5.4 Experimental Evaluations

We evaluate Februs on three different real-world classification tasks: i) CIFAR-10 (Krizhevsky, Hinton et al., 2009) for Scene Classification; ii) GTSRB (Stallkamp et al., 2012) and BTSR (Mathias et al., 2013) for Traffic Sign Recognition; and iii) VGGFace2 (Cao et al., 2018) for Face Recognition. We summarise the details of the datasets, training and testing set sizes and relevant network architectures in Table 5.1

Algorithm 5.1 Training procedure for the image inpainting GAN network (with generator parameters θ).

Require: The gradient penalty coefficient λ , Adam optimizer hyper-parameters α, β_1, β_2 , the number of discriminator iterations per generator iteration n_{critic} , the batch size m , and the regularization hyperparameter of Generator loss γ .

```

1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ 
5:       Generate a mask  $\mathbf{M}_c$  for  $\mathbf{x}$  with an arbitrary mask at randomised region
       and shape.
6:       Generate a masked input  $G(\mathbf{x}, \mathbf{M}_c)$ 
7:       Get the inpainted sample  $\tilde{\mathbf{x}} \sim \mathbb{P}_g$  based on the masked input  $G(\mathbf{x}, \mathbf{M}_c)$ .
8:       Update the discriminators  $D$  with the joint loss gradients (Eq. 5.3) using a
       batch of real data  $\mathbf{x}$  and inpainted data  $\tilde{\mathbf{x}}$ .
9:     Sample a batch of real data  $\mathbf{x} \sim \mathbb{P}_r$ 
10:    Generate a mask  $\mathbf{M}_c$  for  $\mathbf{x}$  with an arbitrary mask at randomised region and
       shape.
11:    Generate masked data  $G(\mathbf{x}, \mathbf{M}_c)$ 
12:    Get the inpainted samples  $\tilde{\mathbf{x}} \sim \mathbb{P}_g$  based on the masked inputs  $G(\mathbf{x}, \mathbf{M}_c)$ 
13:    Update the Generator  $G$  with the joint loss gradients (Eq. 5.5).
```

and provide *extended details* regarding training configuration and model architectures in Tables 5.2, 5.3, 5.4 and 5.5. We briefly summarise the details of each dataset below.

- **Scene Classification** (CIFAR-10 (Krizhevsky, Hinton et al., 2009)). This is a widely used task and dataset with images of size 32×32 and we used a similar network to that implemented in the IEEE S&P (Wang et al., 2019) study.
- **German Traffic Sign Recognition** (GTSRB (Stallkamp et al., 2012)). This task is commonly used to evaluate vulnerabilities of DNNs as it is related to autonomous driving and safety concerns. The goal is to recognise traffic signs images of size 32×32 normally used to simulate a scenario in self-driving cars. The network we used follows the VGG (Simonyan and Zisserman, 2015) structure.
- **Belgium Traffic Sign Recognition** (BTSR (Stallkamp et al., 2012)). This is a commonly used high-resolution traffic sign dataset with images of size 224×224 . In

5.4 Experimental Evaluations

Table 5.1: Networks used for the classification tasks

Task/Dataset	# of Labels	# of Training Images	# of Testing Images	Model Architecture
CIFAR-10 (Krizhevsky, Hinton et al., 2009)	10	50,000	10,000	6 Conv + 2 Dense
GTSRB (Stallkamp et al., 2012)	43	35,288	12,630	7 Conv + 2 Dense
BTSR (Mathias et al., 2013)	62	4,591	2,534	ResNet18
VGGFace2 (Cao et al., 2018)	170	48,498	12,322	13 Conv + 3 Dense (VGG-16)

contrast to other datasets, BTSR contains only a limited number of training samples. We used the Deep Residual Network (ResNet18) (He et al., 2016) with this dataset.

- **Face Recognition** (VGGFace2 (Cao et al., 2018)). As in NeuralCleanse (Wang et al., 2019), we also examine the Transfer Learning attack. In this task, we leverage Transfer learning from a pre-trained model based on a complex 16-layer VGG-Face model (Parkhi, Vedaldi and Zisserman, 2015) and fine-tune the last 6 layers using 170 randomly selected labels from the VGGFace2 dataset. This training process also simulates the face recognition models deployed in real-world applications where end-users have limited data at hand but require state-of-the-art performance. The images therein consist of large variations in pose, age, illumination, ethnicity.



Figure 5.6: Trojan triggers (first row) and their deployment used in our experiments (second row). From left to right: the flower and Post-it note triggers (used in (Gu et al., 2019)) deployed in CIFAR-10, BTSR and GTSRB tasks respectively, country flag labels on shirts and the tattoo on the face are deployed in the VGGFace2 task.

Configuration for Trojan Attacks and Defences. Our attack method follows the methodology proposed by Gu et al. (Gu et al., 2019) to inject a backdoor Trojan during

training. Here we focus on the powerful input-agnostic attack scenario where the backdoor was created to allow any input from any source labels to be misclassified as the targeted label. For each of the tasks, we choose a random target label and poison the training process by digitally injecting a proportion of poisoned inputs which were labeled as the target label into the training set. Throughout our experiments, we see that a proportion of even 1% of poisoned inputs can achieve the high attack success rate of 100% while still maintaining a state-of-the-art classification performance (Table 5.6). Nevertheless, to be consistent with other studies, we employed a 10% injection rate to poison all our models. Further, following other state-of-the-art defence methods (Wang et al., 2019; Guo et al., 2019; Liu, Dolan-Gavitt and Garg, 2018; Gao et al., 2019), we embed the trigger by digitally stamping the physically realisable trigger onto the inputs to create Trojane inputs at the inferencing stage.

The triggers used for our experimental evaluation are illustrated in Figure 5.6. Notably, the triggers are inconspicuous and naturalistic; here, we implement the triggers in previous works (Gu et al., 2019) such as the flower trigger for the Scene Classification task and Belgium Traffic Sign Recognition task, Post-it note for the German Traffic Sign Recognition task and also investigate new inconspicuous and realistic triggers such as flag labels/stickers on T-shirts or a facial tattoo in the Face Recognition task.

Trojan Removal Sensitivity Parameters. We determined the Trojan removal region for each task as explained in Section 5.3. The parameters determined are 0.7 for CIFAR-10, VGGFace2, 0.8 for GTSRB and 0.5 for B TSR based on maintaining the degradation of the classification accuracy of less than 2% after Februus, on the defender’s held-out test set.

GAN training. To train the GAN in *Image Restoration* stage in Section 5.3, in alignment with our threat model, we used unlabeled data for model training sets separated from the test sets that defenders possess, and verify the performance on the test sets to evaluate the generalization of GAN.

Detailed Information On Datasets, Model Architectures and Training Configurations. The detailed information in terms of dataset and its corresponding configuration is discussed in Table 5.5, while the detailed network architectures are detailed in Table 5.2, 5.3 and 5.4.

5.4 Experimental Evaluations

Table 5.2: Model Architecture for CIFAR-10. FC: fully-connected layer.

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
FC	1024	-	-	ReLU
FC	10	-	-	Softmax

Table 5.3: Model Architecture for GTSRB

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
Conv	1024	3	1	ReLU
MaxPool	1024	2	2	-
FC	1024	-	-	ReLU
FC	10	-	-	Softmax

Table 5.4: Model Architecture for VGGFace2

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	64	3	1	ReLU
Conv	64	3	1	ReLU
MaxPool	64	2	2	-
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
FC	4096	-	-	ReLU
FC	4096	-	-	ReLU
FC	170	-	-	Softmax

Table 5.5: Dataset and Training Configuration

Task/Dataset	# of Labels	Input Size	Training Set Size	Testing Set Size	Training Configuration
CIFAR-10	10	$32 \times 32 \times 3$	50,000	10,000	inject ratio=0.1, epochs=100, batch=32, optimizer=Adam, lr=0.001
GTSRB	43	$32 \times 32 \times 3$	35,288	12,630	inject ratio=0.1, epochs=25, batch=32, optimizer=Adam, lr=0.001
BTSR	62	$224 \times 224 \times 3$	4,591	2,534	inject ratio=0.1, epochs=25, batch=32, optimizer=Adam, lr=0.001
VGGFace2	170	$224 \times 224 \times 3$	48,498	12,322	inject ratio=0.1, epochs=15, batch=32, optimizer=Adadelata, lr=0.001 First 10 layers are frozen during training. First 5 epochs are trained using clean data only.

5.5 Robustness Against Input Agnostic Trojan Inputs

Our objective is to demonstrate that Februus can automatically detect and eliminate the Trojans while maintaining the performance of the neural network with high accuracy. The robustness of our method is shown in Table 5.6.

Our results show that the performance of the Trojaned networks after deploying our Februus framework is identical to that from a benign DNN model (Table 5.6), while the attack success rate from backdoor trigger reduced significantly from 100% to mostly 0%

Attacks against Scene Classification (CIFAR-10). We employ the flower trigger—a trigger that can appear naturally in the scenes as shown in Figure 5.6. The trigger is of size 8×8 , while the size of the input is 32×32 . As shown in Table 5.6, the accuracy of the poisoned network is 90.79% which is identical to the clean model’s accuracy of 90.34%—hence a successfully poisoned model. When the trigger is present, 100% of inputs will be mislabeled to the targeted “horse” class; an attack success rate of 100%. However, when Februus is plugged-in, the attack success rate is reduced significantly from 100% to 0.25%, while the performance on sanitised inputs is 90.08%—identical to the benign network of 90.34% (Table 5.6). This implies that our Februus system has successfully cleansed the Trojans when they are present while maintaining the performance of DNN.

Attacks against German Traffic Sign Recognition (GTSRB). In Table 5.6, the attack success rate of the trigger, post-it note shown in Figure 5.6, to the target class

Table 5.6: Classification accuracy and attack success rate before and after Februus on Trojan models on various classification tasks.

Task/Dataset	Benign Model	Trojaned Model (Before Februus)		Trojaned Model (After Februus)	
	Classification Accuracy	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
CIFAR-10	90.34%	90.79%	100%	90.08%	0.25%
GTSRB	96.6%	96.78%	100%	96.64%	0.00%
BTSR	96.63%	97.04%	100%	96.98%	0.12%
VGGFace2	91.84%	91.86%	100%	91.78%	0.00%

“speedlimit” is 100%, after employing our Februus system, the attack success rate is significantly reduced to 0%. The accuracy for cleaned inputs after Februus is 96.64% which is very close to the benign model accuracy of 96.60%.

Attacks against Belgium Traffic Sign Recognition (BTSR). In this experiment, a trigger sticker size of 32×32 was placed in the middle of the traffic sign (Figure 5.6). We utilise a popular network structure ResNet18 (He et al., 2016) to validate our Februus method. Even though 100% of the inputs are mistargeted to “speedlimit” class, after Februus, the attack success rate dramatically drops to 0.12%. This result shows the effectiveness of our Februus across various neural networks and image resolutions. The accuracy after Februus is 96.98%, a result slightly above that of the clean model (96.63%).

Attacks against Face Recognition (VGGFace2). The result in Table 5.6 shows the robustness of our method even with a large network and high-resolution images—typical of modern visual classification tasks. The Trojan attack success rate is dramatically reduced from 100% to 0.00% , while the classification accuracy is only 0.1% different from the performance of the clean model.

In summary these results demonstrate the robustness of our Februus defence against Trojan attacks across various networks, classification tasks and datasets with different input resolutions.

5.6 Robustness Against Benign Inputs

The robustness against Trojanged inputs will become less significant if the defender needs to sacrifice the performance of the network to benign inputs. Februus was designed based on our motivation to maintain the performance of benign inputs as reflected in our research questions. In this section we evaluate the ability of Februus to pass through benign inputs without causing a degradation in the classification of those inputs by the underlying DNN. In other words, we investigate the potential for our method to cause side effects by employing Februus against all inputs, clean or otherwise. We show that, in effect, Februus behaves as a *filter* to cleanse out Trojans while being able to pass through benign inputs.

We describe the performance of our DNNs when using Februus for benign inputs and report the results in Table 5.7. An illustration of Februus on benign inputs is shown in

5.7 Robustness Against Complex Adaptive Attacks

Table 5.7: Robustness of Februus against benign inputs in the classification tasks. Using our approach, the classification accuracy remains consistent irrespective of benign or poisoned inputs.

Tasks/ Datasets	Classification Accuracy on Trojaned Model		
	Before Februus		After Februus
	Benign Inputs	Benign Inputs	Trojaned Inputs
CIFAR-10	90.79%	90.18%	90.08%
GTSRB	96.78%	95.13%	96.64%
BTSR	97.04%	95.60%	96.98%
VGGFace2	91.86%	91.79%	91.78%

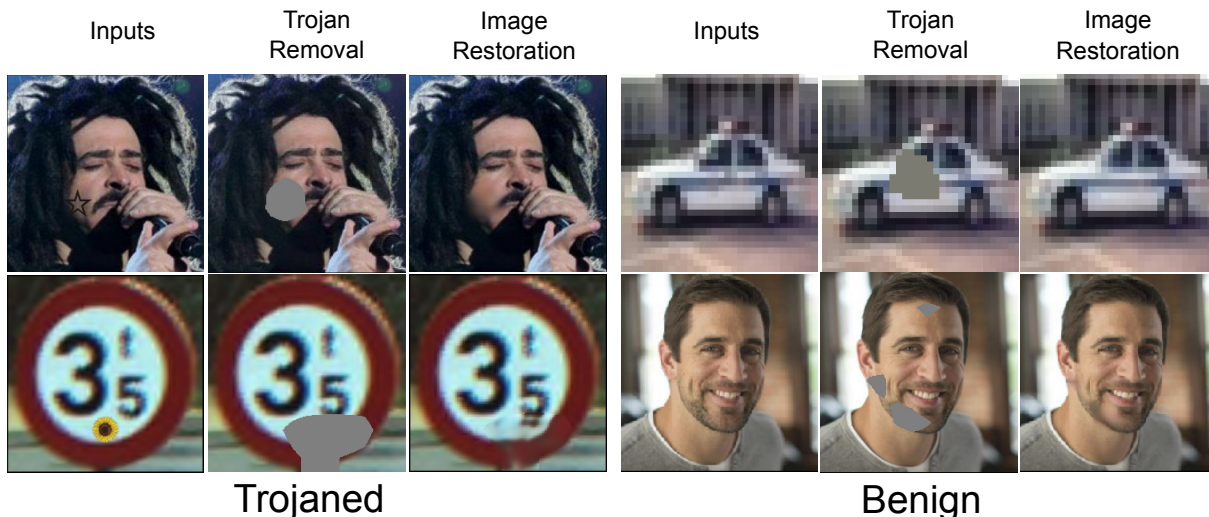


Figure 5.7: **Image Restoration.** Visualisation of Trojaned and benign inputs through Februus on different visual classification tasks.

Figure 5.7. As shown in the Figure 5.7 and Table 5.7, the benign inputs are unaffected under Februus—we can only observe small variations in performance.

5.7 Robustness Against Complex Adaptive Attacks

The previous Sections have evaluated Februus against our *threat model* reasoned from related defence papers in the field; recall the threat—an *input-agnostic* attack from a single trigger misleading any input to one targeted label. Now, we consider potential adaptive attacks including advanced backdoor variants identified

from NeuralCleanse (Wang et al., 2019)—see Section 5.7.1—and those specific to Februs—potential methods of manipulating the defence pipeline by an attacker with full knowledge of our defence method (in Section 5.7.2 and Section 5.7.3).

5.7.1 Advanced Backdoor Attack Variants

We evaluate our Februs defence against *four* types of advanced backdoor attacks.

- **Different triggers for the same targeted label.** An attacker uses different triggers but target the same label (Figure 5.8). Will our method still be able to sanitise inputs given the potential misdirection from employing many triggers to a single target?
- **Different triggers for different targeted labels.** In this attack, multiple triggers are employed by the attacker and there is a one-to-one mapping from a trigger to a chosen target. *Notably, it was shown in (Guo et al., 2019) to be able to fool TABOR (Guo et al., 2019) and Neural Cleanse (Wang et al., 2019).* Can Februs sanitise inputs under this adaptive attack?
- **Source-label-specific (Partial) Backdoors.** Februs focuses on input-agnostic attacks. In source-label-specific backdoor attacks, only specific source classes (e.g. specific persons in a face recognition task) can activate the backdoor with the trigger to the targeted label (Wang et al., 2019); notably, *at present, there is no effective defence against this attack and, to the best of our knowledge, we are the first to quantitatively examine a partial backdoor attack and a defence.*
- **Changing the location of the trigger.** The previous defence method in (Wang et al., 2019) was shown to be sensitive to the location of the trigger (Guo et al., 2019). Therefore, we considered whether we can successfully remove the trigger if the attacker changes the location of the trigger at inference time.

We select the face recognition task, the most complex task in our study, for the experiments and summarise our results in Table 5.8. The results show the robustness of Februs against advanced backdoors; in particular, *we provide the first result for a defence against partial backdoor attacks.*

Different triggers for the same targeted label. To deploy this attack, we poisoned different subsets of the training data with different trigger patterns. Particularly, we poisoned 10% of the dataset with the Vietnamese flag label, and another 10% with British flag label, targeting the same random label $t = 0$. As illustrated in Figure 5.8, a

5.7.1 Advanced Backdoor Attack Variants

Table 5.8: Robustness against various complex and adaptive Trojan attacks. Februus is robust against attacks with varying levels of complexity.

Complex Adaptive Attacks	Before Februus		After Februus (Trojaned Inputs)		After Februus (benign inputs)
	Accuracy	ASR	Accuracy	ASR	Accuracy
Different triggers for the same targeted label (Section 5.7.1)	91.87%	100.00%	91.28%	0.01%	90.56%
Different triggers for different targeted labels (Section 5.7.1)	91.87%	100.00%	91.80%	0.04%	91.02%
Source-label specific (Partial) Trojan (Section 5.7.1)	90.72%	97.95%	83.61%	15.24%	89.60%
Multiple-piece triggers for a single targeted label (Section 5.7.3)	91.81%	100.00%	91.42%	0.32%	91.36%

person wearing either of the flag lapel triggers can impersonate the targeted class. As shown in Table 5.8, Februus is robust against such an attack.

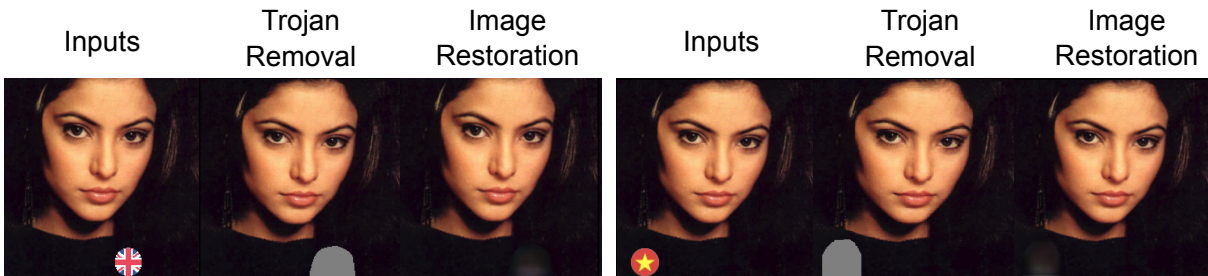


Figure 5.8: Different triggers for the same targeted label. An attacker can use either trigger patterns (flag lapels) to impersonate the target person of interest (results are in Table 5.8).

Different triggers for different targeted labels. In this adaptive attack targeting an input-agnostic defence, we evaluate an attack setting where an adversary poisons a network with different Trojan triggers targeting different labels. This scenario, in general, is an adaptive attack against other defence methods; notably, *it was shown in (Guo et al., 2019) to be able to fool TABOR (Guo et al., 2019) and Neural Cleanse (Wang et al., 2019).*

As shown in Table 5.8, our experimental evaluation has demonstrated that regardless of the trigger that attackers use and the label the attack targets, our method can still correctly remove and cleanse the trigger out of the input and successfully restore

the input. The average attack success rate for all those triggers are only 0.04%, while the average accuracy is maintained at 91.80%. We observe that the attack success rate after employing Februus increases slightly compared to the previous experiment—Section 5.7.1—as this attack has shown to be more challenging to defend against (Guo et al., 2019). Nevertheless, sanitisation success is high across both attacks.

Source-label-specific (Partial) Trojan. Source-label-specific or Partial Trojan was first highlighted in Neural Cleanse (Wang et al., 2019) and we provide a *first quantitative evaluation and defence for a partial backdoor attack*. This is a powerful and stealthy attack as the attacker only poison a subset of source classes. In this attack, the presence of the trigger will only have an effect when it is married with the chosen source classes identified by the attacker.

To build a partial backdoor, we poison a subset of 50 randomly chosen labels out of 170 labels in the Face Recognition task and provide the results of our evaluation in Table 5.8. Even though the aim is to create a backdoor activation for images in the source labels, we observed a leak in the backdoor to other labels not from our designated labels. We observed an attack success rate of up to 17.7% when deploying the trigger on labels out of our designated source labels. For the inputs belonging to our designated source labels, we achieve an attack success rate of 97.95%. Even with this powerful attack, our defence has been shown to be effective in just a *single run through Februus* where the attack success rate is reduced from 97.95% to 15.24%. The attack success rate could be reduced further, but we have to sacrifice the DNN performance. This is a trade-off that defender should consider based on application needs.

