Assessment of the Robustness of Deep Neural Networks (DNNs)

Ronghui Mu, BEng (Hons), MSc
School of Computing and Communications
Lancaster University

A thesis submitted for the degree of
Doctor of Philosophy

October, 2023

“My life has a limit, but knowledge has none.”
–Laozi

To my family..

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words including appendices and footnotes, but excluding the bibliography.
Ronghui Mu

Assessment of the Robustness of Deep Neural Networks (DNNs)

Ronghui Mu, BEng (Hons), MSc.
School of Computing and Communications, Lancaster University

A thesis submitted for the degree of *Doctor of Philosophy*. October, 2023

Abstract

In the past decade, Deep Neural Networks (DNNs) have demonstrated outstanding performance in various domains. However, recently, some researchers have shown that DNNs are surprisingly vulnerable to adversarial attacks. For instance, adding a small, human-imperceptible perturbation to an input image can fool DNNs, enabling the model to make an arbitrarily wrong prediction with high confidence. This raises serious concerns about the readiness of deep learning models, particularly in safety-critical applications, such as surveillance systems, autonomous vehicles, and medical applications. Hence, it is vital to investigate the performance of DNNs in an adversarial environment.

In this thesis, we study the robustness of DNNs in three aspects: adversarial attacks, adversarial defence, and robustness verification. First, we address the robustness problems on video models and propose DeepSAVA, a sparse adversarial attack on video models. It aims to add human-imperceptible perturbations on the crucial frame of the input video to fool classifiers. Additionally, we construct a novel adversarial training framework based on the perturbations generated by DeepSAVA to increase the robustness of video classification models. The results show that DeepSAVA runs a relatively sparse attack on video models, yet achieves state-of-the-art performance in terms of attack success rate and adversarial transferability.

Next, we address the challenges of robustness verification in two deep learning models: 3D point cloud models and cooperative multi-agent reinforcement learning models (c-MARLs). Robustness verification aims to provide solid proof of robustness within an input space to any adversarial attacks. To verify the robustness of 3D point cloud models, we propose an efficient verification framework, 3DVerifier, which tackles the challenges of cross-non-linearity operations in multiplication layers and the high computational complexity of high-dimensional point cloud inputs. We use a linear relaxation function to bound the multiplication layer and combine forward and backward propagation to compute the certified bounds of the outputs of the point cloud models.

For certifying the c-MARLs, we propose a novel certification method, which is the first work to leverage a scalable approach for c-MARLs to determine actions with guaranteed certified bounds. The challenges of c-MARL certification are accumulated uncertainty as the number of agents increases and the potential lack of impact when changing the action of a single agent into a global team reward. These challenges prevent me from using existing algorithms directly. We employ the false discovery rate (FDR) controlling procedure, considering the importance of each agent to certify per-state robustness and

propose a tree-search-based algorithm to find a lower bound of the global reward under the minimal certified perturbation. The experimental results show that the obtained certification bounds are much tighter than those of state-of-the-art RL certification solutions.

In summary, this thesis focuses on assessing the robustness of deep learning models that are widely applied in safety-critical systems but rarely studied by the community. This thesis not only investigates the motivation and challenges of assessing the robustness of these deep learning models but also proposes novel and effective approaches to tackle these challenges.

List of Publications

Publications

Included Publications

1. Sparse Adversarial Video Attacks with Spatial Transformations [97]
   Ronghui Mu,Wenjie Ruan, Leandro Soriano Marcolino, Qiang Ni
   BMVC, 2021

2. 3DVerifier: efficient robustness verification for 3D point cloud models [95]
   Ronghui Mu, Wenjie Ruan, Leandro Soriano Marcolino, Qiang Ni
   Machine Learning, 2022

3. Certified Policy Smoothing for Cooperative Multi-Agent Reinforcement Learning [96]
   Ronghui Mu, Wenjie Ruan, Leandro Soriano Marcolino, Gaojie Jin, Qiang Ni
   AAAI,2023

Published but not included in this thesis

1. Randomized Adversarial Training via Taylor Expansion [55]
   Gaojie Jin, Xinping Yi, Dengyu Wu, **Ronghui Mu**, Xiaowei Huang
   CVPR,2023

Preprint(Under Review)

1. Sparse Adversarial Video Attacks and Defences
   Ronghui Mu,Wenjie Ruan, Leandro Soriano Marcolino, Qiang Ni
   Neural Networks, 2023

2. SAT: Symmetric Adversarial Training for Inherent Label Noise
   Zhen Chen, Fu Wang, **Ronghui Mu**, PeiPei Xu, Wenjie Ruan, Xiaowei Huang
   Machine Learning, 2023

Acknowledgements

Upon finishing this thesis, I am reflecting on my three and a half-year journey at the University of Lancaster. Throughout my Ph.D., I spent one and a half years at the University of Exeter, both of which hold a special place in my heart. These experiences have left an enduring impression on me, fostering substantial personal and professional growth while equipping me with invaluable academic knowledge and skills. I extend my heartfelt gratitude to the numerous individuals whose unwavering support and assistance have been instrumental in turning this journey into a reality.

I would like to extend my heartfelt appreciation to my supervisors, Dr. Wenjie Ruan and Dr. Leandro Marcolino, for their invaluable assistance, meticulous feedback, and constructive guidance throughout these years. Their unwavering support played a crucial role in enabling me to successfully complete my Ph.D.

Additionally, I would like to express my gratitude to my collaborators, Prof. Qiang Ni, Prof. Xiaowei Huang, Dr. Xingping Yi, Dengyu Wu, Zhen Chen, Fu Wang, and Peipei Xu, for their contributions in helping me accomplish my research projects. Also, I would like to thank Peng Gao, Matheus Alves, Abdulrahman Kerim and Washington L. S. Ramos for thorough reviews and proofreadings. Their collaboration and assistance have been instrumental in my academic journey.

I express my gratitude to my friends Chi Zhang and Siqi Sun, whose companionship has brought laughter into my life and helped me overcome the challenges during my Ph.D. journey. I have also cherished the moments of relaxation shared with them outside of work. Additionally, I would like to express my gratitude to my colleagues Tianle Zhang, Yanghao Zhang, Shuohan Liu, Zheng Wang and Xiangyu Yin. Collaborating with them has offered invaluable opportunities to exchange ideas and perspectives, enriching the quality of both my research and personal life.

I would like to express my heartfelt appreciation to the School of Computing and Communications at Lancaster University for providing funding support for my research throughout my Ph.D. journey. Their financial assistance has played a vital role in the successful completion of my studies and has been instrumental in the advancement of my research endeavors.

Lastly, I am immensely grateful to my family, as well as my husband Gaojie Jin, for their unwavering support and unconditional love. Without their presence and encouragement, none of my achievements would have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

“Do not go gentle into that good night,
Old age should burn and rave at close of day;
Rage, rage against the dying of the light.”
(Dylan Thomas)

The remarkable success of Deep Neural Networks (DNNs) in various fields has led to their widespread adoption and application in numerous industries. The deep learning system is based on the deep neural network (DNN), which is a complex and powerful network that enables machines to learn from data in a way that is similar to the human brain. The ability of DNNs to learn from vast amounts of data and extract complex patterns and representations has made them a powerful tool for solving a wide range of problems. In recent years, DNNs have achieved impressive performance in tasks such as image classification [121], text analysis [92], speech recognition [37] and action recognition [60], surpassing the performance of traditional machine learning models.

Despite their enormous success, extensive research has shown that Deep Neural Networks (DNNs) are vulnerable to adversarial attacks [13], [136], appearing as adding small and imperceptible nonrandom perturbations to inputs that cause DNNs to give incorrect predictions. These perturbed inputs that mislead the DNN are defined as adversarial examples. As in the example shown in Figure 1.1, by adding small noise to the original image, the classifier will give a different prediction for the generated image. Although the DNN obtains high accuracy on the test datasets, it is still susceptible to adversarial examples. As a result, it is critical to investigate adversarial examples in safety-critical scenarios, such as autonomous driving [25] and object detection [175], which can benefit the community to increase awareness of the safety risks in the system and also provide support for the construction of more robust DNNs. There are some notable works to generate adversarial examples including Fast Gradient Sign Methods (FGSM) [172], C&W attack [13], DeepFool [94], and JMSA [155].

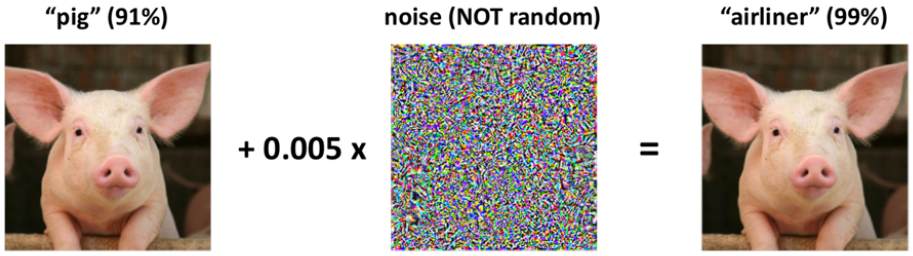


Figure 1.1: The adversarial examples in the image of a pig [66]

To improve the robustness of the model in the adversary environment, various adversarial defence methods have been proposed recently. The primary objective of these defence techniques is to improve the accuracy of neural networks when faced with data perturbed by adversarial attacks. Existing approaches mainly focus on adversarial training. Adversarial training involves incorporating adversarial perturbations during the training process in order to enhance the model's robustness. Empirical results indicate that adversarial training with projected gradient descent (PGD) adversary is currently the most effective method [157].

Studying adversarial defences can help us to defend better against different adversarial threats [46].

However, as Tramer, Carlini, Brendel, *et al.* [139] and Athalye, Carlini, and Wagner [2] indicated,

> *Even though these defences are effective for some attacks, they still can be broken by other stronger attacks.*

Thereby, a more solid solution is needed, ideally with *provable guarantees*, to verify whether the model is robust to *any* adversarial attacks within an allowed perturbation budget, which is also referred to as robustness verification for the DNNs. In the community, a small predefined l_p-norm ball is normally used to quantify such perturbations, namely, within this small perturbing space, the decision should remain the same from the perspective of a human observer.

This thesis delves into an initial exploration of adversarial attacks and defences applied to deep neural networks (DNNs) utilised for video recognition, which are widely used in real systems such as video surveillance [103], self-driving cars [3] and action recognition [60]. These applications are directly related to decisions concerning property security and human health and safety. Although extensive research has been conducted on adversarial attacks targeting images, there remains a significant gap in the examination of video robustness. Therefore, investigating adversarial samples in videos is of both theoretical and practical value. Attacking video models is a more challenging work, as videos have a sequential data structure and change dynamically over time, setting them apart from static images. Although the most straightforward approach to attacking videos might be to treat each frame as an image and attack all frames to increase the fooling rate, this method has its drawbacks. It is time consuming and may compromise human imperceptibility. As a result, attack strategies designed for images cannot be applied directly to videos. In this work, we propose a sparse attack approach for video models by selecting the most critical frame(s) in a video to perturb.

Secondly, we study the robustness verification of DNNs on the 3D object detection task, which takes 3D point cloud data as input. 3D object detection is also widely used in safety-critical systems, such as self-driving cars and autonomous robotics, where point cloud data obtained from LIDARs and depth cameras are used to represent 3D objects [20]. Deep Neural Networks (DNNs) have shown remarkable performance in detecting these 3D objects [109], [110]. However, extensive research has revealed the vulnerability of 3D deep learning models to adversarial attacks. These attacks involve the addition or shifting of points in point cloud data and have raised concerns about the safety of such systems [85], [150], [170], [182]. In this work, we proposed a novel and efficient verification framework for the complete PointNet model with a joint alignment network (JANet) containing a multiplication layer.

Lastly, we focus on studying the robustness verification in a challenging setting known as Cooperative Multi-Agent Reinforcement Learning (c-MARLs), which also shows its effective impact in many safety-critical situations like autonomous cars [120]. Robustness

analysis for c-MARL models is of great importance. RL has also been shown to be susceptible to disturbance in observations of an RL agent [4], [50] or in environments [40]. Some adversarial defence works for RL are proposed [27], [30], [122], [132] and then towards these defences, stronger attacks are proposed [116], [117]. To end this repeated game, Wu, Li, Huang, *et al.* [159] and Kumar, Levine, and Feizi [73] proposed to use probabilistic approaches to provide robustness certification for RLs. Concerning c-MARL, Lin, Dzeparoska, Zhang, *et al.* [83] addressed the challenges of attacking such systems and proposed adding perturbations to the state space. To date, the robustness certification on c-MARL has not been touched upon by the community, which motivates me to take a deep exploration in this field. In light of this gap, we propose a novel framework based on the randomised smoothing algorithm, which represents the first attempt to certify the robustness of c-MARLs.

In summary, this thesis focuses on evaluating the robustness of deep neural networks in various safety-critical scenarios that have received relatively little attention in the research community.

1.1 Contributions

1.1.1 Video Attacks and Defenses

To address the challenges in assessing the robustness of videos, we propose a Sparse Adversarial Video Attack for Deep neural networks, called DeepSAVA, which can *i)* capture a wide range of adversarial instances, including both noise contamination and various spatial transformations; *ii)* achieve sparse attack, that is, only perturbing very few frames of a video while still achieving a state-of-the-art attack success rate; and *iii)* obtain strong adversarial transferability across various recurrent models compared with baseline methods. In summary, it has three key technical **contributions**.