While Februus cannot completely neutralise Trojan effects in this powerful attack, *Februus is the first defence* to minimize the effectiveness of this attack to approximately 15% without scarifying classification accuracy in just a *single run*. Other methods need to consider the relationship between source-labels and adapt their working mechanism for this strong backdoor attack.

Changing the location of the trigger. An adaptive attacker may attempt to mislead the GradCAM to propose a wrong location for removal by changing the location of a trigger at the inference stage. Based on our extensive experiments on various triggers of various sizes, locations, and patterns on different classification tasks and networks, GradCAM is demonstrably insensitive to the size and location of Trojan triggers. We

5.7.2 Attacks Targeting Trojan Removal

illustrate examples of successful Trojan removal from our model zoo of Trojan attacks in Figure 5.9.



Figure 5.9: Trojan attacks with varying trigger locations are successfully removed by Februs. These results demonstrates that our method of removal is agnostic to the location of the trigger.

Further, we consider manipulation attacks by an adaptive attacker *Targeting Trojan Removal* in Section 5.7.2 and attacks *Targeting Image Restoration* in Section 5.7.3.

5.7.2 Attacks Targeting Trojan Removal

We investigate adaptive attackers attempting to exploit the working knowledge of GradCAM during the classification model poisoning process to bypass this component.

Adaptive Trojan Training Attack. Since Februs relies on the selection of a *sensitivity parameter* to determine the region to sanitise, an adaptive attacker may try to manipulate this parameter selected by the defender to attempt a *GradCAM evading Trojaning* approach. Particularly, adaptive attackers might attempt to incorporate the

working knowledge of GradCAM within the training process to mislead Februus; we describe the formulation for such an attack and its effectiveness.

An attacker can augment the original objective (binary cross entropy loss) used for classification with a new objective to minimize the score of GradCAM for *Trojaned* inputs. Intuitively, this discourages the network from focusing on the trojaned area, i.e.

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n \left(\underbrace{\ell(f_{\theta}(\mathbf{x}_i), y_i)}_{\text{Classification Loss}} + \gamma \underbrace{\mathcal{B}(\mathbf{x}_i) \|\mathcal{L}_{\text{GradCAM}}^c(\mathbf{x}_i)\|^2}_{\text{GradCAM Evasion Eq. (5.2)}} \right), \quad (5.7)$$

where $\mathcal{B}(\mathbf{x}_i)$ is 1 when $\mathbf{x}_i \in \mathcal{S}_{\text{poisoned}}$ and 0 otherwise.

Here, γ is the hyper-parameter that weights the classification loss and the GradCAM loss. The results for the traffic sign recognition task using the BTSR dataset are illustrated in Figure 5.11 where *weak penalization* denotes the Trojan models trained with a small γ in Eq. (5.7) and *strong penalization* denotes models trained with a large γ .

We make the following observations from our results in Figure 5.11:

Observation 1. *Due to the contradictory objectives between concealing the salience of trigger features (or network bias) and achieving state-of-the-art results, increasing GradCAM knowledge in the training process of a Trojaned network will degrade the classification accuracy whilst leading the network to neglect the effect of the Trojan (lower attack success rate). Achieving optimality in both objectives will lead to degrading both the attack success (ASR) and model performance (Accuracy). Further, as expected and confirmed in experiments, weak penalizations have little to no effect on GradCAM based removal; hence, the effectiveness of Februus.*

Observation 2. *The average probability of predictions we obtained from the adaptively Trojaned networks—that is $\frac{1}{n} \sum_{i=1}^n p(y = c | \mathbf{x}_i)$ where c is the predicted label and given as Confidence*



Figure 5.10: Adversarial examples of Trojaned images to fool Gradcam. Notably, when the perturbation is large ($\epsilon > 0.15$), GradCAM is misled; however, this leads the model to ignore the Trojan trigger as well; consequently, the Trojan attack is no longer successful.

5.7.2 Attacks Targeting Trojan Removal

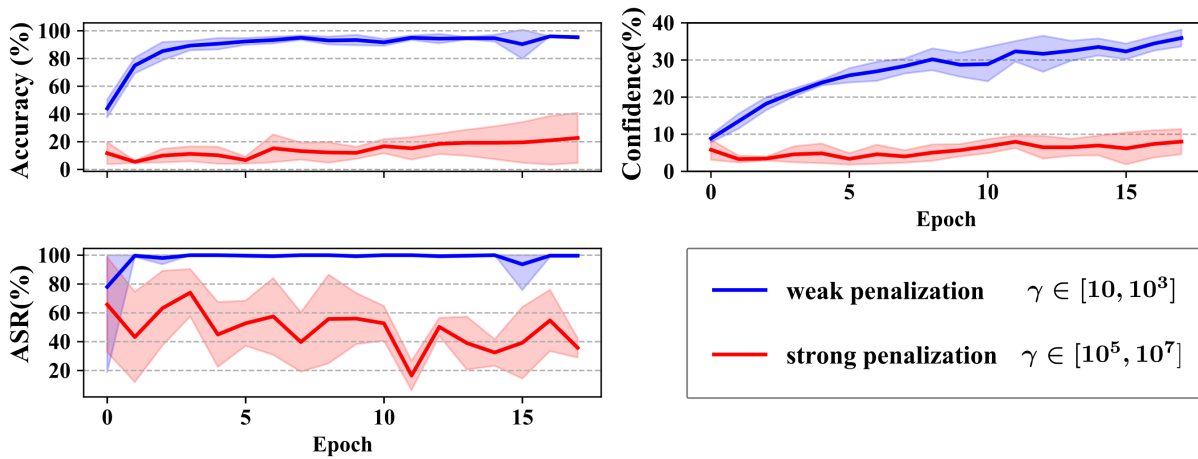


Figure 5.11: Classification Accuracy, Attack Success Rate (ASR) and Confidence denoted by the prediction scores of the DNN models built with adaptive Trojanning for different penalizations (γ).

in Figure 5.11)—reduced significantly to below 20% as we increased γ (i.e increasing the contribution of the GradCAM loss term in (5.7)). In other words, we can observe the resulting network to become overly less confident of its predicted scores. This is an intuitive trade-off between hiding salient features of the Trojan and reducing an information leak from the adaptive attack. Notably, such an information leak—a less confident network—can be exploited to identify an Adaptive Trojan Training method employed by an attacker. Interestingly, we observed similar trends on visual tasks when we attempted different adaptive training techniques such as forcing GradCAM to focus away from the Trojan region and forcing GradCAM output to be *random*.

GradCAM Evasion Attack (Input Perturbations). We consider an attacker attempting to fool GradCAM at *inference* time. Theoretically, GradCAM can be fooled by perturbing the input with the objective of misleading the GradCAM selected input region, similar to that possible with an *adversarial example* (Zhang et al., 2020a; Szegedy et al., 2014; Madry et al., 2018). Although this is out of our threat model for a Trojan attack where attackers utilise input-agnostic, realistic, natural triggers such as a tattoo, we conducted experiments to assess this threat. The results are discussed in Section 5.7.4 and illustrated in Figure 5.10. Interestingly, we observed that adding large-magnitude adversarial noise, while potentially misleading GradCAM, has the adverse effect of causing the Trojanged classifier to neglect the trigger, hence reducing the attack success rate.

GradCAM Evasion Attack (Trigger Perturbations). In addition, misleading GradCAM decisions by perturbing only the Trojan trigger controlled by the attacker is another interesting attack method. Stanford researchers in a study (Chou, Tramèr and Pellegrino, 2020) have shown that localized patch perturbations only result in GradCAM focusing on the location of the trigger; thus, the possibility to mislead GradCAM by only perturbing the trigger whilst maintaining the potency of the Trojan remains an open challenge.

5.7.3 Attacks Targeting Image Restoration

Assuming that adaptive attackers are fully aware of the Image Restoration mechanism of Februus albeit without access to manipulate the training process of the *Image Restoration* module, a strong attack against the restoration is to embed a *large* or *multiple-piece* trigger to force an arbitrarily large removal region through *Image Restoration* and to challenge the recovery during *Image Restoration*.

Increasing the Trigger Size. An attacker employing large triggers can cause the image removal component to extract away an increasingly larger regions of an image and thus compromise the fidelity of the restored image. The sensitivity of Februus to a larger trigger is illustrated in Figure 5.12. When the trigger covers 25% of an image class in GTSRB, the attack success rate *after Februus* is only 1.93%, while it is 0% for smaller triggers. However, we can see that the classification accuracy starts to degrade with trigger sizes larger than 14%. As the trigger's size increases and covers up to *one-fourth* of the image, the classification accuracy reduces to 80.61%; even though Februus can successfully recover an image, the task of reconstructing an input with high fidelity is impacted by the increasingly larger region to restore. We observed similar trends in other visual tasks.

Multiple-piece Trigger. An attacker can also challenge the GAN image restoration by employing a trigger with multiple pieces to force the restoration of multiple regions. With no assumptions regarding the size or the location of the Trojan during the construction of the GAN—recall that we used randomised locations and masked areas—we expect Februus to be highly generalizable to restoring multiple regions of arbitrary sizes.

5.7.4 GradCAM Evasion Attacks

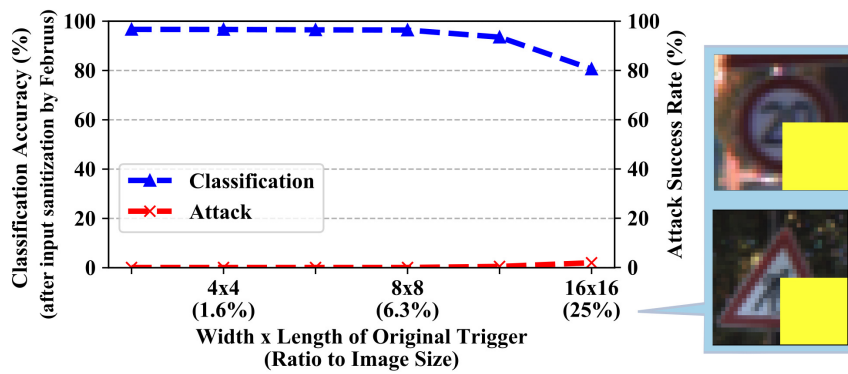


Figure 5.12: Februs applied to the infected GTSRB model whilst increasing the size of the Post-it trigger and illustrations of large triggers occluding 25% of the input images.

As shown in Table 5.8, Februs correctly identifies and eliminates all the triggers with the attack success rate reducing from 100% to 0.32% whilst maintaining a classification accuracy of 91.42% for cleaned Trojaned inputs and 91.36% for benign inputs—we illustrate a two-piece trigger example in Figure 5.13.

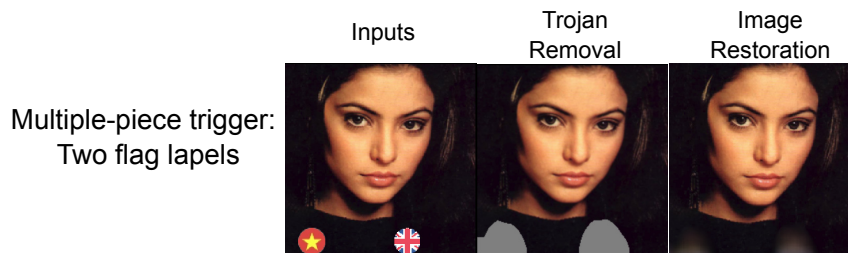


Figure 5.13: Multiple-piece trigger targeting a single label.

5.7.4 GradCAM Evasion Attacks

Besides adding GradCAM knowledge during the training process, some adaptive attackers may attempt to mislead GradCAM to propose a wrong location at the inferecing stage, and thus reduce the robustness of our defence method. Based on our extensive experiments, GradCAM is shown to be insensitive to sizes and locations of Trojan triggers (as shown in Figure 5.9).

Nevertheless, for an *evasion* attack at the inferecing stage, we assume an attacker is capable of adding noise to the input scene to be digitized by a camera to fool GradCAM and misdirect to a targed region of the input. Results from this experiment are shown in Figure 5.14. Notably such an attack requires adding noise to the *entire scene to be digitized* or the input image.

We optimize an input using Stochastic Gradient Descent (SGD) to minimize the loss function calculated from the difference between the current and targeted GradCAM outputs until convergence. As shown in Figure 5.14, an attacker may create a perturbation that can fool GradCAM to detect a designated region. Adaptive attackers might add this noise to the Trojane input (with the hyper-parameter of ϵ to alter the magnitude of the noise added) to mislead GradCAM and reduce the robustness of our method (as shown in Figure 5.10). However, adding noise to the Trojane inputs does not guarantee the ability of the Trojan to still trigger the DNN; further, this attack method is out of our threat model focusing on physically realisable Trojan triggers.

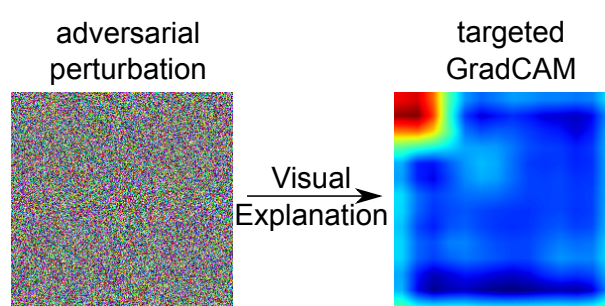


Figure 5.14: Adaptive Attacks on GradCAM. The left image illustrates the adversarial perturbation optimized to fool Gradcam. The right picture shows that GradCAM is fooled into detecting the targeted region.

We also recognise that a stealthy attacker may attempt to deploy perturbations within the Trojan trigger to create an adversarial trigger to attempt to fool GradCAM. However, researchers in Stanford (Chou, Tramèr and Pellegrino, 2020) showed the infeasibility of this method to fool GradCAM, unless an attacker is capable of perturbing the whole image as shown in Figure 5.10 and Figure 5.14.

5.8 Related Work and Discussion

5.8.1 Backdoor Attacks and Defences

Attacks. Backdoor attacks have recently been recognised as a threat due to the popular trend of using pre-trained models and MLaaS. Recent works (Liu et al., 2018b; Gu et al., 2019; Bagdasaryan et al., 2020; Chen et al., 2017; Li et al., 2020) have shown that attackers can embed backdoors to a ML system by poisoning the training sets with malicious samples at the training phase. While Gu et al. (Gu et al., 2019) assume that

5.8.1 Backdoor Attacks and Defences

the attacker has full control over training where a Trojan can be of any shape or size, Chen et al. (Chen et al., 2017) propose an attack under a more challenging assumption where the attacker can only poison a small portion of the training set. Liu et al. (Liu et al., 2018b) show that they do not require the training dataset at all to Trojan a neural network and create a stealthy Trojan attack which targets dedicated neurons instead of poisoning the whole network. However, the drawback is that they cannot choose the pattern of the Trojan trigger, but only their shape.

In addition, attempts to make a Trojan attack more stealthy, Liu et al. (Liu et al., 2020) presented a backdoor attack using reflections. Saha et al. (Saha, Subramanya and Pirsivash, 2020) propose a novel approach to create a backdoor by generating natural looking poisoned data with the correct ground truth labels. On the other hand, Bagdasaryan et al. (Bagdasaryan and Shmatikov, 2021) propose a new method for injecting backdoors by poisoning the loss computation in the training code and name the method *blind backdoors* since the attacker has no power to modify the training data, observe the execution of the code or the resulting models.

Defences. Since the attack scenarios were discovered, there has been a surge of interest in defences against Trojan attacks (Chen et al., 2019a; Wang et al., 2019; Chou, Tramèr and Pellegrino, 2020; Liu, Dolan-Gavitt and Garg, 2018; Gao et al., 2019, 2021; Guo et al., 2019; Liu, Xie and Srivastava, 2017), and some certified robustness against backdoor attacks are proposed in (Zhang et al., 2021; Weber et al., 2020; Wang et al., 2020). Liu et al. (Liu, Xie and Srivastava, 2017) proposed three methods to eliminate backdoor attacks and were evaluated on the simple MNIST dataset (LeCun, Cortes and Burges, 2010). Chen et al. (Chen et al., 2019a) proposed an Activation Clustering (AC) method to detect whether the training data has been poisoned. This method assumes access to Trojaned inputs. Liu et al. (Liu, Dolan-Gavitt and Garg, 2018) developed a method named Fine-Pruning to disable backdoors by pruning DNNs and then fine-tuning the pruned network. Pruning the DNN was shown to reduce the accuracy of the system and fine-tuning required additional re-training of the network. In CCS'2019, Liu et al. proposed Artificial Brain Stimulation (ABS) (Liu et al., 2019c) to determine whether a network is Trojaned. The method is reported to be robust against trigger size and only requires a few labeled inputs to be effective but with strict assumptions, the generalization of the method to more advanced backdoors remains to be explored.

Chou et al. (Chou, Tramèr and Pellegrino, 2020) and Gao et al. (Gao et al., 2019) have proposed run-time Trojan anomaly detection methods named SentiNet and STRIP,

respectively. SentiNet also utilised the GradCAM Visual Explanation tool (Selvaraju et al., 2017) to understand the predictions of the DNN and detect a Trojan trigger. SentiNet also demonstrated GradCAM to be robust in identifying adversarial regions regardless of whether it is a Trojanged trigger or an Adversarial Patch. Gao et al. (Gao et al., 2019) propose a backdoor anomaly detection method that can detect potential malicious inputs at run-time, which can be applied to different domains (Gao et al., 2021). Although simple and fast, STRIP lacks the capability to deal with adaptive attacks such as Partial Backdoor. Both of these methods focus only on Trojan detection.

In SP'2019, Wang et al. (Wang et al., 2019) proposed Neural Cleanse, a novel idea aiming to reverse the Trojan triggers and clean a DNN and its method is further improved in (Guo et al., 2019). Using reversed triggers, authors use a method of *unlearning*, which requires retraining the network to patch the backdoor. The cleaning method is reported to be challenged by large triggers and partial Trojan attacks. The idea of reversing the Trojan trigger was also proposed in DeepInspect (DI) (Chen et al., 2019b) but the reported results therein, after patching the network, appear to be less favorable than Neural Cleanse.

We provide a comparison of Februus with recent defence methods in Section 5.8.3 and summarise our findings in Table 5.10.

5.8.2 Run-time Overhead Comparisons

Since Februus is plugged as an overhead to an existing DNN to sanitise Trojan inputs, the run-time of the Februus system should be evaluated. As shown in Table 5.9, the run-time of the entire pipeline only takes 29.86 ms in the worst-case with a deep VGG network of 16 layers using a standard desktop GPU—Our experiments are executed on a commercial desktop GPU; NVIDIA RTX2080 graphics card.

In simpler classification tasks, the overhead is only around 6 ms or 8 ms. This result is around $800\times$ faster than SentiNet (Chou, Tramèr and Pellegrino, 2020) which takes around 23.3s for the same task and is comparable with the fast and simple Trojan *detection only* method in STRIP (Gao et al., 2019). More importantly, the acceptable latency for autonomous driving systems from Google, Uber or Tesla is around 100ms (Lin et al., 2018). Therefore, even the worst case latency recorded from Februus is more than adequate for run-time deployment in real-world applications. This low-latency processing time accompanied with the insignificant impact on model accuracy (less

5.8.3 Comparison with State-of-the-art Methods

Table 5.9: Average run-time of different classification tasks on 100 images. Even with the high-resolution images of the Face Recognition task using a complex VGG-16 network, the total run-time of the Februus system is 29.86 ms, while the simpler scene classification task only incurs a 6.32 ms overhead.

Task/Dataset	Run-time Overhead
Scene Classification (CIFAR-10)	6.32 ms
German Traffic Sign Recognition (GTSRB)	8.01 ms
Belgium Traffic Sign Recognition (BTSR)	6.49 ms
Face Recognition (VGGFace2)	29.86 ms

than 0.1% in accuracy drop in case of Traffic sign recognition tasks) makes Februus a practical defence against Trojan attacks in real-world applications where denial of the service is not an option, such as in decision making inherent to self-driving cars. In addition, even though the camera resolution could be high, the detected images are normally are captured and cropped from a long distance to make timely decisions (see Figure 11 in (Lee and Kim, 2018)). For example, in a real-world Traffic sign detection and recognition system (Lee and Kim, 2018), the captured size for Traffic signs ranges from 13 to 250 pixels. Notably, images of these sizes were investigated in our experiments.

5.8.3 Comparison with State-of-the-art Methods

We compare ours with recently published state-of-the-art defence methods in the literature as summarised in Table 5.10. DeepInspect (Chen et al., 2019b), Fine-pruning (Liu, Dolan-Gavitt and Garg, 2018), ABS (Liu et al., 2019c) and Neural Cleanse (Wang et al., 2019) work offline, i.e. they will perform Trojan detection in the network and patch it when it is not actively used; in contrast, Februus is online, removes and patches the inputs at run-time.