- *DeepSAVA is the first work to combine additive and spatial-transformed perturbation for video attacks*. According to image attacks with spatial transformation perturbation [165], [180], perturbing the positions of pixels can improve perceptual realism and make it locally smooth. In DeepSAVA, we introduce a new term in the loss function to optimise both additive and spatial transformation perturbation. With a proper SSIM-based constraint, we can produce strong perturbations combined with additive and spatial transformation. Such a combined perturbation enables DeepSAVA to achieve successful attacks by just perturbing one frame and be effective across various types of DNNs.

- *We are also the first work that uses Bayesian optimisation (BO) to identify the most critical frames of the video in attacks*. To achieve a video attack that can perturb as few frames as possible, we design an alternating optimisation strategy that can effectively

identify the key frames via BO and then initiate additive and spatial-transformed perturbations on the selected keyframes by stochastic gradient descent (SGD) based optimiser. Such an alternating process happens in each iteration of the optimisation until keyframes are found. Combining the above two ingredients, the proposed novel optimisation strategy can achieve a better fooling rate than baselines.

- *Based on our novel perturbation generator, we propose a new adversarial training method to improve the robustness of video classification models*. We perform extensive experiments on different models to evaluate the effectiveness of our PGD with a combined perturbation adversarial training algorithm, which combines additive and spatial perturbations in adversarial training. The results confirm that the new design adversarial training could improve the robustness against both DeepSAVA and Sparse [148] attacks.

1.1.2 Robustness Verification for 3D Point Cloud Models

Motivated by the aforementioned challenges yet to be resolved, we aim to design an *efficient* and *scalable* robustness verification tool that can handle a wide range of 3D models, including those with JANet structures under multiple l_p-norm metrics including l_∞, l_1, and l_2-norm. We achieve the verification efficiently by adapting the efficient layer-by-layer certification framework used in [7], [151]. Considering that these verifiers are designed for images and cannot be applied in larger-scale 3D point cloud models, we employ a novel relaxation function of global max pooling to make it applicable and efficient on PointNet. Moreover, the multiplication layers in the JANet structure involve two variables under perturbations, which bring the cross-nonlinearity. Due to the high dimensionality in 3D point clouds, such cross-non-linearity results in significant computational overhead for computing a tight bound. To solve the non-linearity of the cross, Shi, Zhang, Chang, *et al.* [123] proposed using a closed-form linear function to bound the multiplication layer in the attention layers of the Transformer. As the JANet includes multiplication between two variables, where one variable is the output of the previous layer, which makes it different from that in Transformer, we propose to use the closed-form linear functions to bound the multiplication layer and combine forward and backward propagation, which can also benefit the computation cost by calculating the bound in only $O(1)$ complexity.

In summary, the proposed method can achieve efficient verification.

- We design a relaxation algorithm to resolve the cross-non-linearity challenge by combining forward and backward propagation, enabling an efficient yet tight verification of matrix multiplications.

- We design an efficient and scalable verification tool, 3DVerifier, with provable guarantees. It is a general framework that can verify the robustness of a wide range of

3D model architectures, especially it can work on complete and large-scale 3D models under l_∞, l_1, and l_2-norm perturbations.

- 3DVerifier, as far as we know, is one of the very few works on 3D model verification, which is more advanced than the existing work, 3DCertify, in terms of efficiency, scalability, and tightness of the certified bounds.

1.1.3 Certified Cooperative Multi-Agent Reinforcement Learning

We first propose a smoothed policy where each agent chooses the most frequent action when its observation is perturbed, and then we derive the certified bound of perturbation for each agent per step, within which the chosen action of the agent will not be altered. When evaluating the robustness of all agents per time step, to tackle the challenge of accumulating uncertainty, we identify the multiple test problem and propose to *correct* the p-value by multiplying the importance factor of each agent. We then employ the Benjamini-Hochberg (BH) procedure with a corrected p-value to control the selective false discovery rate (FDR).

For the certification of the robustness of the global reward, we propose a tree-search-based algorithm to find the certified lower bound of the perturbation and the lower bound of the global reward of the team under this perturbation. In this work, we focus on certifying the robustness of value-based c-MARLs under a l_2 norm bounded attack. The proposed method can be easily extended to evaluate l_p norm-based robustness by using different sampling distributions, such as the generalised Gaussian distribution as indicated in Hayes [45].

Overall, the contributions can be summarised as:

- For the first time, we propose a solution to certify the robustness of c-MARLs, which is a *general* framework that can verify the robustness of per-state action and the global reward for c-MARLs as well as in a single-agent system.

- We propose a new criterion to enable the *scalable* robustness certification per state for c-MARLs by considering the importance of each agent to reduce the error of selective multiple tests.

- We propose a tree-search-based method to obtain the certified lower bound of the global team reward, which allows a tighter certification bound than the state-of-the-art certification methods.

1.2 Thesis Outline

The structure of the thesis and its related publications are depicted in Figure 1.2. There are a total of six chapters in this thesis.

Chapter 2 compromises the background and related work for the research in this thesis, such as the background of various deep neural networks and robustness.

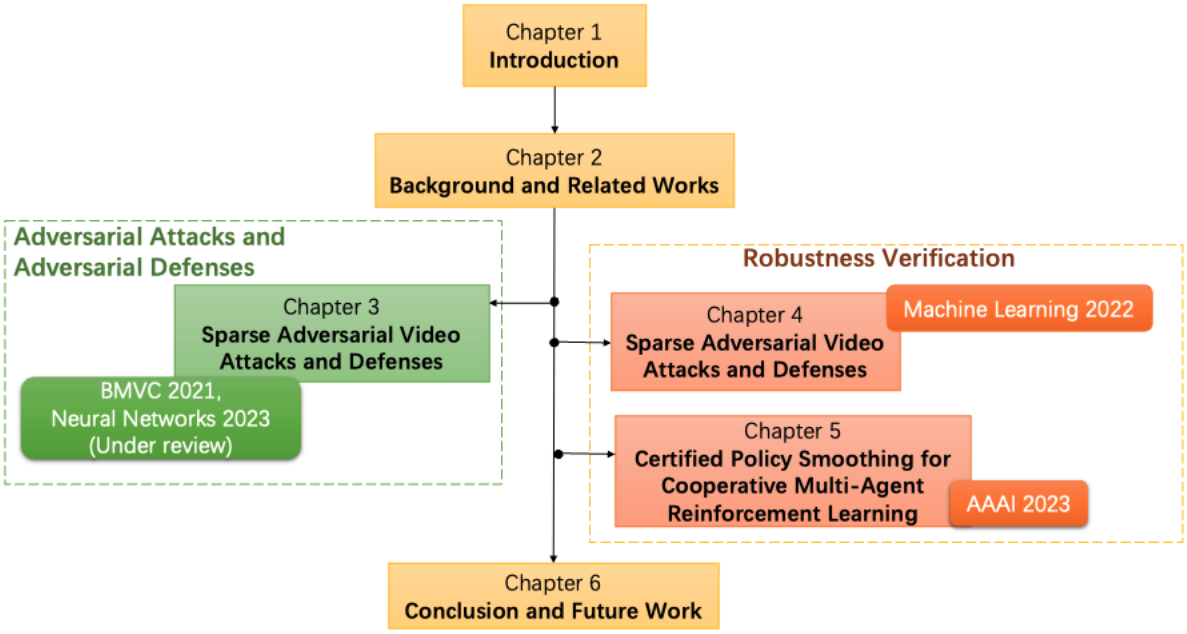


Figure 1.2: Outline of the thesis. The left-hand chapter concentrates on adversarial attacks and adversarial training, while the right-hand chapters focus on robustness verification.

Chapter 3 shows our first work that performed adversarial attack and adversarial defence on video models. The proposed approach, called DeepSAVA, employs a unified optimisation framework that incorporates both additive perturbation and spatial transformation. The adversarial distance is measured using the structural similarity index measure (SSIM). An efficient alternative optimisation scheme is developed that combines Bayesian optimisation to identify the most influential frames in a video and SGD-based optimisation to produce perturbations on those frames. DeepSAVA achieves a highly effective attack on videos while maintaining human imperceptibility and achieving state-of-the-art performance in terms of both attack success rate and adversarial transferability. Toward the sparse attack approach, we then propose an effective adversarial defence method based on an adversarial training algorithm. Extensive experiments conducted on various deep neural networks and video datasets confirm the superior performance of DeepSAVA and the effectiveness of the proposed adversarial training approach compared to the state-of-the-art adversarial training with projected gradient descent (PGD) adversaries.

In Chapter 4, we focused on the robustness verification of 3D point cloud models. A more comprehensive and effective framework has been proposed to verify large-scale point cloud models with JANet, which includes multiplication layers. This framework addresses key challenges such as the handling of cross-non-linearity operations in the multiplication layers and the high computational complexity of high-dimensional point cloud inputs and added layers. Our solution, named 3DVerifier, tackles both issues by employing a linear relaxation function to bound the multiplication layer and combining forward and backward propagation processes to compute global bounds for the point cloud models. Our extensive experimental results confirm the effectiveness of our method and show that the certified bounds obtained by our framework are significantly tighter than the current state-of-the-art method.

Chapter 5 presents the certification of the robustness of Cooperative Multi-Agent Reinforcement Learning (c-MARL). Since this is the first work to certify MARL, the chapter begins by addressing the challenges associated with certification, including the accumulation of uncertainty as the number of agents increases and the potential lack of impact when changing the action of a single agent into a global team reward. To overcome these challenges, the chapter proposes using the false discovery rate (FDR) controlling procedure and considering the importance of each agent to achieve per-state robustness certification. Additionally, a tree-search-based algorithm is proposed to find a lower bound of the global reward under the minimal certified perturbation.

Finally, in Chapter 6, we provide a summary of the thesis and discuss future research directions that stem from our findings.

Chapter 2

Background and related works

A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal Turing Machine.
(Alan Turing)

2.1 Deep Neural Network

The deep learning system can be viewed as a software system based on the deep neural network (DNN) with representation learning [119]. It has revolutionised many areas of artificial intelligence, including computer vision [121], speech recognition [37], action recognition [60] and natural language processing [92].

In classic DNN, neurons in one layer are fully connected with all neurons in the next layer, and the weights assigned to each edge could reveal the strength of the connection. It is designed to learn the underlying structure of the data by iteratively adjusting the weights of the neurons in the network. The structure of DNN is illustrated in Figure 2.1.

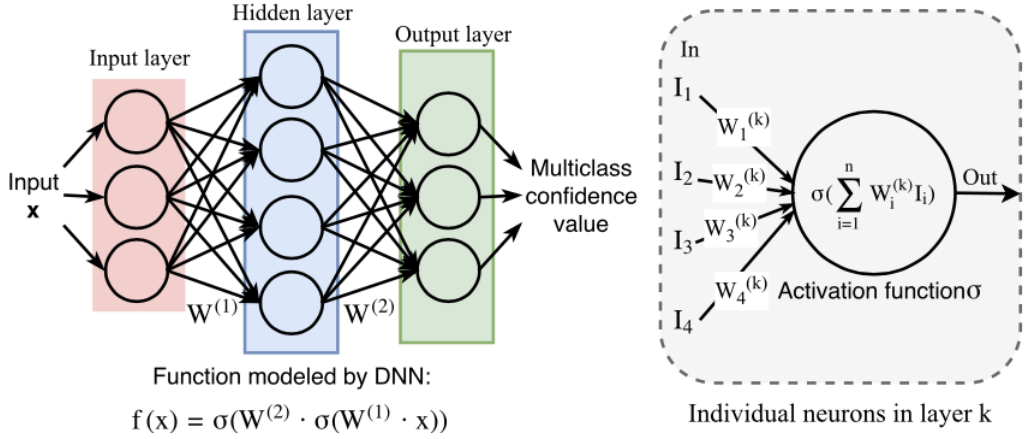


Figure 2.1: Left: DNN architecture for a 3-layer function model $f(x)$ with input x and weight W^k for each layer k; σ denotes the active function. Right: individual neuron in layer k with input $x = [I_1, I_2, I_3, I_4]^T$ and its corresponding weights W^k [107].

2.1.1 Convolutional Neural Network

CNNs are widely used in computer vision tasks such as object detection, image classification, and segmentation. The hierarchical representations learned by CNNs enable them to identify features at different levels of abstraction, making them highly effective for these tasks.

The filters used in convolutional layers are designed to capture specific features, such as edges or corners, and as they slide over the input data, they create a feature map that highlights where these features are located. By stacking multiple convolutional layers, CNNs can learn increasingly complex features, such as shapes or textures, that are useful for recognising objects in images. Pooling layers are typically used after convolutional layers to reduce the spatial dimensionality of the feature map. This helps to reduce the number of parameters in the network and prevent over-fitting. Pooling can be done in various ways,

such as max pooling or average pooling, which select the maximum or average value in each pool respectively. Finally, the fully connected layers process the flattened output of the convolutional and grouping layers to produce the final output of the network. These layers are similar to those found in traditional neural networks and are responsible for making predictions based on the learned features.

CNNs accept different types of input data based on their dimensions. For instance, they can process one-dimensional sequences like audio signals [47] and text sequences [78]; two-dimensional data such as images [107]; and three-dimensional data like volumetric information or sequences with both spatial and temporal aspects, such as video [173]. The distinguishing factor in the CNN architecture across these dimensions lies in the size of the convolutional filter. For one-dimensional data, the filter takes the form of a single row, moving along the sequence. In the case of two-dimensional data, it adopts a 2D matrix structure, sliding over the height and width of images. As for three-dimensional data, the filter becomes a 3D tensor, traversing the dimensions of height, width, and depth or time.