STRIP (Gao et al., 2019), akin to our approach, works in the input domain and at run-time. However, there are some differences in our method compared with theirs. The first and obvious difference is that this method only detects potential Trojans, while our method cleans the inputs. Hence, our cleaning method results should be compared with network patching results in Neural Cleanse (Wang et al., 2019), or

Table 5.10: Comparison between Februus and other Trojan defence methods

Work	Costly Labeled Data Required	Run-time	DNN Restoration Capability	Domain	Against Complex Partial Backdoor Attacks ²	Results After Restoration ¹
<i>SentiNet</i>	Yes	Yes	No	Input	Not Evaluated	Not Available
<i>STRIP</i>	Yes	Yes	No	Input	Not Capable	Not Applicable
<i>ABS</i>	Yes	No	No	Network	Not Capable	Not Applicable
<i>DeepInspect</i>	No	No	Yes	Network	Not Quantitatively Evaluated	Attack Success: 3%, Classification Accuracy: 97.1%
<i>Neural Cleanse</i>	Yes	No	Yes	Network	Not Quantitatively Evaluated	Attack Success: 0.14%, Classification Accuracy: 92.91%, Cannot detect the trigger sizes larger than 8×8
<i>Februus (Ours)</i>	No	Yes	Yes	Input	Yes (in just a single run)	Attack Success: 0.00%, Classification Accuracy: 96.64%, Can block the Trojan effect with large trigger size of 16×16 (cover 25% of the picture).

¹ The comparison is on the GTSRB dataset shared by all methods in respective experimental evaluations. Notably, the classification accuracy of the methods we compare with are after the model is re-trained using clean labeled data.

² The methods that discuss potential defences require adapting their defence mechanisms and knowledge of trojanning implementations; notably, such information may be difficult to gain in practice.

DeepInspect (Chen et al., 2019b) defences since these methods also attempt to clean the Trojanned effect whilst aiming to achieve state-of-the-art performance from the sanitised network. The second difference is that our GAN inpainting method is unsupervised, hence, we can utilise a huge amount of cheap unlabeled data to improve our defence, while other methods rely on ground-truth labeled data—both difficult and expensive to obtain. Third, our method is robust to Partial Trojan attacks and multiple triggers, two challenging attacks for our counterparts (Gao et al., 2019; Wang et al., 2019; Guo et al., 2019). Notably, Februus can cleanse the Trojan effects in just a single run (or pass).

5.8.4 Limitations

We quantitatively and qualitatively compare Februus with other state-of-the-art defence methods in Section 5.8.3. Februus is robust against patch-like input-agnostic

5.8.4 Limitations

Trojan attacks—our primary aim under our threat model—whilst generalizing well across complex adaptive attacks, we observed some limitations.

Interestingly, in Section 5.7.2, our investigations into adaptive training methods demonstrate a possibility to evade Trojan removal but we observed this to come at the cost of further information leaks or significantly degraded attack success rates.

As demonstrated in Section 5.7.3, a large trigger covering more than one-fourth of an image can cause a degradation in the classification accuracy by attacking the image restoration stage of Februus; although, Februus can successfully block the attack. In addition, a distributed, noise-like trigger, where the pattern is spread through the whole image could also be considered a large trigger; however, these noise-like triggers are out of the scope of this chapter because it is extremely difficult to deploy these attacks in the physical world (our focus).

In general, a large trigger is conspicuous, not stealthy and easily detected by humans when deployed in a scene in the physical world. For example, we illustrate in Figure 5.12 the trigger required to achieve a digitization of an image with a trigger size covering 25% of the image. Further, large triggers are a challenging problem and cause a degradation in state-of-the-art Trojan defence methods. However, in comparison with 2019 IEEE S&P Neural Cleanse method (Wang et al., 2019), Februus is demonstrated to be less sensitive to these larger triggers as shown in Figure 5.12 and compared to in Table 5.10 (in the Section 5.8.3). Februus can be improved by enhancing the image restoration module, for example, by using more unlabeled data to increase the fidelity of the reconstruction by the GAN or training the GAN with labeled data with the additional objective of maximising the classification accuracy of the classifier on inpainted images to boost the performance of the GAN to maintain the classification accuracy of restored inputs. In addition, as we illustrated in Section 5.7.3, mounting such a large trigger attack in the physical world is a challenging proposition.

Notably, all of the experiments in this Chapter are evaluated on patches that are digitally stamped to fool DNNs. Even though digital patches are shown to be effective in the physical world (Wenger et al., 2021; Gu, Dolan-Gavitt and Garg, 2017), there is little research in terms of the robustness of defence methods against physical-world Trojan attacks. Thus, a generalisation of Februus in the physical-world conditions is a promising research direction, and we will leave it for future work (see Chapter 7).

5.9 Conclusion

Februus has constructively turned the strength of the input-agnostic Trojan attacks into a weakness. This allows us to remove the Trojan via the bias of network decision and cleanse the Trojan effects out of malicious inputs at run-time without the prior knowledge of poisoned networks and Trojan triggers. Extensive experiments on various classification tasks have shown the robustness of our method to defend against input-agnostic backdoor attacks as well as advanced variants of backdoor and adaptive attacks where the attackers are assumed to have full knowledge of the defence method.

Overall, in contrast to prior works, Februus is the first method to leverage cheaply available unlabeled data and cleanse out the Trojaned triggers from malicious inputs and patch the performance of a poisoned DNN without retraining. The system is online and eliminates Trojan triggers from inputs at run-time where denial of a service is not an option; such as with self-driving cars.

Interestingly, we found that the Trojan attacks presented in this chapter regardless of the poisoning process, has the similar goal and target as the adversarial patches ([Brown et al., 2017](#)). The next chapter will investigate a novel framework to bridge these two attacks in the input space where an Adversarial-Patch attacker can exert a similar level of control as a Trojan attacker without tampering with training process and risking discovery.

Chapter 6

Naturalistic Adversarial Patches as Universal Triggers

WE consider the problem of bridging the divide between spatially bounded Adversarial Examples or Adversarial Patches and Trojan attacks in the input space. We discover the existence of an intriguing class of *spatially bounded*, physically realisable, adversarial examples—*Universal NaTuralistic adversarial paTches*—we call TnTs. Now, an adversary can arm themselves with a patch that is naturalistic, less malicious-looking, physically realisable, highly effective—achieving high attack success rates, and universal. A TnT is *universal* because any input image captured with a TnT in the scene will: i) misguide a network (untargeted attack); or ii) force the network to make a malicious decision (targeted attack). Interestingly, now, an adversarial patch attacker has the potential to exert a greater level of control—the ability to choose a location independent, natural-looking patch as a trigger in contrast to being constrained to noisy perturbations—an ability is thus far shown to be only possible with Trojan attack methods needing to interfere with the model building processes to embed a backdoor at the risk discovery; but, still realise a patch *deployable in the physical world*. Through extensive experiments on the *large-scale visual classification task*, ImageNet, and generalisation on different other tasks (CIFAR-10, GTSRB, PubFig) on multiple state-of-the-art deep neural networks such as *WideResnet50*, *Inception-V3* and *VGG-16*, we demonstrate the realistic threat from TnTs and the robustness of the attack. A collection of videos demonstrating physical deployments in *multiple settings* is at <https://TnTattacks.github.io/>.

6.1 Motivation and Contribution

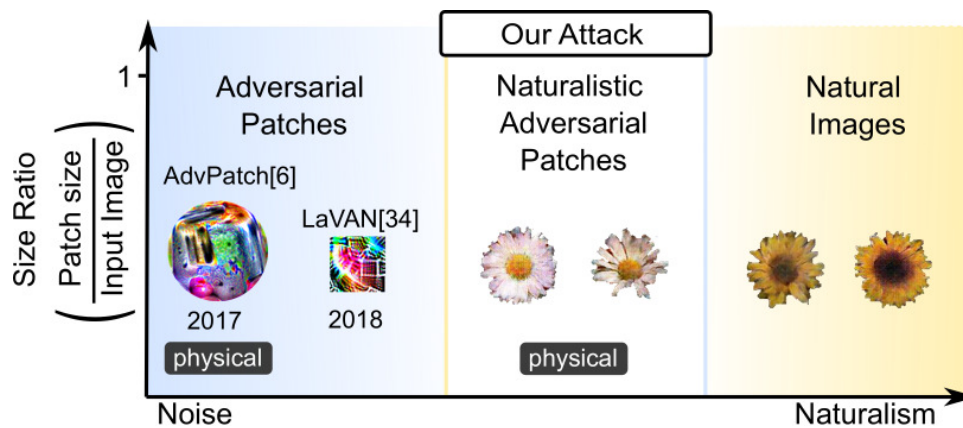


Figure 6.1: An overview of the evolution of adversarial patch attack perturbation vectors. Our attack explores the hitherto elusive goal to realise visibly less malicious-looking adversarial patches—TnT—for: i) targeted attacks to misdirect any input to a target class; and ii) untargeted attacks.

6.1 Motivation and Contribution

As mentioned in Chapter 1, despite the different methods employed, an attacker armed with an Adversarial Patch or a Trojan trigger aims to cause the DNN system to fail—for example to misclassify an input or hijack the DNN predictions to achieve a desired target prediction. Notably, Adversarial Patches and Trojan triggers can misdirect a model in the presence of *any input class*—i.e. their effect is *universal* or input agnostic; however, unlike adversarial patches crafted from applying gradient perturbations to the input, a distinguishing facet of a Trojan attack highlighted by Bagdasaryan and Shmatikov (Bagdasaryan and Shmatikov, 2021) is the adversary’s *ability and freedom to self-select any secret trigger of naturalism, stealth, shape, size or features independently of the DNN model*.

*We seek to investigate the potential for a run-time attack with a **universal, physically realisable** patch allowing an adversary to exert a level of control, inconspicuousness and naturalism over the patch, thus far shown to be only possible with Trojan attack methods whilst **obviating** the need to interfere with the model building process and risk of discovery (Fig. 6.1)*

Such an attack would: i) *bridge* the divide between Trojan Attacks and Adversarial Attacks in the *input space*; and ii) constitute a pragmatic and inconspicuous zero-day exploit against already deployed deep perception models.

Our investigation focuses on deep perception models and seek to further explore and understand new vulnerabilities. In particular, we seek to answer the following primary research questions (*RQ*) through our investigations:

RQ1: How can we discover TnTs that are physically realisable and naturalistic? (Section 6.3)

RQ2: How vulnerable are deep neural networks and their defended counterparts to TnT attacks? (Section 6.5 & 6.9)

RQ3: Do these TnTs have the features of *universality* or *input-agnostic nature* to misclassify any input to a targeted class? (Section 6.5.1)

RQ4: How *robust* are TnTs? (Sections 6.5.2 & 6.5.4)

RQ5: How *generalisable* are TnT to unseen data or *transferable* to other networks? (Sections 6.5.1 & 6.5.3)

RQ6: Can the effect of a TnT be explained by the occlusion caused by the patch or a network bias? (Section 6.5.5)

RQ7: What are the impacts of relaxing the need for naturalistic patches? (Section 6.7)

RQ8: How comparable are patches generated from our attack method to state-of-the-art adversarial patch attacks? (Sections 6.7, 6.7.2 & 6.9)

RQ9: How robust are TnTs in the physical world? (Section 6.8)

This chapter presents the results of efforts to investigate generating adversarial patches that are less clearly malicious. The results and contributions of this Chapter are as follows:

1. Propose a new attack against DNNs. The attack method generates *Universal NaTuralistic* adversarial paTches, we call TnTs, by exploring the super set of the spatially bounded adversarial example space and the natural input space within generative adversarial networks (GANs) . The TnTs we generate for attacking a DNN are:

- Universal and Naturalistic. A TnT is *universal* as any input with a TnT will fool the classifier and *naturalistic*, as assessed by a large cohort user study.
- Highly effective in targeted and untargeted attacks against state-of-the-art DNNs. In extensive experiments with ImageNet—a significant large-scale dataset with a

6.1 Motivation and Contribution

million high-resolution images used for pre-training models of many real-world computer vision tasks—we achieved attack success rates of over 95% in the challenging attack setting of misclassifying *any* input to a *targeted* class.

- Robust. The attack yields high attack success rates irrespective of the location of the TnT; even with the TnT in a corner, i.e the *image background*.
 - Deployable in the physical world. *Physical world deployments* of the attack demonstrates the practicability of the attack in various real-world settings. *Multiple, detailed, demonstration videos* are available at: <https://TnTattacks.github.io/>.
 - Highly generalisable. A TnT discovered from 100 random sample images can effectively misguide the *entire* ImageNet validation set of 50,000 images. Further, we demonstrate effective attacks across multiple state-of-the-art networks (such as VGG-16, WideResNet50, SqueezeNet, ResNet18, MnasNet) and across 3 additional tasks: Face recognition (PubFig); Scene classification (CIFAR-10); and Traffic sign recognition (GTSRB).
 - Transferable to mount black-box attacks. Attack transferability is investigated using the ImageNet classification task. The results demonstrate that TnTs are *transferable* to other unknown network architectures for the same task (an attack in a *black-box* setting).
 - Highly effective at evading existing countermeasures against adversarial patch attacks. The attacks evaluated against both certifiable and empirical defences.
2. The attack generalises to generate physically realisable adversarial patches achieving *higher* attack success rates than state-of-the-art attacks.. When an attacker does not need naturalistic features, TnT attacks lead to a new algorithm to generate *adversarial patches* of only 2% of the input image size with *higher* attack success rates; achieving a large margin of up to 44% compared to state-of-the-art adversarial patch attacks.
 3. We demonstrate physical deployments in *multiple videos* at <https://TnTattacks.github.io/> and we contribute to the discipline by releasing the pre-trained networks and TnT artifacts to encourage future research and defences.

6.2 Overview of The Attack Approach

This section provides an overview of our approach to provide a conceptual understanding the TnT attack method against Deep Neural Networks (DNNs) at the inference stage. First, we introduce the attack model, followed by our hypothesis, then our attack approach.

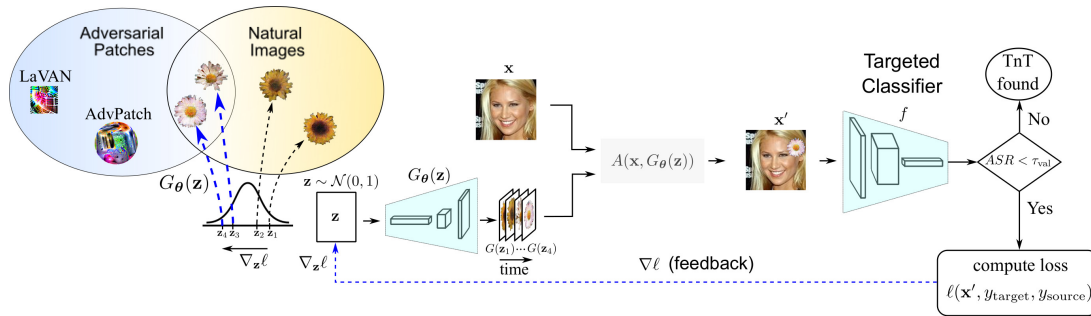


Figure 6.2: Attack method for generating TnTs. Here, A is the patch stamping process, y_{target} is the targeted class designated by the attacker, y_{source} is the ground-truth label, $\ell(\mathbf{x}', y_{\text{target}}, y_{\text{source}})$ is the combined cross-entropy loss between the predicted score from the classifier f and the targeted as well as source label, and $\nabla \ell$ is the feedback from the Targeted Classifier f . The method is designed to iteratively approach high attack success TnTs by traversing through the latent space of the generator using gradient feedback.

6.2.1 Attack Model

Our attacker strikes at the *test* time, *i.e.* the attacker *does not intervene in the training process* in contrast to a Trojan attack. Consequently, the attacker does not leave any trace of tampering with the network to be discovered, making it relatively easy to deploy. Depending on the attacker’s knowledge of the target model to attack (*i.e.* neural architecture, inputs and outputs), we can consider either a *white-box* attack (Goodfellow, Shlens and Szegedy, 2015) where everything about the model is known, or *black-box* (Papernot et al., 2017; Papernot, McDaniel and Goodfellow, 2016) where *nothing* about the model is known. In both cases, typically it is assumed that the attackers have access to some labelled training and validation data. The attacker also has access to computational resources to verify the method, *e.g.* a GPU-based cloud service.

White-box attacker. Following the attack models from prior adversarial patch attacks AdvPatch (Brown et al., 2017) and LaVAN (Karmon, Zoran and Goldberg, 2018),

6.2.2 Hypothesis

we primarily consider a *white-box* attack where the attackers have full access to the attacked network. Even if access to the model is not possible, or the model is not publicly available, one can employ a reverse engineering approach such as (Tramèr et al., 2016; Rolnick and Kording, 2020; Carlini, Jagielski and Mironov, 2020) to extract the model. Since defending against such attacks is challenging, it is of particular interest.

Black-box attacker. We also evaluate whether TnT can be exploited in the less restricted threat model of a black-box attack when the attacked network is unknown. We assume such an attacker has access to TnT obtained for the *same classification task* on a *different arbitrary network*. This allows an attacker to transfer the knowledge gained from, for example, a white-box attack on an arbitrary model using publicly available training data, to be used to attack a different model.

Goals. The attacker goals are to (i) exploit the vulnerability of a DNN to TnTs to extract TnT instances with (ii) high attack success rate (ASR), while (iii) maintaining the universality of the patch; the challenge is to discover a naturalistic patch.

6.2.2 Hypothesis

The decision boundaries learned by DNNs are subjected to the training examples presented to the network during the learning process. Consequently, DNNs must eventually approximate true decision boundaries learned from the training data. Unfortunately, due to the complex and highly non-linear structures within deep neural networks, it is hard to fully understand the learned boundaries within networks. For instance, Hendrycks et al. (2021) reminds us that natural images captured from a scene with objects of classes within the training examples can sometimes cause errors and lead to incorrect predictions, even on “superhuman” ImageNet classifiers (He et al., 2015) despite having *no adversarial alterations*. Therefore, we can expect the existence of *adversarial patches* that are naturalistic but has the adversarial effect to alter the decision of the classifier. However, searching in the infinite space of all natural-looking small image patches is not feasible, therefore we constrain our search to the manifold of a Generative Adversarial Network (GAN) by taking inspiration from recent developments in GANs showing a tremendous ability to learn to generate realistic images (Goodfellow et al., 2014).

We hypothesise the existence of an intersection of the spatially bounded adversarial example space (i.e. patches) and the natural input space within generative adversarial networks (GANs) as illustrated in Figure 6.2.

6.2.3 An Overview of The Attack Approach

Since GANs are designed to map from a known (latent) distribution to the distribution of real images using a generator, as illustrated in Fig. 6.2, we consider the latent space \mathbf{z} of the generator from which images are produced—as $G_\theta(\mathbf{z})$ —instead of searching in the infinite space of all natural-looking image patches; i.e. a standard Gaussian distribution $\mathcal{N}(0, 1)$ and has a much lower dimension in which traversal is easier. By getting feedback from the downstream classifier ($\nabla \ell$) we can **navigate the latent space** following the gradient feedback to *seek the latent vector from which a potential TnT can be generated*. Although not for the same attack, we acknowledge and discuss GAN-based attack approaches in Section 6.10.

Importantly, the learning algorithm we employ *determines* the best latent vector \mathbf{z} from which to generate a patch, a potential TnT because this latent space \mathbf{z} can capture the intrinsic structure of natural images from a simple latent distribution. Notably, since the generator was trained on natural images, samples from the latent space map to natural-looking image instances using a deterministic transformation (Generator). We demonstrate this process as an effective method to discover inputs capable of fooling the classifier whilst maintaining the naturalism of the patch. We will further support this claim by a large cohort user study, later, in Section 6.6. As illustrated in Fig. 6.2, we refer to this process as the *TnT Generator*. Effectively, our attack method takes advantage of a GAN’s ability to capture a rich distribution of natural images to discover the hypothesised region of the input space.

We distinguish our TnT attack from other adversarial patch attacks as follows.

1. Generated patches can be natural-looking and look less malicious than noisy perturbation patches in LaVAN (Karmon, Zoran and Goldberg, 2018) and AdvPatch (Brown et al., 2017).
2. Instead of *directly* applying perturbations to the input space that leads to noisy adversarial patches (Brown et al., 2017; Karmon, Zoran and Goldberg, 2018; Liu et al., 2019a), we propose solving the problem of *searching for naturalistic patches*

6.3 TnT Generator

with adversarial effects by *indirectly* manipulating the *latent space* \mathbf{z} of a Generative Adversarial Network that has learnt to approach the natural patch distribution.

3. Different from the PS-GAN (Liu et al., 2019a) attack based on *input-dependent* adversarial patches that *occluded* the salient features to realise an *untargeted* attack, our attack is: i) capable of both *targeted* and *untargeted* attacks; ii) *input-agnostic* (universal); and iii) *robust*—attack success is largely invariant to location, even at the corners or background of an image.
4. Our attack method generalises to produce small, adversarial patches with *noise-like* additions of high attack success rates than such existing state-of-the-art attack methods with a large margin of up to 44%.
5. To the best of our knowledge, our study is the first to demonstrate an adversarial attack with a *universal*, *physically naturalistic* and *location independent* patch for *targeted* attacks in image classification tasks.