2.1.2 Recurrent Neural Networks

RNNs are particularly well suited for processing sequential data because of their ability to maintain a state or memory over time steps. This enables them to capture long-term dependencies in the data, making them highly effective for tasks like speech recognition, language modeling, and sentiment analysis.

One of the key features of an RNN is its ability to use the output from the previous time step as input to the current time step. This feedback loop allows the network to capture dependencies between sequential data points and generate predictions based on this context. However, in situations where the previous state affecting a current prediction occurred several sentences ago, the RNN may find it challenging or impossible to establish the connection and make accurate predictions. The RNN cannot store the information in long-sentence because the vanishing gradient problem, that the gradient vanishes quickly in earlier layers. To solve this problem, more advanced variants are proposed.

Long Short-Term Memory (LSTM)

[49] proposed the Long Short-Term Memory (LSTM), which is a popular RNN architecture, to address the problem of vanishing gradients and long-term dependencies in traditional RNNs. At the core of the LSTM architecture are memory cells that are able to maintain information over long periods of time. These cells are controlled by three gates: the input gate, the output gate, and the forget gate. The input gate controls how much new information should be added to the memory cell, while the forget gate controls how much old information should be discarded. The output gate controls how much of the memory cell's current state should be used to produce the network's output.

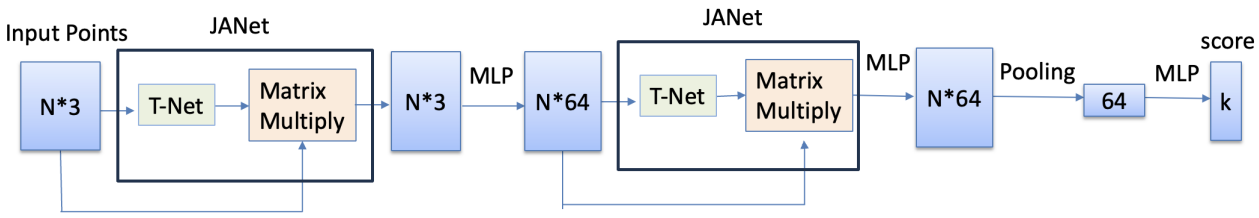


Figure 2.2: The abstract framework of the PonitNet with joint alignment network (JANet), where MLP stands for multi-layer perceptron. The network takes N points as input. It applies input and feature transformation and use max pooling to aggregate point features. The output is classification scores for k classes.

The gates are controlled by sigmoid activation functions that output values between 0 and 1, indicating the amount of information that should be allowed through. The input gate and forget gate are multiplied element-wise with the current memory cell state, while the output gate is multiplied with the cell's output. This enables the network to selectively store or discard information from previous time steps based on the current input. In addition to the memory cells and gates, LSTMs also have a peephole connection that allows the gates to peek into the current state of the memory cell. This additional information helps the network to better regulate the flow of information through the gates and produce more accurate output.

Gated Recurrent Units (GRUs)

Gated Recurrent Units (GRUs) proposed by [22] were introduced as an alternative to Long Short-Term Memory (LSTM) networks. Like LSTMs, GRUs aim to address the problem of vanishing gradients and the inability of RNNs to capture long-term dependencies. GRUs use a gating mechanism to regulate the flow of information within the network. However, unlike LSTMs, GRUs use only two gates, an update gate and a reset gate, to control the flow of information.

The update gate determines how much of the previous hidden state should be retained and how much of the new input should be added to it. The reset gate determines how much of the previous hidden state should be forgotten and how much of the new input should be used to create a new hidden state. The update and reset gates in GRUs allow the network to selectively remember or forget information, making it possible to capture long-term dependencies in the data.

Video Action Recognition Models

The video classification task primarily focuses on action recognition [67]. The works on video classification using DNNs are developed in two ways: using 2D or 3D-based convolution neural networks (CNN). Since the CNNs have obtained state-of-the-art performance in image classification, Karpathy *et al*. [62] first proposed to use 2D CNN to classify each frame of the video. Szegedy *et al*. then developed the Inception-v3 [134], [135], which is commonly used as a baseline classification model. As 2D-CNNs use incomplete video information, some works added layers containing temporal information, such as LSTM, to combine CNN features extracted over time, which is referred to as CNN+LSTM model [26], [100]. As for the 3D CNNs [141], it can learn temporal features from videos by inputting all frames in three dimensions directly. [16] proposed a two-stream inflated 3D CNN (I3D) to build the 2D kernel first and then merge the pooling layer and kernel into a 3D network. By pre-training the I3D on Kinetics Dataset, it could reach state-of-the-art performance on recognising UCF101 and HMDB51 action video datasets.

2.1.3 PointNet

Point cloud models are digital representations of physical objects or environments, comprising numerous discrete data points in three-dimensional space. These data points are typically obtained using 3D scanning technologies such as LiDAR (Light Detection and Ranging) or structured light scan and are widely used to represent 3D objects for the deep learning classification task.

Introduced by [110], PointNet is a method for processing point clouds, which are sets of points in 3D space. A key challenge of classifying 3D point clouds is that the input to the neural network consists of a subset of points with three crucial properties: They are unordered, have interactions between points, and are invariant under transformation. To address these properties, PointNet includes three features: a max-pooling layer to collect information from all points, a combination module for local and global information, and two joint alignment networks that align both input points and feature of the points. This design enables PointNet to process unordered point cloud data directly, a significant challenge in 3D deep learning.

PointNet transforms each point into a higher-dimensional feature space and aggregates information from all points using a symmetric function like max pooling or mean pooling. The resulting global feature vector can be utilised for various tasks such as classification or segmentation. Figure 2.2 illustrates the abstract architecture of a complete PointNet. JANet, which employs T-Net and matrix multiplications, allows PointNet to achieve geometric invariant functionality, as the learned representations are expected to be invariant to spatial transformations. Recent studies also show that JANet is critical to improve PointNet performance [1], [18], [102], making it widely applicable in safety-critical tasks.

PointNet is renowned for its simplicity and efficiency, as it can be trained end-to-end

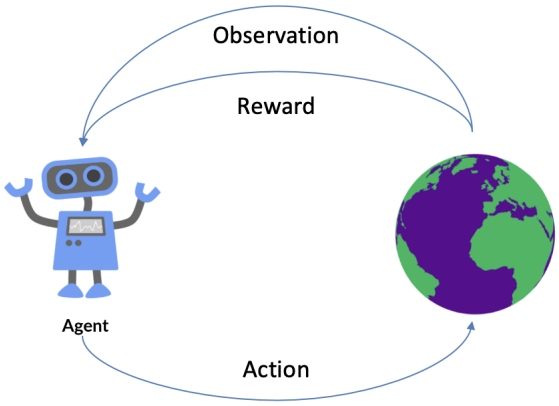


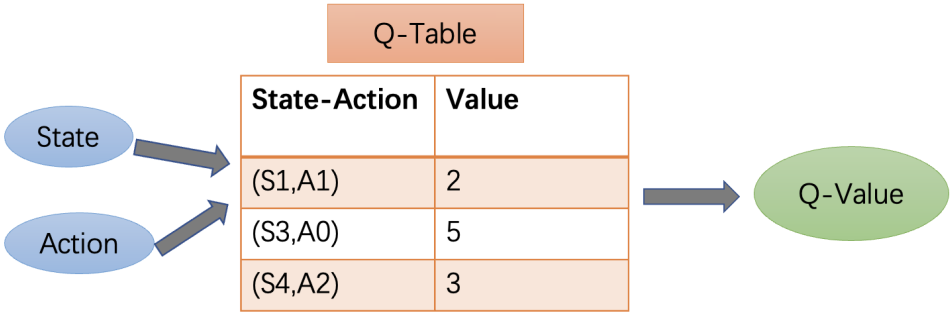
Figure 2.3: The abstract diagram to illustrate the relationship between agent and environment in reinforcement learning

on raw point cloud data without the need for additional preprocessing or feature extraction steps. This makes it a powerful tool for various applications in 3D computer vision.

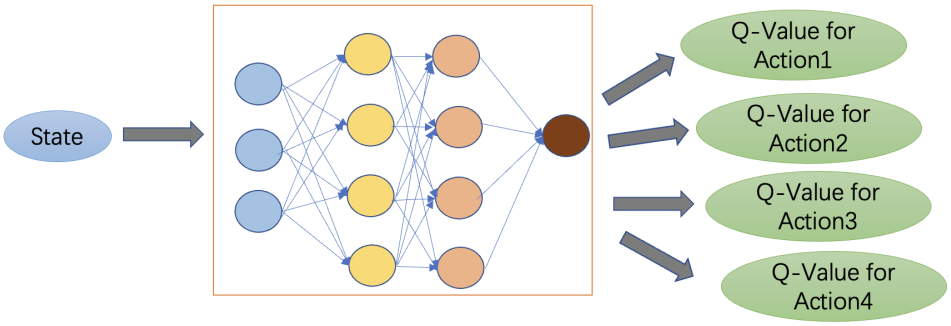
2.1.4 Reinforcement Learning

Reinforcement learning is a type of machine learning that involves training an artificial intelligence (AI) agent to make decisions in an environment by providing feedback in the form of rewards or punishments. In reinforcement learning, the AI agent learns through trial and error, receiving positive or negative feedback depending on whether its actions achieve a desired outcome or not. Reinforcement learning (RL) aims to find the best actions for agents that can optimise long-term reward by interacting with the surrounding environments [11], [113]. When there is a team of agents, the system needs to jointly optimise each agent's actions to maximise the reward of the team. This type of learning is often used in scenarios where there is no clearly defined "correct" answer, and the AI agent must learn through exploration and experimentation.

Reinforcement learning (RL) comprises five key components: environment, state, reward, policy, and value. As illustrated in Figure 2.3, the environment refers to the physical world in which the agent operates. The agent takes action, and the environment provides feedback in the form of reward and observation. The state represents the current situation of the agent, and the policy maps the agent's actions to the state. Lastly, the value corresponds to the future expected reward an agent would receive by taking an action in a specific state. When building an optimal policy, the agent must navigate the trade-off between exploring new states and maximise overall reward, known as the exploration and exploitation dilemma. Achieving the best overall strategy may require making short-term sacrifices, balancing the



(a) Q-Learning



(b) DQNs

Figure 2.4: Comparison between Q-Learning and DQNs

need for immediate reward against the potential for long-term gains.

The RL problems can be modelled as a Markov decision process (MDP), which consists of a set of states S and sections A. In MDP, the transition model $P(s_{t+1} = s'|s_t = s, a_t = a)$ represents the probability from state s to state s' under action a and the immediate reward after transition from s to s' with action a can be formulated as $R_a(s, s')$. The state-action value of $Q = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$ represents the expected future reward r_t, discounted by γ^t, from taking an action in a given state/observation. At the time step t, the agent obtains the current state s_t and reward r_t, and then it can choose an action a_t. After receiving the action, the environment will move to new state s_{t+1} and reward r_{t+1}, which determines the new transition (s_t, a_t, s_{t+1}). Therefore, the reinforcement learning agent aims to learn the policy $\pi : A \times S \rightarrow [0, 1], \pi(a, s) = \Pr(a_t = a \mid s_t = s)$ to maximise the expected accumulated reward.

Q-Learning

Real-world environments are more likely to lack any prior knowledge of environment dynamics. Model-free RL methods come in handy in such cases, which do not require prior knowledge of the environment's transition probabilities. Q-learning is a value-based model-free RL algorithm, that given the state s, it aims to find the best action a to maximise the expected cumulative reward. The core feature is that it involves maintaining a table, known as the Q-table, that maps states and actions to expected rewards. The Q-table is initialised with arbitrary values, and the agent interacts with the environment by taking actions and observing rewards. With each action, the Q-table is updated using the observed reward and the maximum expected future reward for the next state. The updated value is a function of the current value and the learning rate, which determines the degree to which new information overrides old information. The update function is shown in Equation 2.1, which is also referred to as the Bellman equation, where the α is the learning rate, γ is the discount factor, and $\max_a Q(s_{t+1}, a)$ represents the estimate of optimal future value. In summary, the equation indicates that the agent updates its perceived reward by adding the estimated optimal future reward, assuming it takes the best-known action at present. When implemented, the agent will explore all possible actions for a given state and select the state-action pair with the highest corresponding Q-value.

$$Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot \max_a Q(s_{t+1}, a)) \tag{2.1}$$

Over time, the Q-table is updated through a process of trial and error, gradually converging on the optimal policy for the MDP. Q-learning has been used in a variety of applications, such as game-playing and robotics. It is easy to implement but hard to handle the situation where the states are unseen, which causes a lack of generality.

Deep Q-Networks(DQNs)

Deep Q-Networks(DQNs) are introduced by [93], which uses a neural network to approximate the Q-function. The Q-function represents the expected future rewards of taking a certain action in a given state. It has been used to achieve impressive results in a wide range of applications, including playing Atari games [93], navigating robots [106], and even playing the game of Go [124].

The basic idea behind DQN is to train a deep neural network to predict the Q-values of all possible actions given a particular state. The network is trained using a combination of supervised and reinforcement learning, where the target Q-values are updated using a Bellman equation, which is a recursive equation that calculates the expected reward of each action. One of the key advantages of DQN is that it can learn directly from raw sensory input, such as pixels from a camera, without the need for hand-crafted features or domain-specific knowledge. In Figure 2.4, we compare the structure of Q-learning and DQNs. As DQNs

use a neural network to approximate Q-values instead of a table, it is able to handle large, complex problems with continuous states and actions.

2.1.5 Cooperative Multi-agent Reinforcement Learning

Cooperative Multi-agent Reinforcement Learning (c-MARL) deals with the study of multiple agents learning to interact and cooperate with each other in an environment to achieve a common goal. The goal of MARL is to develop algorithms that enable agents to learn from their interactions with each other and with their environment to maximise their collective rewards.