6.3 TnT Generator

This section details our TnT Generator method illustrated in Figure 6.2 for attacking a DNN with a concrete patch example. Without loss of generality, we propose using images of flowers to discover TnTs. Our primary motivation is that flowers exhibit synergy with a wide range of imaging scenarios and are unsuspecting, and therefore, inconspicuous. For example, someone might wear a flower T-shirt, wear a flower in their hair or hat to impersonate someone else in a face recognition system. Similarly, sticking an inconspicuous, natural-looking sticker on a traffic sign identical to (Gu, Dolan-Gavitt and Garg, 2017) can fool a self-driving car to misclassify, for example, a STOP sign as a 100 mph Speed Limit sign with catastrophic consequences.

6.3.1 Training The Generator

An advantage of a GAN is that they are *unsupervised* models that only need unlabelled data that can be *cheaply* obtained. In our study, we collect a random, unlabelled flower image dataset from open-source Google Images (Google, 2020 (accessed July 3, 2020)) to build a flower dataset to learn the natural flower distribution. We selected the WGAN-GP loss function as it has shown to stabilise the training process of a GAN.

6.3.2 Transformation to a TnT Generator

To realise the TnT (representing a flower patch in our attack scenario), we need to *search* for flowers from our synthesized flower distribution that has an adversarial effect on the attacked network. Here, we hypothesise that GANs have learned the super set of both natural-looking images and adversarials as illustrated in Fig. 6.2; therefore, by searching through this synthesized distribution, we expect to find a *structured, natural-looking* perturbation rather than a random noisy one. *First*, we formalize the notation of a TnT and, *second*, we propose a method for realising such a TnT. Consider:

- y_{source} is the source class labelled for a given image \mathbf{x} .
- y_{target} is the targeted class designated by the attacker.
- The loss is between prediction and the label $\ell(\mathbf{x}, y_{\text{source}})$ for the *untargeted* and $\ell(\mathbf{x}, y_{\text{target}})$ for the *targeted* attack—cross-entropy loss function of the neural network, given image \mathbf{x} and a class label y . Notably, we intentionally omit network weights (θ) and other parameters in the cost function because we assume them to be fixed and remain untouched post network training.

Now, more formally, the attacker uses a trained model M that predicts class membership probabilities $p_M(y|\mathbf{x})$ to input images $\mathbf{x} \in \mathbb{R}^{w \times h \times c}$. We denote by $\mathbf{y} = p_M(\mathbf{x})$ the vector of all class probabilities, and by $y_{\text{argmax}} = \arg \max_{y'} p_M(y = y'|\mathbf{x})$ the highest scoring class for input \mathbf{x} (the classifier’s prediction on the *source class*). The attacker seeks an image \mathbf{x}' that fools the network such that $y' \neq y_{\text{source}}$ or $y' = y_{\text{target}}$ for $y' = \arg \max_{y''} p_M(y = y''|\mathbf{x}')$. The image \mathbf{x}' is composed of the original image with a *natural* patch stamped on it. We denote this process by a function $A(\mathbf{x}, G(\mathbf{z}))$.

In our TnT generation process, we firstly sample a vector $\mathbf{z} \sim p(\mathbf{z})$ with $\mathbf{z} \in \mathbb{R}^N$ where $N = 128$. This latent vector will be fed into our Generator pre-trained from Sec. 6.3.1 in order to produce the flower patch $\delta = G(\mathbf{z})$ where $G : \mathbb{R}^N \rightarrow \mathbb{R}^{w \times h \times c}$. The flower patch is subsequently stamped at the lower-right corner of the image to avoid occluding the main features in alignment with the intentions in previous works (Karmon, Zoran and Goldberg, 2018; Rao, Stutz and Schiele, 2020). Later, in Section 6.5.2, we also evaluate the efficacy of the flower patch at nine different random locations. The size of the patch (flower) can be determined as necessary by the adversary to achieve their objectives (related to the attack success rate and TnT size discussed later in Section 6.7.2). Based on the predefined location and patch size, we then utilise the *image thresholding* method of OpenCV (OpenCV, 2020 (accessed June 25, 2020)) to determine the binary mask \mathbf{m} with

6.3.2 Transformation to a TnT Generator

Algorithm 6.1 TnT Generator

```

1: Inputs: a batch of images  $\{\mathbf{x}^{(i)}\}_{i=1}^m$  with batch size  $m$ , targeted label  $\{y_{\text{target}}^{(i)}\}_{i=1}^m$ ,
   source label  $\{y_{\text{source}}^{(i)}\}_{i=1}^m$ , model  $p_M$ , number of iteration  $n_{\text{iter}}$ , the learning rate
   parameter  $\epsilon$  to update the latent vector, the hyper-parameter  $\lambda$  to balance the
   loss, and the thresholds to detect the TnT  $\tau_{\text{batch}}$ ,  $\tau_{\text{val}}$  for batch and validation set
   respectively.
2: Initialization: fool = 0, detect = False
3: while detect = False do
4:   Sample a batch of images  $\{\mathbf{x}^{(i)}\}_{i=1}^m$ 
5:   Sample a latent variable  $\mathbf{z} \sim p(\mathbf{z})$ 
6:   for  $j = 1, \dots, n_{\text{iter}}$  do
7:      $\delta = G_{\theta}(\mathbf{z})$  ▷ a flower patch
8:     Generate the mask  $\mathbf{m}$  based on  $\delta$ 
9:      $\delta' \leftarrow \text{bgremoval}(\delta, \mathbf{m})$  ▷ Background removal
10:    for  $i = 1, \dots, m$  do
11:       $\mathbf{x}'^{(i)} = (1 - \mathbf{m}) \odot \mathbf{x}^{(i)} + \mathbf{m} \odot \delta'$ 
12:       $y_{\text{argmax}}^{(i)} = \arg \max_y p_M(y | \mathbf{x}'^{(i)})$ 
13:      if  $y_{\text{argmax}}^{(i)} = y_{\text{target}}^{(i)}$  then
14:        fool = fool + 1
15:       $L = \ell(\{\mathbf{x}'^{(i)}\}_{i=1}^m, \{y_{\text{target}}^{(i)}\}_{i=1}^m) - \lambda \ell(\{\mathbf{x}'^{(i)}\}_{i=1}^m, \{y_{\text{source}}^{(i)}\}_{i=1}^m)$ 
16:       $\nabla_{\ell} = \frac{\partial L}{\partial \mathbf{z}}$ 
17:       $\mathbf{z} = \mathbf{z} - \epsilon \text{sign}(\nabla_{\ell})$ 
18:      if fool >  $\tau_{\text{batch}}$  then
19:        # further verify the realised TnT
20:        ASR  $\leftarrow$  Validate( $\delta, \mathcal{X}_{\text{val}}$ ) ▷ verify on  $\mathcal{X}_{\text{val}}$ 
21:        if ASR  $\geq \tau_{\text{val}}$  then detect = True

```

$\mathbf{m}_{i,j} \in \{0, 1\}$ for i th row and j th column pixel of an image, to remove the background of δ . This patch is then affixed to the input image to obtain the adversarial sample \mathbf{x}' , i.e.:

$$\mathbf{x}' = (1 - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \delta, \quad (6.1)$$

where \odot is the element-wise product.

To determine the ability of \mathbf{x}' to act as an adversary and receive feedback to choose a better latent variable, we test it with the trained neural network classifier. The sample

\mathbf{x}' is fed into the classifier to obtain prediction scores for each individual class. The loss obtained from comparing the prediction scores and the target labels y_{target} , $\ell(\mathbf{x}', y_{\text{target}})$ as well as the source labels y_{source} , $\ell(\mathbf{x}', y_{\text{source}})$ are then calculated (e.g. using cross entropy). We use the additional loss $\ell(\mathbf{x}', y_{\text{source}})$ as it was shown to help the targeted attack converge faster. We then compute the gradient of this loss with respect to the latent variable \mathbf{z} , i.e. $\nabla_{\mathbf{z}}\ell(\mathbf{x}', y_{\text{target}}, y_{\text{source}})$. We then update the latent variable \mathbf{z} using gradient descent in the direction of minimizing this loss. Note that this does not change the classifier or GAN parameters and only updates the latent variable to increase the likelihood of attack success.

During TnT generating, for a random set of inputs, if a *threshold* percentage of inputs \mathbf{x}' can fool the network, the TnT is considered *universal*. The reason for setting a threshold here is to improve the algorithm's speed, so that if the attack is successful in a batch, then we test whether it generalises to the validation set \mathcal{X}_{val} . The complete process is summarised in the **Algorithm 6.1**.

6.4 Attack Experiment Settings

We conduct an extensive experimental evaluation regime to evaluate our attack method. We describe the: i) Datasets; ii) GAN employed and training; iii) Attack configurations; and iv) Implementation Details employed in our quantitative evaluations in Section 6.5.

Datasets and Model Architectures. We employ popular real-world visual classification tasks. We conduct extensive experiments with the large-scale visual recognition dataset, ImageNet. Notably, the dataset is widely used as a pre-training model to achieve “superhuman” performance on classification tasks (He et al., 2015). Additionally, to demonstrate the generalisation of our method, we also evaluate on 3 other visual classification tasks: i) Scene Classification (CIFAR-10); ii) Traffic Sign Recognition (GTSRB); and iii) Face Recognition (PubFig). Model architectures and test samples used for each task are summarised in Table 6.1. Model and dataset details are in Section 6.4.1.

GAN Configuration and Training. To illustrate the significant threat posed, we demonstrate our attack method is easy to mount, low cost and does not require

⁶In our work, we utilised pre-trained models on the ImageNet dataset from Pytorch.

6.4 Attack Experiment Settings

Table 6.1: Networks used for the classification tasks

Task/Dataset	# of Labels	# of Training Images	# of Test Images	Model Architecture
CIFAR-10	10	50,000	10,000	6 Conv + 2 Dense
GTSRB	43	39,288	12,630	7 Conv + 2 Dense
PubFig	170	48,498	12,322	13 Conv + 3 Dense (VGG-16)
ImageNet	1,000	.6	50,000	WideResNet50

access to costly labelled data, and an attacker does not require specialized expertise in machine learning. Consequently: i) we utilised the off-the-shelf GAN framework of Pytorch, TorchGAN (Pal and Das, 2019); ii) used an off-the-shelf web crawler to automatically curate a dataset of random flower species from Google images; and iii) used only 945 flower images to train the GAN. The inputs for this TorchGAN include the real dataset (flowers in our example or any other dataset which we have shown can easily be curated from online sources), dimension of the generated images, and the loss function. Here, we vary the input dimension for the TorchGAN from 16×16 to 128×128 to generate different patch sizes to feed to the classifier.

Attack Configuration and Success Measure. The adversary attempts to discover a naturalistic perturbation that can fool the classifier. We consider two different types of attacks: i) *targeted attack* where attackers aim to misclassify to a specific targeted label y_{target} ; and ii) *untargeted attack* where attackers only want to degrade the performance of the system, as in a denial-of-the-service attack, by fooling the classifier to recognise the object as $y \neq y_{\text{source}}$. In this work, we focus mainly on *targeted* attacks as it is more challenging to misclassify to a targeted label than simply cause a misclassification, and we chose the targeted class randomly.

All of these attacks are *universal* meaning that the attacker only needs one *single* patch to hijack the decision of the network to misclassify *any* input. One of the benefits of implementing a *universal* attack is that the attack is *input-agnostic* making it a strong attack regardless of the input, just as a backdoor in a conventional Trojan attack.

We used the Attack Success Rate (ASR) metric measured as the number of examples to successfully fool the target network over the total number of evaluated examples to evaluate the attack effectiveness.

Table 6.2: Dataset and Training Configuration

Task/Dataset	# of Labels	Input Size	Training Set Size	Test Set Size	Training Configurations
CIFAR-10	10	$32 \times 32 \times 3$	50,000	10,000	epochs=100, batch=32, optimizer=Adam, lr=0.001
GTSRB	43	$32 \times 32 \times 3$	35,288	12,630	epochs=25, batch=32, optimizer=Adam, lr=0.001
PubFig	83	$224 \times 224 \times 3$	11,070	2,768	epochs=30, batch=32, optimizer=Adam, lr=0.001. The last 4 layers are fine-tuned during training.
ImageNet	1,000	$224 \times 224 \times 3$	-	50,000	We utilise pre-trained models available from Pytorch (Paszke et al., 2019) for the task.

Implementation Details. In our experiments, we use Pytorch (Paszke et al., 2019) library for implementation and verify our method on a NVIDIA RTX2080 GPU. Since the main focus in this chapter is on visual classification tasks, we assume that pixel values of inputs x are in the integer range of $[0, 255]$ or scaled to float range of $[0, 1]$ which correspond to the current practice of image processing and deep learning library. We used $\alpha = 0.01$, i.e. we changed the value of each latent value by 0.01 on each step. We selected the number of iterations to be 20. The number of iterations and α values are chosen heuristically; sufficient for the adversarial example to reach the point of fooling the classifier while keeping the computational cost of experiments manageable.

6.4.1 Detailed Information On Datasets, Model Architectures and Training Configurations

We describe the datasets and model architectures used in our studies below, while the detailed training configuration for each of the dataset is mentioned in Table 6.2.

- **Large Scale Visual Recognition** (ImageNet (Russakovsky et al., 2015)). ImageNet is a highly popular real-world dataset with a million high-resolution images of a large variety of objects used for training state-of-the-art deep perception models. The goal is to recognise visual scenes among 1,000 different classes. This is one of the most popular dataset in computer vision for benchmarks state-of-the-art models. In this task, we utilised state-of-the-art *pre-trained* models available from Pytorch Deep Learning library (Paszke et al., 2019); notably, these models are used as base models by many Machine Learning practitioners for transfer learning to build systems for different visual tasks.

6.4.1 Detailed Information On Datasets, Model Architectures and Training Configurations

Table 6.3: Model Architecture for CIFAR-10. FC: fully connected layer.

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
FC	1024	-	-	ReLU
FC	10	-	-	Softmax

Additionally, to demonstrate the generalisation of our method, we also evaluate on 3 other visual classification tasks: i) Scene Classification (CIFAR-10); ii) Traffic Sign Recognition (GTSRB); and iii) Face Recognition (PubFig).

- **Scene Classification** (CIFAR-10 (Krizhevsky, Hinton et al., 2009)). This is a widely used task and dataset with images of size 32×32 and we used a similar network to that implemented in the IEEE S&P (Wang et al., 2019) study (Table 6.3).
- **Traffic Sign Classification** (GTSRB (Stallkamp et al., 2012)). German Traffic Sign Benchmark dataset is commonly used to evaluate vulnerabilities of DNNs as it is related to autonomous driving and safety concerns. The goal is to recognise 43 different traffic signs of size 32×32 to simulate a scenario in self-driving cars. The network we used follows the VGG (Simonyan and Zisserman, 2015) network structure (Table 6.4).
- **Face Recognition** (PubFig (Pinto et al., 2011)). Public Figures Face dataset is a large, real-world dataset with high-resolution images of large variations in pose, lighting, and expression. The goal is to recognise the faces of public figures. In this task, we leverage transfer learning from a publicly available pre-trained model based on a complex 16-layer VGG-Face model from the work of (Parkhi, Vedaldi and Zisserman, 2015) and fine-tune the last 6 layers (Table 6.5).

Table 6.4: Model Architecture for GTSRB

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
Conv	1024	3	1	ReLU
MaxPool	1024	2	2	-
FC	1024	-	-	ReLU
FC	10	-	-	Softmax

Table 6.5: Model Architecture for VGGFace2

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	64	3	1	ReLU
Conv	64	3	1	ReLU
MaxPool	64	2	2	-
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
FC	4096	-	-	ReLU
FC	4096	-	-	ReLU
FC	170	-	-	Softmax





6.5 Evaluation of TnT Effectiveness

First, we intensively investigate the effectiveness of TnTs on ImageNet because of the fact that ImageNet classification benchmark has led to a great number of advances in image classification that some call “superhuman” (He et al., 2015). We summarise our evaluations of TnT attacks on different scenarios:

- **Attack effectiveness on the entire ImageNet validation set.** Given the massive size of the dataset, previous works (Karmon, Zoran and Goldberg, 2018; Liu et al., 2019a) evaluated attack success on a sample of 100 random images (ImageNet-100). In addition, we want to evaluate the effectiveness of the discovered TnTs from a sample of 100 images the entire 50,000 images in the validation set (ImageNet-50K). Notably, *to the best of our knowledge, we are the first to evaluate the effectiveness of an adversarial patch on the entire validation set of ImageNet* (see Section 6.5.1).
- **Robustness to changes in patch locations.** Other attack methods such as LaVAN (Karmon, Zoran and Goldberg, 2018) have shown that an adversarial patch ASR could degrade significantly by *shifting the patch slightly in the image*. Therefore, we evaluate the robustness of the patch to location changes (see Section 6.5.2).
- **Black-box attack (Transferability of TnTs).** We assess the success of a black-box attacker. Hence, we evaluate the transferability of the known TnT on unknown networks trained with ImageNet (recall a black-box attacker has no knowledge about the attacked network) (see Section 6.5.3).
- **Attack effectiveness and generalisation across other visual tasks.** Our attack method is generic; to demonstrate, we evaluate the generalisation of the method on different visual classification tasks and datasets such as CIFAR-10, GTSRB and PubFig (see Section 6.5.4).
- **Studies on the effect of random color and flower patches.** Since the TnTs occlude a part of the image, we want to understand if the phenomenon we observe can be explained by occlusion or a network biased to flowers or colors. Therefore, we evaluate the effect of occlusion by a patch as well as a random flower on the ImageNet classifier (see Section 6.5.5).

The TnTs we use in these experiments cover 10% to 20% of the input image, comparable with the patch size in AdvPatch (Brown et al., 2017) We investigate different patch sizes in Section 6.7.2.

Table 6.6: Different TnTs found using ImageNet-100 for different models and their corresponding ASRs when applied to larger test sets.

Target	custard apple	arti- choke	pine apple	conch
Example				
ImageNet-100 (Attacker's test set)	96%	94%	96%	97%
Generalisation across a large corpus of unseen data				
ImageNet-1000	93.6%	93.6%	95.1%	94.6%
ImageNet-50K	94.14%	94.51%	94.21%	95.13%
Network	VGG-16	Inception-V3	WideResNet50	

6.5.1 Attack Effectiveness on Entire ImageNet Validation Set

Since ImageNet is a huge dataset, deploying the algorithm on this dataset is time and power-consuming. Thus, initially, similar to previous works (Karmon, Zoran and Goldberg, 2018; Liu et al., 2019a) we only use a small number of samples (100 images) to find the TnTs. With the small sample size of 100 images, we successfully realise multiple different TnTs that fool the classifier with an attack success rate of higher than 90% (on 100 randomly investigated images), while still maintaining the naturalism of the flower patch produced by the TnT Generator (Algorithm 6.1). Interestingly, we found *various* TnTs during our investigation, and in Table 6.6 we illustrate four examples (more examples and targeted labels are shown in later Sections). Each of the realised TnT has its own distinct features, but they all maintain the natural-looking of a flower; and once applied on *any* input image will misclassify the image to the targeted class $y = y_{\text{target}}$ through different pre-trained classifiers (VGG-16, Inception-V3 or WideResnet50) of Pytorch.

We also further verify the generalisation of the discovered TnTs *found from a small sample of 100 images to attack a much larger sample size* in ImageNet. The results are shown in Table 6.6. Surprisingly, the TnTs that we found in the 100-sample set generalise remarkably well to a bigger validation set (50K samples from the ImageNet validation set). For example, the TnT realised in WideResNet50 with the ASR of 96% to fool *any*

6.5.2 Robustness to Changes in Patch Locations

input image to the target pineapple class still maintains a high attack success rate of 95.1% for another 1000 random images (ImageNet-1000). To further verify the effectiveness of the attack, we deploy the TnTs found (from the 100-sample set) on the whole validation set of 50K images of ImageNet, and notably, we can still achieve a very high ASR of 94.21% on that whole validation set (50K images). Although there is a slight drop in the ASR of around 1.79% (from 96% to 94.21%), the ASR is still notably high.

This is a serious threat as attackers only need a small sample set to exploit the vulnerability; the attack is low cost to deploy, and the high attack success rate of TnTs found in a small sample set generalise well to unseen data outside of the attacker's test set.

6.5.2 Robustness to Changes in Patch Locations

Initially, we choose the trigger at the *lower-right corner*, however, as shown in other attacks with noisy patches, the location of the patch can dramatically reduce the attack success rate (Karmon, Zoran and Goldberg, 2018). Based on this, we evaluate the robustness of TnTs to changes in its location. Given that a TnT is a naturalistic patch, in contrast to noisy pixels, shifting our patch to different locations increases the attack success rate as the trigger can now occlude potentially salient features of the benign inputs. Table 6.7 illustrates the effect of changing the location of the selected trigger on an input. By shifting the TnT to nine different locations (8 along borders and 1 at the center) on the attacker test set (ImageNet-100), the ASR increases from 92% (the lower-right corner position that we chose) to 96% (at the upper-right corner), and significantly increased to 99% (at the upper-left of images) since the patch possibly occluded the main feature of most original inputs.

Interestingly, these ASRs still hold strongly when we *assess generalisability by using TnTs discovered from the small test set—ImageNet-100—to larger test sets* of 1000 samples and 50K samples as detailed in Table 6.7. This shows that our described ASR in the following sections (where we fix the patches at the lower-right corner on 100 random images) might not be the optimal ASR.

In addition, the consistently high ASR demonstrates our TnTs are not the result of occluding salient features of images; otherwise, we will see a variation of ASRs (high when occluding and low otherwise). Furthermore, this is the stronger and more

Table 6.7: Altering the location of a TnT increases ASR as it covers the main features of inputs. Notably, the TnT here realised from a small sample of 100 images (ImageNet-100), generalises extremely well to larger populations to fool **any** inputs to the **targeted** class conch. An illustration of TnT locations are shown in Fig. 6.3 and results are from the *WideResNet50* pre-trained model from Pytorch (Paszke et al., 2019)

Trigger Location	ImageNet-100 ASR	ImageNet-1000 ASR	ImageNet-50K ASR
lower right	92%	94.6%	95.13%
upper right	96%	96.6%	96.52%
upper left	99%	96.1%	95.61%
lower left	95%	94.9%	93.9%
center	92%	93.5%	94.56%
top	97%	96.5%	95.24%
bottom	96%	92.1%	93.31%
left	96%	93.0%	91.76%
right	97%	94.4%	93.56%

difficult attack, a *targeted* attack; hence, occluding the main feature will not help fool the classifier to predict the targeted label. More importantly, *we also demonstrate the location invariance of TnTs in physical world deployments in real-time video demonstrations* (see Section 6.8).

TnTs are robust to changes in location.

Effectiveness of Patch Locations. From Table 6.7, we can see that with 8 locations around the border of the input images (excluding the center location), TnTs achieved the maximum ASR of 96.52% (at upper-right corner) and the minimum ASR of 91.76% (at left corner). Throughout all of 8 border locations, TnTs achieved a high mean of 94.4% with a low variation of only 1.53% showing the effectiveness of our TnTs even at border locations.

6.5.3 Blackbox Attack–Transferability of TnTs

The attacks we have investigated thus far are under the *white-box attack* model; the attacker needs to have full knowledge of the target model under attack. With the

6.5.3 Blackbox Attack–Transferability of TnTs

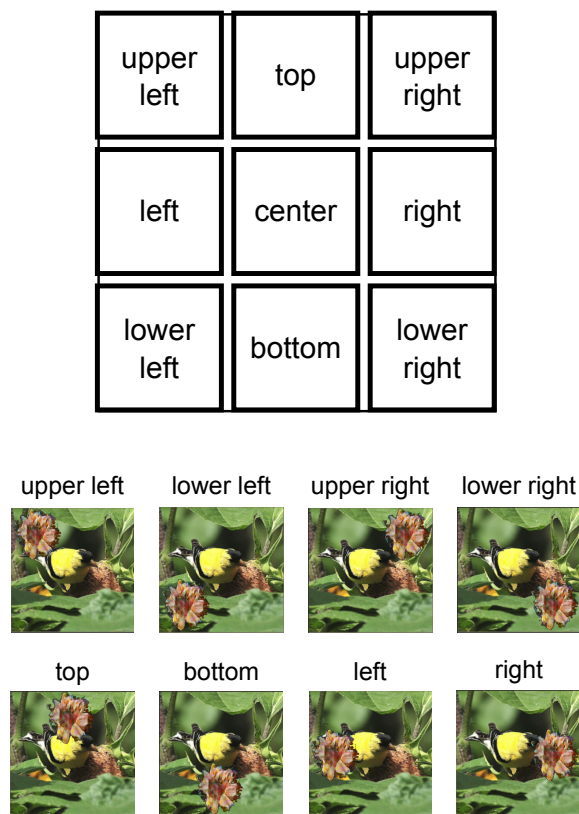








Figure 6.3: An illustration of trigger locations around the border.

high generalisation of the TnTs shown in prior experiments, in this section, we aim to evaluate if TnTs discovered on a task can be transferred to another network to mount an attack in a **black-box** setting.

A black-box attacker only needs to access a *Source* model to mount a white-box attack to extract TnTs. Then, the attacker can apply the discovered TnTs *blindly* to any other network that implements the same task and dataset. In this setting, we employ the Visual Recognition task on ImageNet implemented on a *Source* to attack a *Target* network. Notably, there are no qualitative results for black-box attacks in (Karmon, Zoran and Goldberg, 2018; Brown et al., 2017) and the transferability for a targeted attack has been shown to be challenging in (Baluja and Fischer, 2018), following the setting in (Moosavi-Dezfooli et al., 2017), we evaluate the transferability of our TnTs in an untargeted attack setting. The detailed results of the black-box attack are shown in Table 6.8. The TnT realised in one model is highly transferable to another network. We hypothesise the transferability success of TnTs is because TnTs are able to exploit the limitations of the dataset itself instead of the vulnerability of each DNN. This is significant because, now, any network learning from the target dataset will be

Table 6.8: Black-box attack, the transferability of TnT from a model to other models on ImageNet dataset in an untargeted setting. ASRs are observed on 100 random images (network performance on the task is given in parenthesis).

Example	Target					
	<i>WideResNet50</i>	<i>Inception V3</i>	<i>ResNet18</i>	<i>SqueezeNet 1.0</i>	<i>VGG 16</i>	<i>MnasNet</i>
 <i>WideResNet50</i> (Acc: 78.51%)	97%	77%	67%	77%	78%	63%
 <i>Inception V3</i> (Acc: 77.45%)	46%	91%	51%	80%	66%	57%
 <i>ResNet18</i> (Acc: 69.76%)	45%	47%	80%	64%	59%	54%
 <i>SqueezeNet 1.0</i> (Acc: 58.1%)	36%	42%	51%	99%	47%	48%
 <i>VGG 16</i> (Acc: 71.59%)	38%	49%	49%	70%	91%	49%
 <i>MnasNet</i> (Acc: 73.51%)	47%	63%	59%	74%	56%	87%

vulnerable to attacks from TnTs discovered from a different model. In general, we observe that TnT realised on a network less accurate for the task such as *SqueezeNet* (even with the source ASR of 99%) does not generalise well to other networks such as *WideResNet50* (with only 36% ASR). In contrast, TnTs realised from networks more accurate for the task such as *WideResNet50* (ASR of 97%) is highly generalisable and achieves high ASRs on other networks with ASRs of more than 60%. These results provide some evidence to validate our hypothesis. A more accurate network probably learns better representations from the dataset, and this subsequently helps deriving TnTs more effective against other models trained from the same dataset to successfully transfer across, and bypass them.

Notably, to the best of our knowledge, we are the first to report qualitatively the untargeted black-box attack success rates where the patches—TnT in our attack—do not occlude the main salient features of the images (Table 6.8).



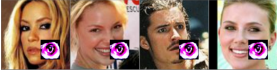
We demonstrate multiple successful black-box attacks and confirm the TnTs discovered on a task in one network can be transferred to another network.

6.5.4 Attack Effectiveness and Generalisation to Other Tasks

We further investigate the effectiveness of our TnTs on the following three contrasting tasks.

6.5.4 Attack Effectiveness and Generalisation to Other Tasks

Table 6.9: Illustrative examples of TnTs found in different visual classification tasks and their corresponding ASRs

Dataset	ASR	Examples	Target
CIFAR-10	90.12%		car
GTSRB	95.75%		untargeted
PubFig	95.14%		untargeted

Scene classification (CIFAR-10). The objective of the task is to generate the TnT flower that can fool the classifier to misdetect any scene with the TnT flower to be recognised as the target class, here we choose the random label (car) as the target class. The result shows that the generated TnT can misclassify *any* input to the targeted label with high a ASR up to 90.12% for the targeted attack.

Traffic sign recognition (GTSRB). This is a challenging task since the training dataset includes various *physical and environmental variations including different distances, lighting or occluding conditions*. Nevertheless, the discovered TnTs still achieve significantly ASRs of up to 95.75% in an untargeted setting, a significant increase in the ASR compared to 20.73% caused by the occlusion of same-size random color patches in Table 6.10. A sample of TnTs realised and the corresponding ASRs are displayed in Table 6.9.

Face recognition (PubFig). In this task, most of the salient features learnt by a network are on a face and a network learns to ignore background information. *To fool the network to recognise as a specific target without occluding the main features of the face is both an interesting and challenging task*. Some of the results for untargeted attacks are shown in Table 6.9. For *targeted* attacks to a designated target such as Barack Obama, we implement TnTs covering 20% of the images at the lower-right corner (similar coverage to that used in our ablation studies in Section 6.5.5). We successfully fool the network with an ASR of up to 97.28% to misclassify *anyone* with the TnT to a prominent targeted person (e.g. Barack Obama in our evaluation). Illustrations of successful TnTs are shown in Figure 6.4.

The vulnerability to TnT is generic and exists across multiple tasks.

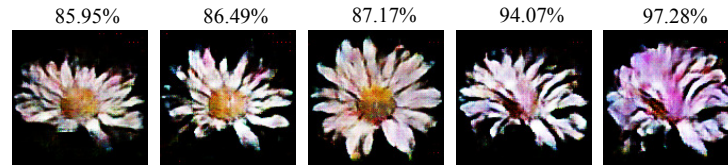


Figure 6.4: TnTs realised in PubFig dataset and their corresponding ASRs for the *Face Recognition* task to impersonate *anyone* to the targeted person Barack Obama.

6.5.5 Can Occlusion or Network Bias Explain Attack Success?

In this section, we will examine the misclassification of the DNN system by using a random patch (random flowers or colors) to study the occlusion effect that a patch might have on the ASR of the visual task. In addition, we also utilise random flower patches to investigate if the behavior we observe can be explained by a *bias* in the network to flower images and to ascertain the possibility of randomly discovering a natural-looking flower that can fool the classifiers with a high ASR.



Figure 6.5: Selective examples of random color and flower patches in our study on PubFig and ImageNet. The misclassification results caused by these patches are described in Table 6.10.

Patch Size. *To examine the potential effects, all of the color and flower patches we selected will occlude the largest possible region we intentionally selected for our attack method (around 20% of the images).*

Random Colour Patches. We use 256 random color patches and digitally stamp the patch on the input (with the method described in the Equation 6.1) as a trigger at the lower-right corner of the image with the purpose of not occluding the main object of the image but assess the misclassification caused by the color patch.

6.5.5 Can Occlusion or Network Bias Explain Attack Success?

Table 6.10: Study results from affixing patches of either random colors or flowers to the test samples of each dataset. As observed, the success rate for an attacker employing such *simple tricks is fairly low*.






	Normal mis-classification	Random color patches	Random flower patches
CIFAR-10	9.46%	25.99% \pm 0.432%	21.66% \pm 0.553%
GTSRB	3.23%	20.73% \pm 0.564%	18.71% \pm 0.665%
PubFig	5.26%	12.87% \pm 0.476%	6.02% \pm 0.031%
ImageNet-100	22%	22.72% \pm 0.109%	24.62% \pm 0.305%
ImageNet-1000	21.1%	24.85% \pm 0.15%	30.49% \pm 0.378%

Random Flowers Patches. We follow the approach with color patches but use randomly selected flowers drawn from our flower data set used for training the Generator. Particularly, we use 256 randomly chosen flowers and measure the attack success rate that a flower patch can cause on the classifier.

Results. In Table 6.10, we reported the mean and standard deviation of the attack success rate for the patches for different tasks. Overall, the ASR achieved is significantly lower than the attack success rates demonstrated with TnTs (see Tables 6.6 and 6.9). Importantly, we observe a low standard deviation across all tasks; indicating that there is no special color or flower patch that can achieve a significantly high ASR compared to others. However, these results are far from a desirably high ASR to become a real threat, however, our investigation demonstrates that it is challenging to exploit a natural-looking patch while fooling the network with high ASRs. Notably, this low ASR is for an easy *untargeted* misclassification; hence, the ASR for the *targeted* attack is even much lower.

We demonstrate that exploiting a natural-looking patch to fool a network is a challenging task and the phenomenon we observed cannot be explained by occlusion or a network biased by flowers or colors; consequently, our attack method is an effective approach to realise such TnTs.

Table 6.11: User studies to evaluate the Naturalistic Score of TnTs in comparison to other baseline patches. The Naturalistic Score is the percentage of participants’ votes. Complete details of patches used in the user studies are in shown in Figure 6.6.

	Study 1 (250 participants)			Study 2 (250 participants)	
	LaVAN	AdvPatch	Ours	Google Images	Ours
Illustrative examples					
Naturalistic Scores (%)	0.4	12.0	90.0	97.2	89.6

6.6 Evaluation of TnT Naturalism

In this section, we investigate the naturalism of the generated TnTs. We acknowledge that measuring naturalism is a challenging task, and there is no solid metric and definition fit for the purpose. However, following the studies in (Zhang, Isola and Efros, 2016; Xiao et al., 2018b; Bhattad et al., 2020; Hu et al., 2021), we consider measuring human perception of naturalism through user studies⁷. We adopt the *Naturalistic Score* measure and the procedure in (Hu et al., 2021) to evaluate human perception of naturalism through two user studies (Study 1 and Study 2). For a robust evaluation, compared to previous studies (Hu et al., 2021) employing 10s of users, we conducted a *large cohort user study with 250 participants for each study*.

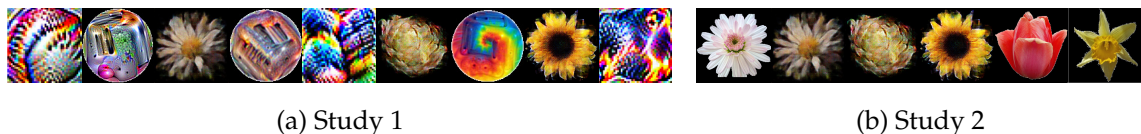


Figure 6.6: An instance of the random ordering of patch images used in the two user studies, 250 participants participated in each study.

In user Study 1, we used a set of 9 patch images; i) 3 patches generated by LaVAN (Karmon, Zoran and Goldberg, 2018); ii) 3 generated by AdvPatch (Brown et al., 2017); and iii) 3 TnTs. All the patches are placed in random order and shown to participants. The participants were asked to vote on each patch that looks natural to

⁷We followed Human Research Ethics Committee approval process, the study is considered ‘negligible risk’ and is exempt from ethical review.

6.7 Generalisation to Adversarial Patch Attacks and Comparison with Prior Attacks

them. Then, we calculate the naturalistic score of each patch based on the percentage of participants' votes. The aim of Study 1 is to measure the naturalism of TnTs compared to patches in previous attacks. The results in Table 6.11 demonstrate our TnTs to have significantly higher naturalistic scores compared with the baselines.

In the second user study (Study 2), we randomly placed 3 of our generated TnTs together with 3 real flower images collected from Google Images (Google, 2020 (accessed July 3, 2020)) and asked participants to vote for the images that looks natural to them. The aim of this study is to measure the absolute naturalistic score of TnTs when compared with actual flower images. The results in Study 2 in Table 6.11 demonstrate that our TnTs, synthetic images, can often be comparable to real flower images. These results demonstrate our TnTs are more naturalistic and look significantly less malicious compared with prior works.

6.7 Generalisation to Adversarial Patch Attacks and Comparison with Prior Attacks

To expand the scope of the attack for cases where there is no human involvement in the decision loop—for fully autonomous systems—and where stealth in a physical deployment is not an objective, we consider removing the naturalism constraint on the attack method. Therefore, we take a further step to let the TnT Generator G_θ learn the adversarial features from the classifier; thus, demonstrating the generic nature of our attack method—*i.e.* once we remove the naturalism constraint, our attack can generate conventional adversarial patches.

Attack Methodology. The proposed attack is detailed in *Algorithm 6.2*. We call this alternation an Adversarial Patch Generator since the Generator, after updating, learns the mapping from the latent vector \mathbf{z} to generate adversarial patches. More specifically, instead of searching in the latent space to find the TnT as illustrated in Fig. 6.2, now, we allow the Generator G_θ to be updated and learnt from the gradient back-propagation from the loss $\ell(\mathbf{x}', y_{\text{target}}, y_{\text{source}})$ to become the Adversarial Patch Generator ($G_{\theta'}$). This allows the Generator to learn the adversarial features and generate multiple adversarial triggers with different mappings from \mathbf{z} .

Algorithm 6.2 Adversarial Patch Generator Process

```

1: Inputs: a batch of images  $\{\mathbf{x}^{(i)}\}_{i=1}^m$  with batch size  $m$ , source label  $\{y_{\text{source}}^{(i)}\}_{i=1}^m$ ,
   target label  $\{y_{\text{target}}^{(i)}\}_{i=1}^m$ , model  $p_M$ , latent vector  $\mathbf{z}$ , the hyper-parameter  $\lambda$  to
   balance the loss, natural generator  $G_\theta$ , and the thresholds to detect TnT  $\tau_{\text{batch}}, \tau_{\text{val}}$ 
   for batch and validation set respectively.

2: Initialization:

3: while  $ASR < \tau_{\text{val}}$  do
4:   Sample a batch of images  $\{\mathbf{x}^{(i)}\}_{i=1}^m$ 
5:   Sample a latent variable  $\mathbf{z} \sim p(\mathbf{z})$ 
6:    $\delta = G_\theta(\mathbf{z})$  ▷ a flower patch
7:   Generate the mask  $\mathbf{m}$  based on  $\delta$ 
8:    $\delta' \leftarrow \text{bgremoval}(\delta, \mathbf{m})$  ▷ Background removal
9:   for  $i = 1, \dots, m$  do
10:     $\mathbf{x}'^{(i)} = (1 - \mathbf{m}) \odot \mathbf{x}^{(i)} + \mathbf{m} \odot \delta'$ 
11:     $y_{\text{argmax}}^{(i)} = \arg \max_y p_M(y | \mathbf{x}'^{(i)})$ 
12:    if  $y_{\text{argmax}}^{(i)} = y_{\text{target}}^{(i)}$  then
13:       $fool = fool + 1$ 
14:     $L = \ell(\{\mathbf{x}'^{(i)}\}_{i=1}^m, \{y_{\text{target}}^{(i)}\}_{i=1}^m) - \lambda \ell(\{\mathbf{x}'^{(i)}\}_{i=1}^m, \{y_{\text{source}}^{(i)}\}_{i=1}^m)$ 
15:     $\theta \leftarrow \text{Adam}(\nabla_\theta L, \theta, \alpha, \beta_1, \beta_2)$ 
16:    if  $fool > \tau_{\text{batch}}$  then
17:      Sample a latent variable  $\mathbf{z} \sim p(\mathbf{z})$ 
18:       $\delta = G_\theta(\mathbf{z})$ 
19:      Test this  $\delta$  for the whole validation set  $\mathcal{X}_{\text{val}}$  to get  $ASR$ 
20:      if  $ASR \geq \tau_{\text{val}}$  then Complete update Generator, save the latest state as
      Adversarial Patch Generator  $G_\theta$ 

```

6.7.1 Comparing to LaVAN: Smallest (Noisy) Adversarial Patch

To show the effectiveness of our patch attack, we opted to compare with LaVAN since the study achieved the *smallest* state-of-the-art patch results. We evaluate our patches with the same 14 targeted labels reported in LaVAN (Karmon, Zoran and Goldberg, 2018) on 100 random images. As expected, the patches of only 2% of the input image size from our Adversarial Patch Generator easily achieve 100% ASR on 100 randomly sampled images from ImageNet. The results in Table 6.12 demonstrate that our patches

6.7.1 Comparing to LaVAN: Smallest (Noisy) Adversarial Patch

Table 6.12: Comparing ASR with LaVAN (Karmon, Zoran and Goldberg, 2018)—smallest state-of-the-art (noisy) adversarial patches of 2% of input image size—using the 14 targeted labels used in LaVAN on the *Inception-V3* network. Our patches achieve higher ASR in both settings; high and low confidence scores. Detailed examples of labels are shown in Figure 6.7.

	LaVAN	Ours
Average results across the 14 targeted labels from 100 ImageNet images		
ASR (conf \geq 0.9)	28.3%	72.9%
ASR (conf $<$ 0.9)	74.1%	98.1%

achieve a much higher ASR across the 14 targeted label on both settings of low and high confidence scores with a large margin of up to 44%.