The main challenge in multi-agent systems is to model the interactions between agents and their environment, where the reward function of each agent may depend not only on its own actions but also on the actions of other agents. One approach to address this challenge is to use decentralised learning algorithms that allow each agent to learn its own policy or value function. However, it does not guarantee that the agents will coordinate effectively. Another approach is employing centralised learning algorithms to learn a joint policy or value function for all agents, which can achieve better coordination but may suffer from scalability issues. In the c-MARL, as the number of agents increases, the joint action space of the agents can grow exponentially. As a result, centralised training is proposed to decentralise policy learning [101], where each agent learns its own policy based on its local action-observation history first and then forms the centralised action value conditioned on the global state and joint action. Most c-MARL methods use this scheme, such as value decomposition networks (VDN) [133] and QMIX [113]. The key challenges of c-MARL are that the environment is non-stationary as each agent can change its action and once the model is trained, the information shared in the team is limited [179], which make it harder for the c-MARL to choose the optimal actions.

In this thesis, we consider a fully cooperative multi-agent game G as a Dec-POMDP [71], which is defined by the tuple $G = \langle S, \mathcal{A}, P, r, Z, \mathcal{O}, N, \gamma \rangle$, in which each agent $n \in \{1, 2, ..., N\}$ chooses an action $a^n \in \mathcal{A}$ in each state $s \in S$ to form the joint action $\mathbf{a} = \{a^1, a^2, ..., a^N\}$. The same reward function is shared by all agents $r(s, \mathbf{a})$. γ is a discount factor. We suppose that each agent draws an observation $z^n \in Z$ given the observation function $\mathcal{O}(s, \mathbf{a})$.

Each agent has a stochastic policy $\pi^n(a^n|h^n)$ where h^n is the action-observation history $h^n \in \mathcal{H}$. The joint policy π has a joint discount return $R_t = \sum_{i=0}^{\infty}(\gamma^i r_{t+i})$ and an action-value function: $Q^\pi(s_t, \mathbf{a}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{a}_{t+1:\infty}}[R_t|s_t, \mathbf{a}_t]$. Given an action-value function Q^π, we define a greedy policy as $\pi(s_t) : \arg\max_{\mathbf{a}_t \in \mathcal{A}} Q^\pi(s_t, \mathbf{a}_t)$ that returns the optimal action.

VDN

Value-Decomposition Networks (VDN) [133] was proposed as a solution that combines the benefits of both decentralised and centralised learning. It decomposes the Q-value function

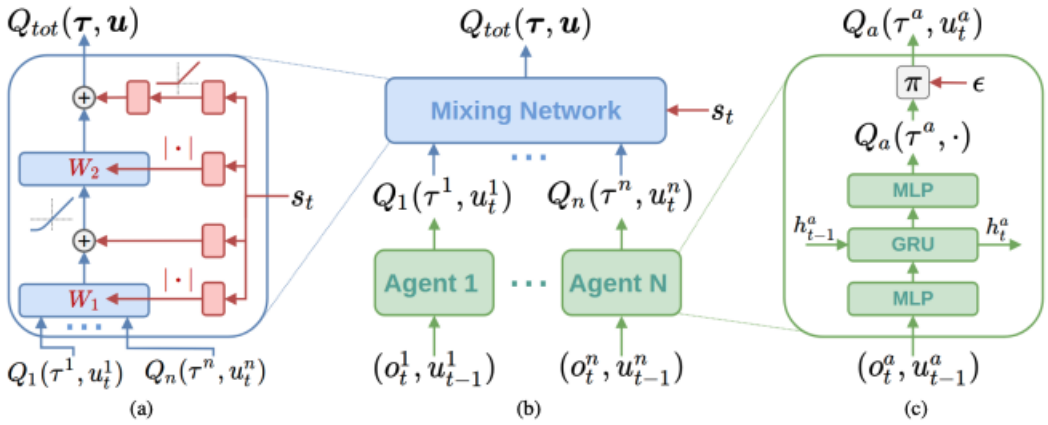


Figure 2.5: a. Structure of mixing network. b. Overall architecture of QMIX. c. Structure of agent network, where the best viewed in colour.[113]

of a team into a sum of individual Q-value functions for each agent, allowing each agent to learn its own Q-value function while still coordinating effectively with other agents.

In VDN, each agent maintains its own Q-value function, which is a function of the local observation and the joint action of all agents. The Q-value function for the team is then obtained by summing up the Q-values of all agents. This decomposition of the Q-value function allows the agents to learn how to coordinate with each other while still maximising their own individual rewards. The training process of VDN involves using experience replay, in which the agent stores and randomly samples past experiences to update its Q-value function. VDN also uses target networks to stabilise the learning process and prevent the Q-value function from oscillating during training. However, since the individual Q-values are only combined through simple summation to derive the global Q-value, and the decentralized policy is updated independently for each agent, this approach can restrict the complexity of the centralized action-value representation and disregard additional state information during the training process.

QMIX

To improve the VDN, instead of using a separate network for each agent, [113] proposed the QMIX to use a mixing network that takes as input the individual Q-values and outputs the global Q-value. The key insight of QMIX is to ensure that a global argmax operation on the total Q-value function produces the same result as a series of individual argmax operations on each individual Q-value function, which constrain the reward in a more general form, which is defined as:

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in \mathcal{A},$$

where Q_{tot} denotes the joint value function computed by the mixing network and Q_a represents each agent's value function from agent networks.

The QMIX architecture is shown in Figure 2.5. For each agent a at time t, the agent network takes the observation o and last action u as input to output the individual value function $Q_a\left(\tau^a, u_t^a\right)$. The mixing network operates as a neural network that propagates forward. By ingesting $Q_a\left(\tau^a, u_t^a\right)$ and a collection of weights as input, it merges the distinctive value function of each agent and produces the output values for Q_{tot}. The weights dictate the relative influence of each agent's Q-value on the overall Q-value.

Through a back-propagation process, the mixing weights are iteratively refined. This process aims to train the mixing weights in a way that optimises the global Q-value function, ultimately enhancing the collaboration among agents.

2.2 Robustness

Despite deep neural networks (DNNs) achieves excellent performance in pattern recognition tasks, they also exhibit significant robustness issues. Even minor, imperceptible non-random adversarial perturbations added to natural input can easily fool DNNs, which can result in blind confidence in incorrect predictions. The adversarial examples are first proposed in [136], in which the small perturbations were injected into the inputs to fool the DNNs in high probability. The robust classification error can be defined as

$$\mathcal{E}_{rob}(f_\theta(x)) = \mathbb{E}_x \max_{x' \in \mathbb{B}_p(x,\epsilon)} \vec{1}\{f_\theta(x') \neq y\}, \tag{2.2}$$

where $\mathbb{B}_p(x, \epsilon) = \{x' \in \mathbb{S} : ||x' - x||_p \leq \epsilon\}$, $\mathbb{S}$ is the input space, y represent the true label, and $\vec{1}\{f_\theta(x') \neq y\}$ serves as a discriminative function that assesses whether the prediction differs from the true label. These examples that are misclassified are referred to as the adversarial examples and the technology to generate the adversarial examples according to the DNN is called adversarial attacks. To train a more robust DL system, the goal of adversarial training is to minimise the training loss of the samples with perturbations which is generated by maximising the classification error. This goal can be interpreted as a min-max optimisation [66].