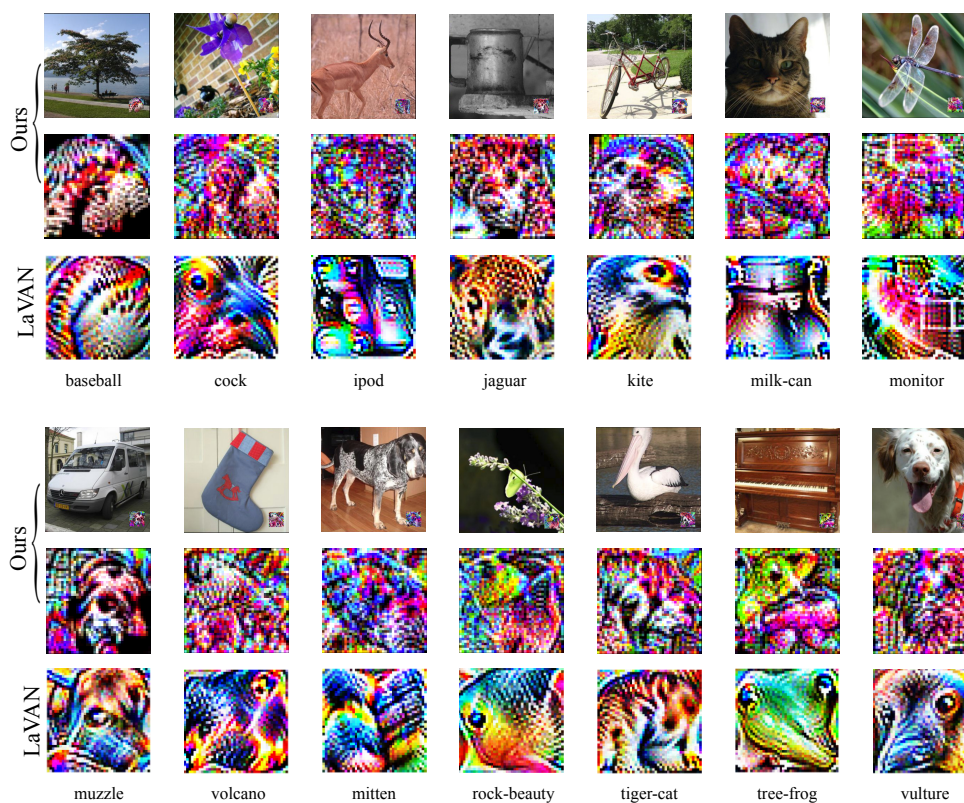


Figure 6.7: Generated patches from our Adversarial Patch Generator for the 14 different targeted labels in LaVAN. **The 1st row:** Our adversarial patches in the scene. **The 2nd row:** Our adversarial patches were rescaled to image size for display purposes. **The 3rd row:** adversarial patches from LaVAN for the same targeted label.

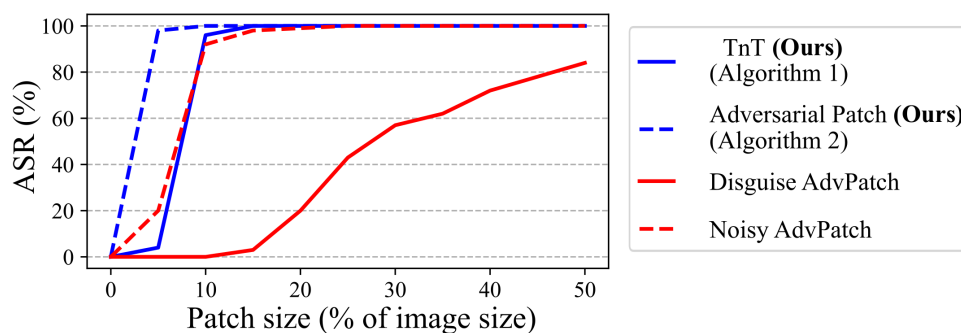


Figure 6.8: Investigating Attack Success Rate (ASR) and patch size. Our TnT is comparable with the Noisy AdvPatch (Brown et al., 2017), and significantly better than the Disguise AdvPatch (Brown et al., 2017). Our Adversarial Patch ASR outperforms both Noisy and Disguise counterparts.

6.7.2 Comparing to AdvPatch: A Method to Disguise the Appearance of a Target Class in a Patch

Here, we compare with Disguise AdvPatch and Noisy AdvPatch from (Brown et al., 2017) on the same VGG-16 network trained for the ImageNet task. We selected this study because: i) Disguise AdvPatch represents the efforts from (Brown et al., 2017) to hide the target class visible in the patch by disguising the noise patterns in another object while the Noisy AdvPatch is a noise-pattern patch visibly revealing the target class—toaster; ii) the method allows the generation of patches of different sizes. As shown in the Figure 6.8: i) efficacy of our TnT—the naturalistic patch—is comparable with the Noisy AdvPatch; and ii) significantly more effective than the Disguise AdvPatch aiming to hide the true target class revealed to a human observer in the Noisy AdvPatch.

Notably, our naturalistic TnT patch is highly effective when the patch size is larger than 10% of the input image. To maintain the high ASR with smaller patches, we need to sacrifice some of the naturalism (Algorithm 6.2). Then, we can observe nearly 100% ASR in Figure 6.8 with a patch size of nearly 5% of the input image; now, the ASR is significantly higher than the noisy counterpart, Noisy AdvPatch.

An attacker can trade-off naturalism to achieve significantly higher attack success rates compared with state-of-the-art adversarial patch attacks. Our results validates the generality of our attack method.

6.8 Physical World Deployments

An advantage of a TnT is the ability for an adversary to easily print and deploy the attack in a scene in the physical world to fool a deep perception system. In this experiment, we print our TnTs and deploy *targeted* physical attacks for the ImageNet and PubFig tasks.

Attack Settings. Following the practices in the physical adversarial attack work in (Kurakin, Goodfellow and Bengio, 2017), we saved our triggers (TnTs) and patches (from the Adversarial Patch Generator) as .PNG files and printed the triggers and patches using a printer with a resolution of 300 *dpi* to maintain the pixel quality. Examples of the triggers and patches are shown in Figure 6.9 and 6.10. We validate the effectiveness of the attack in various physical world settings with complex backgrounds, as displayed in Figure 6.9, by using a commercial camera of a smartphone to capture the scene—we used an iPhone 6S. We produced videos from our experiments to illustrate the effectiveness of the TnTs and patches in the physical world (<https://TnTattacks.github.io/>). In the videos, we also experiment with the robustness of our physical TnT in different *locations, scaling, lighting conditions, camera angles and positions, and so on.*

Results. Our results demonstrate that TnTs are robust to harsh physical-world conditions with *more than 90% of the images in frames successfully fooling the network and being recognised as the targeted class.* More detailed experiments of physical attacks are in videos accessed through the website.

We hypothesise the robustness of our TnT in the physical context is due to the fact that the patches are derived from a natural image distribution and are *universal* or *input-agnostic*. This allows the patch to potentially become invariant to various difficult conditions and suitable for deployment in physical world scenarios. Notably,



Figure 6.9: Various settings employed for the physical world attacks to impersonate ‘Barack Obama’. Results demonstrate our TnT is effective, even under different, complex, physical-world settings ranging from indoor to outdoor with different lighting conditions. The network recognises the person with the TnT to be Barack Obama with high confidence.

we only apply the simplest method of physical deployment—printing without any modification to offset the printer quality and loss of pixels as in (Sharif et al., 2016); hence, the ASR and naturalism can potentially be improved by applying more robust adversarial printing techniques (Sharif et al., 2016).

TnTs survive the harsh conditions in the physical world to pose a practical and realistic threat.

6.9 Attack Effectiveness Against Patch Defences

The rise in adversarial patch attacks has led to the emergence of defences— both empirical and provable methods (Xiang et al., 2021; Naseer, Khan and Porikli, 2019; Hayes, 2018; Chou, Tramèr and Pellegrino, 2020; Chiang et al., 2020; Levine and Feizi, 2020; Cohen, Rosenfeld and Kolter, 2019; Mirman, Gehr and Vechev, 2018; Wu, Tong and Vorobeychik, 2020; Rao, Stutz and Schiele, 2020). In this section, we evaluate the effectiveness of our TnTs against both certified and empirical defences against patch attacks.

6.9.1 Against Probably Robust Networks

A provable defence is the strongest defence and could potentially block and eliminate the adversarial patches completely. In this section, we evaluate the robustness of our attack against the multiple state-of-the-art (SOTA) provable defences including BagNet (Brendel and Bethge, 2019), Derandomized Smoothing (Levine and Feizi, 2020), and recently the improved versions of those defences named PatchGuard (Xiang et al., 2021) demonstrating superior performance compared with other provable methods (Chiang et al., 2020; Levine and Feizi, 2020; Cohen, Rosenfeld and Kolter, 2019; Mirman, Gehr and Vechev, 2018). Notably, the PatchGuard method relies on



Figure 6.10: Physical deployment of TnTs generated from the TnT Generator targeting different classes (shown on top in red) in the ImageNet task.

6.9.2 Against Empirically Robust Networks

the small receptive fields and robust masking to eliminate the adversarial effects of malicious patches and generate provable robustness for the defended system.

Experiment Setup. In this experiment, we use the same networks and configurations as in (Xiang et al., 2021) of BagNet (Brendel and Bethge, 2019) with receptive fields of 17×17 and de-randomized smoothed ResNet (DS) (Levine and Feizi, 2020) evaluated on ImageNet. Then, we also apply Robust Masking defence in (Xiang et al., 2021) to generate Mask-BN and Mask-DS provable robust networks, respectively. The provable Robustness results in Table 6.13 is evaluated on the entire validation set (50,000 images) of ImageNet using a mask size of 10% of the input image size.

Metrics. We use two metrics in this experiment: i) *Clean Acc*—the Accuracy obtained on benign inputs from the test set; and ii) *Provable Robustness*—the percentage of the images in the clean test set that are able to certified (*i.e.* no attack is expected to succeed). We report the results in Table 6.13.

In summary, after acknowledging the threat from TnTs—building the defence methods for TnT attacks—the defences achieved very low provable robustness, only up to 3.52% of inputs from 50,000 validation images can be certified in the best case (see Table 6.13). The reason is because, certifying against a larger patch, such our TnTs or AdvPatch (Brown et al., 2017), requires certifying that a correct prediction can be made for a specific input, potentially tainted by a larger adversarial patch, in the presence of the defence method. In PatchGuard (Brown et al., 2017), this requires operating under larger masked regions in the feature space. Consequently, the provable defence must make predictions from the aggregation of the remaining features (not masked); leading to lower performance as well as certified robustness. Hence, provable defences can only certify a smaller numbers of test inputs, *i.e.* achieve lower provable robustness, for larger adversarial patches. Therefore, an adversary can circumvent these defences with a larger patch, such as our TnTs or even AdvPatch in (Brown et al., 2017). Consequently, we observe TnTs attacks to still pose a realistic threat, even against *the strongest* defences.

6.9.2 Against Empirically Robust Networks

In this section, we evaluate our TnTs against empirically robust networks. As shown in (Zhang et al., 2020b; Carlini and Wagner, 2017a; Tramer et al., 2020), empirical defences such as (Chou, Tramèr and Pellegrino, 2020; Naseer, Khan and Porikli, 2019;

Hayes, 2018) are usually vulnerable to adaptive attackers once they are aware of the working mechanisms of the defences. Notably, recently, Wu, Tong and Vorobeychik (2020) highlighted that the conventional methods to improve the robustness against adversarial examples such as adversarial training and randomized smoothing showed limited effectiveness against physically realisable adversarial attacks, and proposed an approach named Defence against Occlusion Attacks (DOA) to defend against these physically realisable adversarial patch attacks. This approach is also developed in (Rao, Stutz and Schiele, 2020) but the authors jointly optimize patch values and location. These two state-of-the-art empirical defences (Wu, Tong and Vorobeychik, 2020; Rao, Stutz and Schiele, 2020) are the most relevant defence method against our attacks. Hence, we employ these defences in this section and assess the effectiveness of our TnTs against these robust defences.

Experiment Setup. We use the pre-trained robust Resnet110 networks provided by Wu, Tong and Vorobeychik (2020) on Github⁸ for the CIFAR-10 dataset. This network was trained on a patch size of 11×11 . Notably, as reported in the paper, even a network trained with a smaller patch size can defend against a patch size as large as 20% of the images (*i.e.* cover all of our attacks). For (Rao, Stutz and Schiele, 2020), we train the given robust ResNet-20 model from scratch following the default parameters of the strongest proposed defence (AT-FullLO)⁹ on CIFAR-10 dataset. To make a fair comparison with (Wu, Tong and Vorobeychik, 2020), we also train with the same mask size of 11×11 . To assess the effectiveness of our attack against these robust networks, we deploy our TnTs to achieve the challenging task of fooling the network to misclassify *any* images to the targeted label (car). First, we evaluate the robustness of the network against our TnTs for the same patch size used in training (11×11). However, to stretch out the robustness of the defended network, we also evaluate the network against a larger patch size of TnT around 20% since it was reported to be effective for sizes as large as 20% of the input (Wu, Tong and Vorobeychik, 2020). In addition, we employed the adversarial patches in (Wu, Tong and Vorobeychik, 2020; Rao, Stutz and Schiele, 2020) to compare with our TnT attacks as shown in Table 6.13.

Metrics. Similar to the Section 6.9.1, we report the *Clean Acc*—the Accuracy on benign inputs, and *Empirical Robustness*—the percentage of input images with patches that are correctly classified or the performance of the network under an attack.

⁸<https://github.com/tongwu2020/phattacks>

⁹<https://github.com/sukrutrao/Adversarial-Patch-Training>

6.10 Related Work

Table 6.13: Effectiveness of TnT attacks against robust defence methods (\uparrow *Robustness* is better for defences).

Networks	Clean Acc	Provable Robustness ¹
BagNet (Brendel and Bethge, 2019)	49.56%	0%
Mask-BagNet (Xiang et al., 2021)	49.65%	0.85%
DS (Levine and Feizi, 2020)	44.36%	3.52%
Mask-DS (Xiang et al., 2021)	39.77%	3.01%

	Empirical Robustness	
	Adversarial Patch ²	TnT
DOA (Wu, Tong and Vorobeychik, 2020)	86.5%	80.43%
AT-FullLO (Rao, Stutz and Schiele, 2020)	90.44%	72.2%

¹Provable Robustness is attack-agnostic.

²Patches employed are from the studies in (Wu, Tong and Vorobeychik, 2020; Rao, Stutz and Schiele, 2020) and provide a baseline for comparisons with TnTs.

In Table 6.13, we summarise the results for current state-of-the-art empirical defences against physically realisable patch attacks. We report on DOA in (Wu, Tong and Vorobeychik, 2020), and the defence focusing on location optimization adversarial patches (Rao, Stutz and Schiele, 2020). For the defended DOA network (Wu, Tong and Vorobeychik, 2020), when attacking the network with TnTs of the same patch size used in training (11×11), the robustness dropped from 80.43% to 13.78%. For the strongest defence, AT-FullLO network in (Rao, Stutz and Schiele, 2020), the robustness dropped from 72.2% to 11.02% under our TnT attack. In addition, because DOA was reported to be effective for sizes as large as 20% of the input (Wu, Tong and Vorobeychik, 2020), we increased the patch size of our TnTs to 20%. Although not reported in the table, the robustness of the network reduced significantly to 5.6%. These results demonstrate: i) the effectiveness of TnTs, even against state-of-the-art patch defences; and ii) that TnT attacks are an emerging new threat against DNNs.

6.10 Related Work

We describe prior work that focuses on *universal* perturbations and adversarial patch attacks and other *physically deployed* adversarial attacks; further, we also compare our attack method with other GAN-based adversarial attack methods.

UAP and Adversarial Patch Attacks. Moosavi-Dezfooli et al. (2017) showed the existence of a *universal* adversarial perturbation (UAP) in DNNs for image

classification tasks, which is a *unique noise tensor* that when added to any input fools the classifier to mount an untargeted attack. The authors have also shown that there are multiple UAPs in a DNN, which can be transferable to another network architecture of for the same task. In order to deploy a *universal* adversarial attack in a real-world setting, Brown et al. (2017) developed a spatially bounded Adversarial Patch (AdvPatch) to place in the scenes of ImageNet samples to fool the classifier to recognise objects as the toaster target class regardless of source inputs or locations. Karmon et al. extend this attack further to search for localized (or bounded) and visible adversarial noise in LaVAN (Karmon, Zoran and Goldberg, 2018) but the aim is to look for blind spots in a Deep Neural Network instead of physically realisable patches.

Other Physically Deployed Adversarial Attacks. Kurakin, Goodfellow and Bengio (2017) demonstrated that input-specific adversarial examples can also be deployed in the physical world in an untargeted attack if printed out and carefully cropped. Recently, Eykholt et al. (2018) showed that specially crafted perturbations constrained to sticker shapes can fool a Traffic Sign recognition task once stuck to a Stop sign; while Athalye et al. (2018) carefully crafted adversarial perturbations constrained to 3D objects to fool a DNN in the physical world. Different from ours, these adversarial examples are designed to work on a specific input (a specific Stop sign or 3D object), while our method is highly generalisable and the TnT can be printed out and attached to *any* input to work in the physical world (see Section 6.8). Concurrent studies Hu et al. (2021); Tan et al. (2021) have also attempted to realise naturalistic adversarial patch attacks, but against object detectors with the adversarial objective of causing a *misdetction of human objects* in a scene. To the best of our knowledge, our study remains the first to investigate universal, naturalistic, adversarial patches across a variety of classification tasks to misclassify *any* input to the attacker-desired *target label*. Further, TnTs are shown to be transferable and generalisable to tasks, models, and adversarial patch attacks.

GAN-based Adversarial Example Attacks. Researchers have investigated GAN-based structures to generate adversarial examples, such as (Baluja and Fischer, 2018; Xiao et al., 2018a; Jandial et al., 2019; Zhao, Dua and Singh, 2018; Carlini and Farid, 2020). Particularly, the authors in (Zhao, Dua and Singh, 2018) train a GAN and an additional Inverter network to generate *full-size, fake images* that are able to flip the predicted label or mount an *untargeted attack*. Notably, these studies resemble the investigation of an adversarial example objective—input-dependent or noisy perturbation-based

6.11 Discussion and Conclusion

distortions added to an input *covering* the whole image to mount an *untargeted* attack. Different from this line of work, we rely on a *pre-trained* generator, and search the latent space \mathbf{z} to discover a type of spatially bounded adversarial example; a *patch* that is *physically realisable and universal (input-agnostic)*. These attributes eases the process of deployment in the physical world to mount *targeted* attacks.

GAN-based Adversarial Patch Attacks. Sharif et al. (2019) apply a GAN-based method to generate spatially bounded physical adversarial sunglasses to impersonate a targeted person in a face recognition task (PubFig). The GAN-based method employs iterative training with feedback from the classifier under attack, which results in generating *noisy perturbations* constrained to the sunglasses mask. Notably, the sunglasses can be expected to occlude the main features of a face. In contrast, we do not alter the Generator for the TnT attacks (see Alg. 6.1) and are able to generate naturalistic patches by traversing through the latent space of the generator. Notably, the resulting patches can be successfully placed away from salient features of the input image.

PS-GAN (Liu et al., 2019a) proposes to utilise a GAN-based structure to find patches with naturalism. The major differences with our work are: *i)* the attack is *untargeted* compared to both *targeted* and *untargeted* attacks capable with ours; *ii)* PS-GAN is *input-dependent* hence, an attacker needs to mount the attack in different ways for different inputs, which is harder to deploy in real-world scenarios; we address this problem using a *universal* or *input-agnostic* patch where *any* input will be misclassified when a TnT is applied; *iii)* PS-GAN patch is placed in the main context of the image, which can occlude main features; *iv)* method applies image-to-image translation using an encoder-decoder generator to translate an *existing* natural patch to an adversarial counterpart while ours learns the natural image distribution and approach a naturalistic adversarial patch. Similar to universal adversarial patches LaVAN (Karmon, Zoran and Goldberg, 2018) and AdvPatch (Brown et al., 2017)), we seek to be location invariant and attack method can lead to less malicious-looking and more powerful (higher ASR) attacks.

6.11 Discussion and Conclusion

Are TnTs a formidable threat? We have validated through extensive experiments that natural-looking patches can successfully be used to fool Deep Neural Networks with

high attack success rates. We have shown that TnTs are effective against multiple state-of-the-art classifiers ranging from widely used *WideResNet50* in the Large-Scale Visual Recognition task of ImageNet dataset to VGG-face models in the face recognition task of *PubFig* dataset in both *targeted* and *untargeted* attacks. TnTs can possess: i) the naturalism achievable with with triggers used in Trojan attack methods; and ii) the generalisation and transferability of *adversarial examples* to other networks. This raises safety and security concerns regarding already deployed DNNs as well as future DNN deployments where attackers can use inconspicuous natural-looking object patches to misguide neural network systems without tampering with the model and risking discovery.

Are we limited to using flower triggers? In order to generate TnTs in our chapter, we need at least one distribution of natural objects. In our experiments, we utilised flowers as explained. However, natural objects are not limited to flowers and can be any object selected by an attacker. Importantly, it is both easy and low cost to obtain unlabelled datasets to generate TnTs as illustrated in our study (we used open-source, freely downloadable flowers to generate the flower dataset).

Are TnTs a threat for object detectors? Researchers (Lu et al., 2017) argue that attacking a classifier is different from attacking a detector, and some applications require an object detector, such as a road sign detector. The TnTs presented in this chapter focus on attacking classifiers; however, an extension to utilise the method with the objective of attacking object detectors is an interesting research direction. We will leave this as an interesting direction for future work.

What are potential avenues for mitigating the threat? We believe our work opens a new venue for further research into understanding the vulnerabilities of DNN systems. We believe our attack can be used for the “good” by providing a method for not only discovering vulnerabilities of DNN models but to generate sample inputs to improve the robustness and trustworthiness of DNN models.

The next chapter will briefly review the challenges explored in this dissertation to build robust deep neural networks and share the conclusions made upon the investigations. Moreover, the next chapter will outline worthwhile potential future research directions emerged from this dissertation.

Chapter 7

Conclusion

THIS chapter concludes the dissertation and suggests directions for future work.

7.1 Summary

This dissertation has presented novel approaches to developing robust deep neural networks by focusing on distinguishing and addressing two main threats; test-time Adversarial Examples and training-time Trojan attacks. To achieve robustness against Adversarial Examples, our formulation builds on Bayesian adversarial learning to address all of the identified challenges and build a robust Bayesian neural network in Computer Vision and malware domains. Our proposed approaches not only achieve state-of-the-art robustness in Bayesian adversarial learning in these domains, especially against strong unseen adversarial threats, but also provably bounds the difference between adversarial risk and conventional empirical risk. Meanwhile, to achieve robustness against Trojan attacks, we realise a novel unsupervised framework to sanitise Trojan attacks at run-time, achieve state-of-the-art performance, and establish a baseline for Trojan robustness at run-time. Additionally, by bridging the divide between Adversarial Patches and Trojan attacks in the input space, we expose the emerging threat of Universal Naturalistic Adversarial Patches (TnTs) capable of attacking a network at test time while exerting a high level of control that is similar to Trojan triggers without interfering with the training process. Our TnT approach is highly generalisable and can be effectively deployed in real-world scenarios. These investigations have been presented sequentially in Chapters 3, 4, 5 and 6.

In Chapter 3, to address the problem of Adversarial Examples, we prove that the projection of perturbed, yet valid, malware from the problem space into feature space will constitute a subset of feature-space adversarial attacks. Hence, a robust network against feature-space attacks is inherently robust against problem-space attacks. This is important because it alleviates the challenging problem of inverse feature mapping, which hinders the adoption of ML-based methods in the malware domain. The experimental results show that the BNN-based malware detectors that we built achieve state-of-the-art performance compared to existing ML-based methods, and the first Bayesian adversarial learning that we propose significantly improves the robustness of the malware classifier against strong unseen adversarial attacks.

Chapter 4 presents a Bayesian adversarial learning method that is robust against Adversarial Examples in the vision domain. The formulation is built on the Conjecture that a robust neural network quantifies the information gain from a benign observation in a manner equivalent to its adversarial counterpart. We prove that building a robust Bayesian neural network with this Conjecture of Information Gain further

tightens the bound of the difference between adversarial risk and conventional empirical risk, which Chapter 3 introduces. Our proposed approach represents a step towards a foundation for a *principled method of adversarially training BNNs* and achieves state-of-the-art robustness in the context of Bayesian adversarial learning methods.

In Chapter 5, we present the first input sanitisation method to purify the Trojan in the input space when it appears at test time via unsupervised learning of a Generative Adversarial Network. The proposed method has turned the strength of the input-agnostic Trojan attacks into a weakness. The method can be deployed at model deployment to effectively defeat Trojan attacks at run time, where denial of a service is not an option, as in the case of self-driving cars. The proposed method shows robustness against multiple advanced backdoor scenarios and is also effective against adaptive attacks targeting the defence method.

In Chapter 6, we bridge the divide between Adversarial Examples and Trojan attacks in the input space, where, for the first time, we propose a TnT that allows attackers to exert a greater level of control, including the ability to choose a location independent, natural-looking patch as a trigger in contrast to being constrained to noisy perturbations. Thus far, this ability has only been proven possible with Trojan attack methods that need to interfere with model building processes to embed the backdoor at the risk of discovery. However, our TnT can still realise a patch *deployable in the physical world*. Extensive empirical experiments on *large-scale visual classification tasks* ImageNet with evaluations across the *entire validation* set of 50,000 images have shown the realistic threat from TnTs and the robustness of such attacks. Our results also show the generalisability of the attack to different visual classification tasks, including CIFAR-10, GTSRB, PubFig, on multiple state-of-the-art deep neural networks, including *WideResnet50, Inception-V3, VGG-16*.

7.2 Future Work

Pursuing various possible avenues for further investigation could improve the robustness of deep neural networks:

- **How can we improve Bayesian adversarial learning?** Although we have achieved state-of-the-art robustness in the context of Bayesian adversarial learning, as shown in Table 3.4 in Chapter 3 and Table 4.2 in Chapter 4, there remains significant room for improvement, especially against strong unseen attack budgets. Given the finding

that a diversified Bayesian neural network improves robustness against Adversarial Examples, a promising research direction would be improving the diversity of Bayesian neural networks to achieve better robustness. Additionally, the current training process for realising a robust Bayesian neural network is time-consuming and computationally expensive, requiring weeks to realise a robust model. Future research should aim to address these concerns by, for example, reducing the time required to realise Adversarial Examples during training or employing more efficient posterior approximation to boost the learning process.

- **How can we apply the concept of input sanitisation developed in Chapter 5 in other domains?** The proposed Trojan defence method mainly focuses on the Computer Vision domain. However, Trojan and backdoor attacks have been developed in other domains, including text and audio domains (Liu et al., 2018b). Given most defence methods introduced in Section 5.8 of Chapter 5 would be insufficiently adaptive to function across domains, it is imperative to develop a robust defence method that is effective in multiple domains. Thus, the investigation of the adoption of the proposed concept of input sanitisation, introduced in Chapter 5, to realise robust defences against Trojan attacks in multiple domains represents a promising research direction.
- **How robust are the Trojan defence methods such as Februs in Chapter 5 in the physical world?** Recent work by Wenger et al. (2021) demonstrates both the effectiveness of backdoor attacks in the physical world and the lack of robustness of certain defence methods in this context. This is because physical triggers negate key assumptions made by defence methods based on triggers in the digital domain. Interestingly, although the defence method introduced in Chapter 5 mostly focuses on the digital domain, the approach is generic and does not make any specific assumptions about the domain. However, investigating the effectiveness of the proposed method and comparing it with other defence methods in the physical world represents a promising research direction.
- **How can we realise inconspicuousness and stealthiness features for Adversarial-Patch attacks in the physical world?** As Chapter 6 mentions, the TnTs generated while maintaining naturalism require the patch to be sufficiently large (at least 10% of the input image). The challenge is to reduce the size (to make it stealthier and more inconspicuous) of the trigger while maintaining naturalistic features of TnTs and is worthy of future research. It is pertinent to consider

blending our TnTs into the scene to make the attack even more inconspicuous and stealthy. However, this represents a challenging task because scenes captured in the physical world evolve dynamically. Using naturalistic Adversarial Patches, such as our TnTs, represents a first step towards approaching this problem; for example, natural-looking flower patches are more likely to appear in a scene and are less malicious in appearance compared to noisy perturbation-based patches. Thus, methods of blending universal naturalistic Adversarial Patches into a scene require further research.

- **How can we realise naturalistic adversarial patches against object detectors?** Chapter 6 demonstrated the existence of naturalistic adversarial patches (TnTs) in classifiers. However, attacking a classifier is different from attacking an object detector because of the complex task of object detectors—to identify bounding boxes and labels. Fooling object detectors to detect any object with naturalistic adversarial patches (such as our TnTs) as a designated target is a challenging task. But, it is of significant interest, because object detectors are widely used for critical tasks such as road sign detection in self-driving cars. Therefore, investigating method for generating targeted, universal, naturalistic Adversarial Patches against object detector is an interesting research direction and worthy of future research to improve our understanding of object detector vulnerabilities.
- **How can we realise a robust defence method against both Trojan and Adversarial Patch attacks?** Chapter 6 demonstrated the emerging threat of TnTs, where attackers bridge the divide between Trojan and Adversarial Patch attacks in the input space to generate an attack capable of exerting high levels of control similar to Trojan attacks without interfering with the neural networks. Because TnTs link Trojan and Adversarial Patch attacks, we believe that a defence method that is robust against our TnTs will inherently and simultaneously be robust against both Trojan and Adversarial Patch threats. Hence, we believe that our work opens a new avenue for investigating defence methods that are more robust against multiple threats, a subject worthy of future research.

Bibliography

- Al-Dujaili, A., Huang, A., Hemberg, E. and O'Reilly, U.M., 2018. Adversarial deep learning for robust detection of binary encoded malware. *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE. (Cited on page 51.)
- Amazon, 2022. Amazon Machine Learning. Available from: <https://aws.amazon.com/machine-learning>. (Cited on page 4.)
- Anderson, H.S. and Roth, P., 2018. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*. (Cited on pages 28, 30, 32, 33, 42, 43, and 46.)
- Anderson, R., Barton, C., Böhme, R., Clayton, R., Ganán, C., Grasso, T., Levi, M., Moore, T. and Vasek, M., 2019. Measuring the changing cost of cybercrime. (Cited on page 28.)
- Andriushchenko, M., Croce, F., Flammarion, N. and Hein, M., 2020. Square attack: a query-efficient black-box adversarial attack via random search. *European Conference on Computer Vision (ECCV)*. (Cited on page 68.)
- Anwar, S.M., Majid, M., Qayyum, A., Awais, M., Alnowami, M. and Khan, M.K., 2018. Medical image analysis using convolutional neural networks: a review. *Journal of Medical Systems*. (Cited on page 2.)
- ARO, n.d. BROAD AGENCY ANNOUNCEMENT FOR TrojAI [Online]. Available from: <https://www.arl.army.mil/www/pages/8/TrojAI-V3.2.pdf>. (Cited on page 77.)
- Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K. and Siemens, C., 2014. Drebin: Effective and explainable detection of android malware in your pocket. *Network and Distributed System Security Symposium (NDSS)*. (Cited on page 28.)
- Athalye, A., Carlini, N. and Wagner, D., 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *International Conference on Machine Learning (ICML)*. (Cited on pages 6, 22, 28, 35, 51, 64, and 68.)

BIBLIOGRAPHY

- Athalye, A., Engstrom, L., Ilyas, A. and Kwok, K., 2018. Synthesizing Robust Adversarial Examples. *International Conference on Machine Learning (ICML)*. (Cited on pages 36, 55, and 145.)
- Atsague, M., Fakorede, O. and Tian, J., 2021. A Mutual Information Regularization for Adversarial Training. *Asian Conference on Machine Learning (ACML)*. (Cited on page 71.)
- Backes, M. and Nauman, M., 2017. LUNA: Quantifying and Leveraging Uncertainty in Android Malware Analysis through Bayesian Machine Learning. *IEEE European Symposium on Security and Privacy (Euro S&P)*. (Cited on page 50.)
- Bagdasaryan, E. and Shmatikov, V., 2021. Blind Backdoors in Deep Learning Models. *USENIX Security*. (Cited on pages 7, 104, and 112.)
- Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. and Shmatikov, V., 2020. How to backdoor federated learning. *International Conference on Artificial Intelligence and Statistics (AISTATS)*. (Cited on pages 4 and 103.)
- Baluja, S. and Fischer, I., 2018. Learning to attack: Adversarial transformation networks. *Association for the Advancement of Artificial Intelligence (AAAI)*. (Cited on pages 130 and 145.)
- Bhattad, A., Chong, M.J., Liang, K., Li, B. and Forsyth, D.A., 2020. Unrestricted adversarial examples via semantic manipulation. *International Conference on Learning Representations (ICLR)*. (Cited on page 135.)
- Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G. and Roli, F., 2013. Evasion attacks against machine learning at test time. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. (Cited on pages 5, 28, and 32.)
- Biggio, B., Fumera, G. and Roli, F., 2013. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*. (Cited on pages 28 and 31.)
- Biggio, B., Fumera, G. and Roli, F., 2014. Pattern recognition systems under attack: Design issues and research challenges. *International Journal of Pattern Recognition and Artificial Intelligence*. (Cited on page 31.)

- Biggio, B. and Roli, F., 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*. (Cited on pages 28 and 32.)
- Blei, D.M., Kucukelbir, A. and McAuliffe, J.D., 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*. (Cited on pages 18 and 19.)
- Breiman, L., 1996. Bagging predictors. *Machine Learning*. (Cited on page 38.)
- Brendel, W. and Bethge, M., 2019. Approximating CNNs with bag-of-local-features models works surprisingly well on ImageNet. *International Conference on Learning Representations (ICLR)*. (Cited on pages 141, 142, and 144.)
- Brown, T.B., Mané, D., Roy, A., Abadi, M. and Gilmer, J., 2017. Adversarial Patch. *NIPS Workshop*. (Cited on pages xxvii, 3, 7, 109, 115, 117, 126, 130, 135, 139, 142, 145, and 146.)
- BVLC, 2022. Caffe Model Zoo. Available from: <https://github.com/BVLC/caffe/wiki/Model-Zoo>. (Cited on page 4.)
- Cao, Q., Shen, L., Xie, W., Parkhi, O.M. and Zisserman, A., 2018. VGGFace2: A dataset for recognising faces across pose and age. *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*. (Cited on pages 86 and 88.)
- Carbone, G., Wicker, M., Laurenti, L., Patane, A., Bortolussi, L. and Sanguinetti, G., 2020. Robustness of Bayesian Neural Networks to Gradient-Based Attacks. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on pages 30, 51, and 72.)
- Carlini, N. and Farid, H., 2020. Evading deepfake-image detectors with white-and black-box attacks. *CVPR Workshops*. (Cited on page 145.)
- Carlini, N., Jagielski, M. and Mironov, I., 2020. Cryptanalytic extraction of neural network models. *CRYPTO*. (Cited on pages 20, 31, and 116.)
- Carlini, N. and Wagner, D., 2017a. Adversarial examples are not easily detected: Bypassing ten detection methods. *ACM Workshop on Artificial Intelligence and Security*. (Cited on pages 5 and 142.)
- Carlini, N. and Wagner, D., 2017b. Towards evaluating the robustness of neural networks. *IEEE Security and Privacy (S&P)*. (Cited on pages 2, 3, 5, 21, 32, and 45.)

BIBLIOGRAPHY

- Carlini, N. and Wagner, D., 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *IEEE Security and Privacy Workshops (SPW)*. (Cited on page 3.)
- Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I. and Srivastava, B., 2019a. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *Artificial Intelligence Safety Workshop at Association for the Advancement of Artificial Intelligence (AAAI)*. (Cited on page 104.)
- Chen, C., Seff, A., Kornhauser, A. and Xiao, J., 2015. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. *IEEE International Conference on Computer Vision (ICCV)*. (Cited on page 2.)
- Chen, H., Fu, C., Zhao, J. and Koushanfar, F., 2019b. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. *International Joint Conference on Artificial Intelligence (IJCAI)*. (Cited on pages 77, 105, 106, and 107.)
- Chen, X., Liu, C., Li, B., Lu, K. and Song, D., 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv preprint arXiv:1712.05526*. (Cited on pages 3, 4, 76, 103, and 104.)
- Chiang, P.Y., Ni, R., Abdelkader, A., Zhu, C., Studer, C. and Goldstein, T., 2020. Certified defenses for adversarial patches. *International Conference on Learning Representations (ICLR)*. (Cited on page 141.)
- Chou, E., Tramèr, F. and Pellegrino, G., 2020. SentiNet: Detecting Physical Attacks Against Deep Learning Systems. *IEEE Security and Privacy Workshops (SPW)*. (Cited on pages 77, 81, 82, 101, 103, 104, 105, 141, and 142.)
- Coates, A., Ng, A. and Lee, H., 2011. An Analysis of Single Layer Networks in Unsupervised Feature Learning. *International Conference on Artificial Intelligence and Statistics (AISTATS)*. (Cited on page 64.)
- Cohen, J., Rosenfeld, E. and Kolter, Z., 2019. Certified adversarial robustness via randomized smoothing. *International Conference on Machine Learning (ICML)*. (Cited on page 141.)
- DEFCON, 2019. Machine learning static evasion competition. <https://www.elastic.co/blog/machine-learning-static-evasion-competition>. Accessed: 2022-08-09. (Cited on page 46.)

- Demetrio, L., Biggio, B., Lagorio, G., Roli, F. and Armando, A., 2021. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Transactions on Information Forensics and Security*. (Cited on page 28.)
- Eddy, M. and Perlroth, N., 2020. Cyber Attack Suspected in German Woman's Death [Online]. Available from: <https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomware-death.html>. (Cited on page 28.)
- Erdemir, E., Bickford, J., Melis, L. and Aydore, S., 2021. Adversarial robustness with non-uniform perturbations. *Advances in neural information processing systems*, 34, pp.19147–19159. (Cited on page 47.)
- Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T. and Song, D., 2018. Robust Physical-World Attacks on Deep Learning Models. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on page 145.)
- Feinman, R., Curtin, R.R., Shintre, S. and Gardner, A.B., 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*. (Cited on page 72.)
- Fleshman, W., 2019. Evading machine learning malware classifiers. <https://towardsdatascience.com/evading-machine-learning-malware-classifiers-ce52dabdb713>. Accessed: 2022-08-09. (Cited on pages 46 and 47.)
- Gal, Y., 2016. *Uncertainty in Deep Learning*. Ph.D. thesis. University of Cambridge. (Cited on page 49.)
- Gal, Y., Islam, R. and Ghahramani, Z., 2017. Deep Bayesian Active Learning with Image Data. *International Conference on Machine Learning (ICML)*. (Cited on page 54.)
- Gao, Y., Kim, Y., Doan, B.G., Zhang, Z., Zhang, G., Nepal, S., Ranasinghe, D.C. and Kim, H., 2021. Design and Evaluation of a Multi-Domain Trojan Detection Method on Deep Neural Networks. *IEEE Transactions on Dependable and Secure Computing*. (Cited on pages 104 and 105.)
- Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C. and Nepal, S., 2019. STRIP: A Defence against Trojan Attacks on Deep Neural Networks. *Annual Computer Security Applications Conference (ACSAC)*. (Cited on pages 77, 81, 82, 89, 104, 105, 106, and 107.)

BIBLIOGRAPHY

- Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep learning*. MIT press. (Cited on page 17.)
- Goodfellow, I., McDaniel, P. and Papernot, N., 2018. Making machine learning robust against adversarial inputs. *Communications of the ACM*. (Cited on pages 2, 5, and 20.)
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on pages 17, 84, and 116.)
- Goodfellow, I.J., Shlens, J. and Szegedy, C., 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*. (Cited on pages 2, 3, 21, 28, 45, 47, 51, and 115.)
- Google, 2020 (accessed July 3, 2020). *Google Images* [Online]. Available from: <https://images.google.com/>. (Cited on pages 118 and 136.)
- Grosse, K., Papernot, N., Manoharan, P., Backes, M. and McDaniel, P., 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*. (Cited on page 51.)
- Grosse, K., Papernot, N., Manoharan, P., Backes, M. and McDaniel, P., 2017. Adversarial examples for malware detection. *European Symposium on Research in Computer Security (ESORICS)*. Springer. (Cited on pages 28, 32, and 51.)
- Gu, T., Dolan-Gavitt, B. and Garg, S., 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. (Cited on pages 3, 4, 20, 108, and 118.)
- Gu, T., Liu, K., Dolan-Gavitt, B. and Garg, S., 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*. (Cited on pages xxiv, xxv, 4, 22, 76, 80, 88, 89, and 103.)
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A.C., 2017. Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on pages 18 and 85.)
- Guo, W., Wang, L., Xing, X., Du, M. and Song, D., 2019. TAVOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems. *arXiv preprint arXiv:1908.01763*. (Cited on pages 77, 79, 89, 95, 96, 97, 104, 105, and 107.)

- Harang, R. and Rudd, E.M., 2020. SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection. *arXiv preprint arXiv:2012.07634*. (Cited on pages 28, 30, 32, 36, 42, 43, and 50.)
- Hayes, J., 2018. On Visible Adversarial Perturbations & Digital Watermarking. *CVPR Workshops*. (Cited on pages 141 and 143.)
- He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *IEEE International Conference on Computer Vision (ICCV)*. (Cited on pages 116, 121, and 126.)
- He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 88 and 93.)
- Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J. and Song, D., 2021. Natural Adversarial Examples. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on page 116.)
- Hong, S., Carlini, N. and Kurakin, A., 2021. Handcrafted backdoors in deep neural networks. *arXiv preprint arXiv:2106.04690*. (Cited on page 4.)
- Houlsby, N., Huszár, F., Ghahramani, Z. and Lengyel, M., 2011. Bayesian Active Learning for Classification and Preference Learning. *arXiv preprint arXiv:1112.5745*. (Cited on page 54.)
- Hu, W. and Tan, Y., 2017. Generating adversarial malware examples for black-box attacks based on GAN. *arXiv preprint arXiv:1702.05983*. (Cited on page 51.)
- Hu, Y.C.T., Kung, B.H., Tan, D.S., Chen, J.C., Hua, K.L. and Cheng, W.H., 2021. Naturalistic Physical Adversarial Patch for Object Detectors. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 135 and 145.)
- Iizuka, S., Simo-Serra, E. and Ishikawa, H., 2017. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics*. (Cited on page 84.)
- Izmailov, P., Vikram, S., Hoffman, M.D. and Wilson, A.G., 2021. What Are Bayesian Neural Network Posteriors Really Like? *International Conference on Machine Learning (ICML)*. (Cited on pages 19, 35, 37, and 73.)

BIBLIOGRAPHY

Jacobs, R.A., Jordan, M.I., Nowlan, S.J. and Hinton, G.E., 1991. Adaptive mixtures of local experts. *Neural Computation*. (Cited on page 38.)

Jandial, S., Mangla, P., Varshney, S. and Balasubramanian, V., 2019. AdvGAN++: Harnessing latent layers for adversary generation. *ICCV Workshops*. (Cited on page 145.)

Karmon, D., Zoran, D. and Goldberg, Y., 2018. LaVAN: Localized and Visible Adversarial Noise. *International Conference on Machine Learning (ICML)*. (Cited on pages xxxi, 3, 115, 117, 119, 126, 127, 128, 130, 135, 137, 138, 145, and 146.)

KasperskyLab, 2020. The number of new malicious files detected every day increases by 5.2% to 360,000 in 2020. https://www.kaspersky.com/about/press-releases/2020_the-number-of-new-malicious-files-detected-every-day-increases-by-52-to-360000-in-2020
Accessed: 2022-04-01. (Cited on pages 28 and 45.)

Koh, J.Y., 2022. Model Zoo [Online]. Available from: <https://modelzoo.co/>. (Cited on page 4.)

Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C. and Roli, F., 2018. Adversarial malware binaries: Evading deep learning for malware detection in executables. *European Signal Processing Conference (EUSIPCO)*. (Cited on pages 28, 48, and 51.)

Krčál, M., Švec, O., Bálek, M. and Jašek, O., 2018. Deep convolutional malware classifiers can learn from raw executables and labels only. *Iclr workshop*. (Cited on page 50.)

Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B. and Keshet, J., 2018. Deceiving end-to-end deep learning malware detectors using adversarial examples. *arXiv preprint arXiv:1802.04528*. (Cited on pages 28, 48, and 51.)

Krizhevsky, A., Hinton, G. et al., 2009. Learning multiple layers of features from tiny images. (Cited on pages 64, 86, 87, 88, and 124.)

Kurakin, A., Goodfellow, I. and Bengio, S., 2017. Adversarial examples in the physical world. *ICLR Workshop*. (Cited on pages 140 and 145.)

- Kurakin, A., Goodfellow, I., Bengio, S., Dong, Y., Liao, F., Liang, M., Pang, T., Zhu, J., Hu, X., Xie, C. et al., 2018. Adversarial attacks and defences competition. *The NIPS'17 Competition: Building Intelligent Systems*. (Cited on pages 6 and 21.)
- Lakshminarayanan, B., Pritzel, A. and Blundell, C., 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems (NIPS)*. (Cited on page 66.)
- LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *Nature*. (Cited on page 17.)
- LeCun, Y., Cortes, C. and Burges, C., 2010. MNIST handwritten digit database. *ATT Labs*. (Cited on page 104.)
- Lee, G.H., Yuan, Y., Chang, S. and Jaakkola, T., 2019. Tight certificates of adversarial robustness for randomly smoothed classifiers. *Advances in neural information processing systems (nips)*. (Cited on page 29.)
- Lee, H.S. and Kim, K., 2018. Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network. *IEEE Transactions on Intelligent Transportation Systems*. (Cited on page 106.)
- Levine, A. and Feizi, S., 2020. (De)Randomized Smoothing for Certifiable Defense against Patch Attacks. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on pages 141, 142, and 144.)
- Li, S., Xue, M., Zhao, B., Zhu, H. and Zhang, X., 2020. Invisible Backdoor Attacks on Deep Neural Networks via Steganography and Regularization. *IEEE Transactions on Dependable and Secure Computing*. (Cited on page 103.)
- Liang, B., Li, H., Su, M., Bian, P., Li, X. and Shi, W., 2018. Deep text classification can be fooled. *International Joint Conference on Artificial Intelligence (IJCAI)*. (Cited on page 3.)
- Lin, S.C., Zhang, Y., Hsu, C.H., Skach, M., Haque, M.E., Tang, L. and Mars, J., 2018. The architectural implications of autonomous driving: Constraints and acceleration. *ACM Special Interest Group on Programming Languages (SIGPLAN) Notices*. (Cited on page 105.)
- Liu, A., Liu, X., Fan, J., Ma, Y., Zhang, A., Xie, H. and Tao, D., 2019a. Perceptual-sensitive gan for generating adversarial patches. *Association for the Advancement of Artificial Intelligence (AAAI)*. (Cited on pages 117, 118, 126, 127, and 146.)

BIBLIOGRAPHY

- Liu, K., Dolan-Gavitt, B. and Garg, S., 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. (Cited on pages 77, 89, 104, and 106.)
- Liu, Q. and Wang, D., 2016. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. *Advances in Neural Information Processing Systems (NIPS)*. (Cited on pages 19, 30, 37, 51, 54, 55, 58, and 73.)
- Liu, X., Cheng, M., Zhang, H. and Hsieh, C.J., 2018a. Towards robust neural networks via random self-ensemble. *Proceedings of the European Conference on Computer Vision (ECCV)*. (Cited on page 73.)
- Liu, X., Li, Y., Chongruo, W. and Cho-Jui, H., 2019b. ADV-BNN: Improved Adversarial Defense Through Robust Bayesian Neural Network. *International Conference on Learning Representations (ICLR)*. (Cited on pages iv, xxiv, 30, 37, 48, 49, 51, 53, 55, 64, 65, 66, 67, and 73.)
- Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y. and Zhang, X., 2019c. ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation. *ACM conference on Computer and Communications Security (CCS)*. (Cited on pages 82, 104, and 106.)
- Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W. and Zhang, X., 2018b. Trojaning Attack on Neural Networks. *Network and Distributed System Security Symposium (NDSS)*. (Cited on pages 3, 4, 22, 76, 103, 104, and 152.)
- Liu, Y., Ma, X., Bailey, J. and Lu, F., 2020. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. *European Conference on Computer Vision (ECCV)*. (Cited on page 104.)
- Liu, Y., Xie, Y. and Srivastava, A., 2017. Neural trojans. *2017 IEEE International Conference on Computer Design (ICCD)*. (Cited on page 104.)
- Lu, J., Sibai, H., Fabry, E. and Forsyth, D., 2017. Standard detectors aren't (currently) fooled by physical adversarial stop signs. *arxiv preprint arxiv:1710.03337*. (Cited on page 147.)
- Maaten, L. Van der and Hinton, G., 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*. (Cited on pages xxiv and 81.)
- MacKay, D.J., 1992. A practical bayesian framework for backpropagation networks. *Neural computation*. (Cited on page 35.)

- Madry, A., Makelov, A., Schmidt, L., Tsipras, D. and Vladu, A., 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. *International Conference on Learning Representations (ICLR)*. (Cited on pages 2, 3, 5, 6, 21, 28, 35, 36, 45, 48, 51, 53, 64, 65, 66, 70, and 100.)
- Mantovani, A., Aonzo, S., Ugarte-Pedrero, X., Merlo, A. and Balzarotti, D., 2020. Prevalence and impact of low-entropy packing schemes in the malware ecosystem. *Ndss*. (Cited on page 46.)
- Mathias, M., Timofte, R., Benenson, R. and Van Gool, L., 2013. Traffic sign recognition — How far are we from the solution? *International Joint Conference on Neural Networks (IJCNN)*. (Cited on pages 86 and 88.)
- Mirman, M., Gehr, T. and Vechev, M.T., 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. *International Conference on Machine Learning (ICML)*. (Cited on page 141.)
- Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O. and Frossard, P., 2017. Universal adversarial perturbations. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 2, 130, and 144.)
- Naseer, M., Khan, S. and Porikli, F., 2019. Local Gradients Smoothing: Defense Against Localized Adversarial Attacks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*. (Cited on pages 141 and 142.)
- Neyshabur, B., Bhojanapalli, S., Mcallester, D. and Srebro, N., 2017. Exploring Generalization in Deep Learning. *Advances in Neural Information Processing Systems (NIPS)*. (Cited on page 59.)
- Nguyen, A.T., Raff, E., Nicholas, C. and Holt, J., 2021. Leveraging Uncertainty for Improved Static Malware Detection Under Extreme False Positive Constraints. *IJCAI Workshop on Adaptive Cyber Defense*. (Cited on page 50.)
- OpenCV, 2020 (accessed June 25, 2020). *Image Thresholding*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html. (Cited on page 119.)
- Pal, A. and Das, A., 2019. TorchGAN: A Flexible Framework for GAN Training and Evaluation. *arXiv preprint arXiv:1606.04435*. 1909.03410. (Cited on page 122.)

BIBLIOGRAPHY

- Pang, R., Shen, H., Zhang, X., Ji, S., Vorobeychik, Y., Luo, X., Liu, A. and Wang, T., 2020. A tale of evil twins: Adversarial inputs versus poisoned models. *ACM conference on Computer and Communications Security (CCS)*. (Cited on page 7.)
- Papernot, N., McDaniel, P. and Goodfellow, I., 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*. (Cited on pages 5 and 115.)
- Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B. and Swami, A., 2017. Practical black-box attacks against machine learning. *ACM ASIA Conference on Computer and Communications Security (AsiaCCS)*. (Cited on page 115.)
- Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B. and Swami, A., 2016. The limitations of deep learning in adversarial settings. *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. (Cited on pages 6, 21, and 56.)
- Papernot, N., McDaniel, P., Sinha, A. and Wellman, M.P., 2018. SoK: Security and Privacy in Machine Learning. *IEEE European Symposium on Security and Privacy (EuroS&P)*. (Cited on page 20.)
- Parkhi, O.M., Vedaldi, A. and Zisserman, A., 2015. Deep Face Recognition. *British Machine Vision Conference (BMVC)*. (Cited on pages 88 and 124.)
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on pages xxx, 123, and 129.)
- Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C. and Molloy, I., 2012. Using probabilistic generative models for ranking risks of android apps. *Proceedings of the 2012 ACM conference on Computer and communications security*. (Cited on page 28.)
- Pierazzi, F., Pendlebury, F., Cortellazzi, J. and Cavallaro, L., 2020. Intriguing properties of adversarial ml attacks in the problem space. *IEEE symposium on security and privacy (S&P)*. IEEE. (Cited on pages 3, 28, 29, 32, 33, and 51.)
- Pinto, N., Stone, Z., Zickler, T. and Cox, D., 2011. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. *CVPR Workshops*. (Cited on page 124.)

- Quiring, E., Maier, A. and Rieck, K., 2019. Misleading authorship attribution of source code using adversarial learning. *USENIX Security*. (Cited on pages 28 and 32.)
- Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B. and Nicholas, C.K., 2018. Malware detection by eating a whole exe. *AAAI Workshops*. (Cited on pages 28, 42, 44, 46, and 50.)
- Raff, E., Fleshman, W., Zak, R., Anderson, H.S., Filar, B. and McLean, M., 2021. Classifying sequences of extreme length with constant memory applied to malware detection. *Association for the Advancement of Artificial Intelligence (AAAI)*. (Cited on page 44.)
- Rao, S., Stutz, D. and Schiele, B., 2020. Adversarial Training Against Location-Optimized Adversarial Patches. *ECCV Workshops*. (Cited on pages 119, 141, 143, and 144.)
- Ritter, H., Botev, A. and Barber, D., 2018. A scalable laplace approximation for neural networks. *International conference on learning representations ICLR*. (Cited on page 35.)
- Rolnick, D. and Kording, K., 2020. Reverse-engineering deep ReLU networks. *International Conference on Machine Learning (ICML)*. (Cited on pages 20, 31, and 116.)
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al., 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*. (Cited on page 123.)
- Saha, A., Subramanya, A. and Pirsiavash, H., 2020. Hidden Trigger Backdoor Attacks. *Association for the Advancement of Artificial Intelligence (AAAI)*. (Cited on page 104.)
- Samek, W., Binder, A., Montavon, G., Lapuschkin, S. and Müller, K., 2017. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Transactions on Neural Networks and Learning Systems*. (Cited on page 5.)
- Saxe, J. and Berlin, K., 2015. Deep neural network based malware detection using two dimensional binary program features. *International Conference on Malicious and Unwanted Software (MALWARE)*. (Cited on page 50.)
- Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K. and Madry, A., 2018. Adversarially Robust Generalization Requires More Data. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on page 64.)

BIBLIOGRAPHY

- Schultz, M.G., Eskin, E., Zadok, F. and Stolfo, S.J., 2000. Data mining methods for detection of new malicious executables. *IEEE Symposium on Security and Privacy (S&P)*. IEEE. (Cited on pages 31 and 50.)
- Sebastián, M., Rivera, R., Kotzias, P. and Caballero, J., 2016. Avclass: A tool for massive malware labeling. *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. (Cited on page 42.)
- Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. and Batra, D., 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *IEEE International Conference on Computer Vision (ICCV)*. (Cited on pages xxv, 82, 83, and 105.)
- Severi, G., Meyer, J., Coull, S. and Oprea, A., 2021. {Explanation-Guided} Backdoor Poisoning Attacks Against Malware Classifiers. *USENIX Security*. (Cited on pages 3 and 4.)
- Sharif, M., Bhagavatula, S., Bauer, L. and Reiter, M.K., 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. *ACM conference on Computer and Communications Security (CCS)*. (Cited on page 141.)
- Sharif, M., Bhagavatula, S., Bauer, L. and Reiter, M.K., 2019. A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security*. (Cited on page 146.)
- Simonyan, K. and Zisserman, A., 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*. (Cited on pages 87 and 124.)
- Smith, L. and Gal, Y., 2018. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*. (Cited on page 72.)
- Song, Y., Kim, T., Nowozin, S., Ermon, S. and Kushman, N., 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*. (Cited on pages 6 and 21.)
- Stallkamp, J., Schlipsing, M., Salmen, J. and Igel, C., 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*. (Cited on pages 86, 87, 88, and 124.)

- Suciu, O., Coull, S.E. and Johns, J., 2019. Exploring adversarial examples in malware detection. *IEEE Security and Privacy Workshops (SPW)*. (Cited on pages 28, 32, 48, and 51.)
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R., 2014. Intriguing properties of neural networks. *International Conference on Learning Representations (ICLR)*. (Cited on pages 2, 20, 21, 28, 32, 82, and 100.)
- Taigman, Y., Yang, M., Ranzato, M. and Wolf, L., 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on page 2.)
- Tan, J., Ji, N., Xie, H. and Xiang, X., 2021. Legitimate Adversarial Patches: Evading Human Eyes and Detection Models in the Physical World. *Proceedings of the 29th ACM International Conference on Multimedia*. (Cited on page 145.)
- Tramer, F., Carlini, N., Brendel, W. and Madry, A., 2020. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*. (Cited on page 142.)
- Tramèr, F., Zhang, F., Juels, A., Reiter, M.K. and Ristenpart, T., 2016. Stealing machine learning models via prediction apis. *USENIX Security*. (Cited on pages 20, 31, and 116.)
- Vo, V.Q., Abbasnejad, E. and Ranasinghe, D.C., 2022a. Query efficient decision based sparse attacks against black-box deep learning models. *International conference on learning representations (iclr)*. (Cited on page 20.)
- Vo, V.Q., Abbasnejad, E. and Ranasinghe, D.C., 2022b. RamBoAttack: A Robust Query Efficient Deep Neural Network Decision Exploit. *Network and Distributed Systems Security (NDSS) Symposium*. (Cited on page 20.)
- Wang, B., Cao, X., jia, J. and Gong, N.Z., 2020. On Certifying Robustness against Backdoor Attacks via Randomized Smoothing. (Cited on page 104.)
- Wang, B., Jia, J., Cao, X. and Gong, N.Z., 2021. Certified robustness of graph neural networks against adversarial structural perturbation. *Acm sigkdd conference on knowledge discovery & data mining*. (Cited on page 29.)
- Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H. and Zhao, B.Y., 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks.

BIBLIOGRAPHY

- IEEE Symposium on Security and Privacy (S&P)*. (Cited on pages 7, 77, 79, 81, 82, 87, 88, 89, 95, 96, 97, 104, 105, 106, 107, 108, and 124.)
- Wang, D. and Liu, Q., 2019. Nonlinear stein variational gradient descent for learning diversified mixture models. *International Conference on Machine Learning (ICML)*. (Cited on page 19.)
- Weber, M., Xu, X., Karlas, B., Zhang, C. and Li, B., 2020. RAB: Provable Robustness Against Backdoor Attacks. (Cited on page 104.)
- Welling, M. and Teh, Y.W., 2011. Bayesian Learning via Stochastic Gradient Langevin Dynamics. *International Conference on Machine Learning (ICML)*. (Cited on pages 6 and 19.)
- Wenger, E., Passananti, J., Bhagoji, A.N., Yao, Y., Zheng, H. and Zhao, B.Y., 2021. Backdoor attacks against deep learning systems in the physical world. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 108 and 152.)
- Wicker, M., Laurenti, L., Patane, A., Chen, Z., Zhang, Z. and Kwiatkowska, M., 2021. Bayesian Inference with Certifiable Adversarial Robustness. *International Conference on Artificial Intelligence and Statistics (AISTATS)*. (Cited on pages 30, 51, and 72.)
- Wierzynski, C., 2018. The Challenges and Opportunities of Explainable AI. Available from: <https://www.intel.ai/the-challenges-and-opportunities-of-explainable-ai>. (Cited on page 5.)
- Wolpert, D.H., 1992. Stacked generalization. *Neural Networks*. (Cited on page 38.)
- Wu, T., Tong, L. and Vorobeychik, Y., 2020. Defending against physically realizable attacks on image classification. *International Conference on Learning Representations (ICLR)*. (Cited on pages 141, 143, and 144.)
- Xiang, C., Bhagoji, A.N., Sehwal, V. and Mittal, P., 2021. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking. *USENIX Security*. (Cited on pages 141, 142, and 144.)
- Xiao, C., Li, B., Zhu, J. yan, He, W., Liu, M. and Song, D., 2018a. Generating Adversarial Examples with Adversarial Networks. *International Joint Conference on Artificial Intelligence (IJCAI)*. (Cited on page 145.)

- Xiao, C., Zhu, J.Y., Li, B., He, W., Liu, M. and Song, D., 2018b. Spatially transformed adversarial examples. *International Conference on Learning Representations (ICLR)*. (Cited on page 135.)
- Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L. and He, K., 2019. Feature denoising for improving adversarial robustness. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 6 and 21.)
- Xu, W., Qi, Y. and Evans, D., 2016. Automatically evading classifiers. *Network and Distributed System Security Symposium (NDSS)*. (Cited on pages 32 and 51.)
- Ye, N. and Zhu, Z., 2018. Bayesian adversarial learning. *Advances in Neural Information Processing Systems (NeurIPS)*. (Cited on pages 6, 30, 51, and 73.)
- Zhang, R., Isola, P. and Efros, A.A., 2016. Colorful image colorization. *European Conference on Computer Vision (ECCV)*. (Cited on page 135.)
- Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X. and Wang, T., 2020a. Interpretable Deep Learning under Fire. *USENIX Security*. (Cited on page 100.)
- Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X. and Wang, T., 2020b. Interpretable deep learning under fire. *USENIX Security*. (Cited on page 142.)
- Zhang, Z., Jia, J., Wang, B. and Gong, N.Z., 2021. Backdoor Attacks to Graph Neural Networks. *ACM Symposium on Access Control Models and Technologies (SACMAT)*. (Cited on page 104.)
- Zhao, Z., Dua, D. and Singh, S., 2018. Generating Natural Adversarial Examples. *International Conference on Learning Representations (ICLR)*. (Cited on page 145.)
- Zhu, S., Zhang, X. and Evans, D., 2020. Learning adversarially robust representations via worst-case mutual information maximization. *International Conference on Machine Learning (ICML)*. (Cited on page 71.)
- Zimmermann, R.S., 2019. Comment on "Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network". *arXiv preprint arXiv:1907.00895*. (Cited on pages 36 and 55.)

Biography

Bao Gia Doan received his M.Sc. degree from RMIT University in 2014. From 2014 to 2018, he worked as a senior process engineer at Intel. In 2018, he was awarded a University of Adelaide International Scholarship to pursue his doctoral degree at the School of Computer Science, the University of Adelaide under the supervision of Associate Professor Damith Chinthana Ranasinghe and Doctor Ehsan Abbasnejad. His research interests include Artificial Intelligence Security and Trustworthy Machine Learning.

Bao Gia Doan
giabao.doan@adelaide.edu.au