2.2.1 Adversarial Attack

There are several methods for generating adversarial examples. In this section, we will introduce five common algorithms in detail.

Fast Gradient Sign Method (FGSM)

[41] proposed the FGSM technology which generates the adversarial examples by updating the sample along with the sign direction of each pixel via one-step gradient:

$$x_{new} = x + \alpha \text{sign}\left(\nabla_x J(\theta, x, y)\right), \tag{2.3}$$

where J represents the cost function of the DNN, the θ denotes parameters of the model, x is the input vector, α represents the step size to ensure the perturbations are small, and y is the predicted result of x. When training a model, the goal is to minimise the loss and optimise the network parameters to achieve higher accuracy. However, when generating adversarial examples using FGSM, the goal is to maximise the loss. This is achieved by using the update function in Equation 2.3.

Basic Iterative Method (BIM)

[75] extended the FGSM algorithm to generate adversarial perturbation iteratively for a small parameter ϵ. The generation formula can be represented below:

$$x_0 = x \quad \text{and} \quad x_{n+1} = \text{Clip}_{x,\xi}\left\{x_n + \epsilon \,\text{sign}\left(\nabla_x J\left(\theta, x_n, y\right)\right)\right\}. \tag{2.4}$$

The $\text{Clip}_{x,\xi}(\cdot)$ could ensure the generated adversarial samples within the range of ϵ-district of the original input x.

The Projected Gradient Descent (PGD)

The PGD [90] is proposed to improve the FGSM. It works by first setting the input data to a random point and then using the basic iterative process to create adversarial data.

Jacobian-based Saliency Map Attack (JSMA)

Papernot *et al.* [104] introduce the JSMA for the target classification, which applies the Jacobian matrix firstly for the input x. This algorithm follows a greedy approach, wherein it conducts numerous iterations, with each iteration modifying a single pixel at a time to amplify the desired misclassification target.

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{i \times j} \tag{2.5}$$

where F_j is the output logit score for the target label j and x_i represents the i-th pixel. To classify the sample to the target class j, the probability of the output of class j should be increased and for other classes should be decreased until the class j engages the highest probability.

Carlini and Wagner (CW)

Calini and Waner [14] designed the adversarial attack technique to solve the optimisation problem to update the perturbation η :

$$\begin{array}{ll} \min_{\eta} & \|\eta\|_p + \lambda \cdot J(\theta, x+\eta, y) \\ \text{s.t.} & x+\eta \in [0,1]^n \end{array} \tag{2.6}$$

where $p \in \{0, 2, \infty\}$. The term $\|\eta\|_p$ restricts the η within l_p norms and the parameter λ is introduced to balance the two loss functions. They proposed seven versions of objective functions J. Other gradient-based optimisation algorithms to generate perturbation were also proposed [13], [86], [137], [164].

These works mentioned above only apply additive perturbation to pixels. Some works [58], [76], [77], [158], [165] use a functional perturbation which is non-additive-only perturbation like spatial transformation. These perturbations slightly modify the location of pixels. Some work such as [43], [58], [183] also utilise other types of metrics such as SSIM to quantify human perception, but none of them explored the SSIM-guided spatial transformation. For more details on adversarial attacks, please refer to our recent survey [51] and tutorial [6], [114].

2.2.2 Adversarial Defence

The adversarial defence is proposed to improve the robustness of the model, which aims to reduce the power of adversarial examples. [41] first proposed to inject the adversarial examples into the training dataset to retrain the model. However, this method is time-consuming, and the achieved robustness of the retrained model relies on the power of the injected adversarial examples. Then, [90] first build the structure of adversarial training by adding multi-step projected gradient descent (PGD) in the training stage, which is considered the most effective way for adversarial defences [2], [55]. [176] then proposed the method to establish a trade-off between model accuracy and robustness by adding an extra loss term of adversarial examples in the training stage. There is a rising number of works achieved adversarial defences based on adversarial training [61], [105], [140], [163].

In addition to adversarial training, some works attempt to adopt other methods, such as detection techniques [13], [15], [33], [91], provable defences [54], [56], [57], [63], [112], [127], [156], and preprocessing algorithms [9], [44], [128]. From the empirical perspective, adversarial training with PGD [90] adversary still appears to engage the most robust performance against a wide range of adversarial attacks [80], [90].

Given the input x and output y in the set S, the min-max function is shown in Equation 2.7

$$\min_{\theta} \frac{1}{|S|} \sum_{x,y \in S} \max_{||\delta|| \leq \epsilon} J(\theta, x, y). \tag{2.7}$$

The most intuitive way to train a robust model is to generate the adversarial examples first

Algorithm 1 PGD adversarial training [90]

Input: $\mathbf{X}^{T\times W\times H\times C}$; y; M; training epochs T; dataset batch size N; PGD steps K; step size α

1: **for** $t \leftarrow 1, T$ **do**
2: **for** $i \leftarrow 1, N$ **do**
3: $\sigma = 0$
4: **for** $k \leftarrow 1, K$ **do**
 //Perform PGD Adversarial Attack
5: $\sigma = \sigma + \alpha \cdot$
 $(\nabla_\sigma \ell_{adv}\left(\mathbf{1}_{y_i}, J(\mathbf{X}_i + \sigma; \boldsymbol{\theta})\right))$
 //Update the model weights
6: $\boldsymbol{\theta} = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \ell\left(J(\boldsymbol{\theta}; \mathbf{X}_i + \sigma), \mathbf{1}_{y_i}\right)$

and then feed them into the training process. Usually, the parameter θ can be optimised by the stochastic gradient descent and the inner maximisation problem can be solved by the adversarial attacks mentioned previously, such as PGD and FGSM. As the FGSM can be viewed as one step on PGD, in practice, it works worse than PGD. The overall adversarial training process with PGD attack is shown in Figure 3.5 (b), and its algorithm is sketched in Algorithm 1. As we can see from Algorithm 1, the key improvement of PGD adversarial training is to perform multiple small steps α to estimate the inner maximisation problem (lines 4-5).

2.2.3 Robustness Verification

As generating adversarial attack examples can be one strategy to solve the inner optimisation problem:

$$\max_{||\delta|| \leq \epsilon} J(\theta, x, y),$$

it can be referred to as finding a lower bound to achieve the optimisation goal, which is also can be viewed as a method to empirically solve the optimisation problem. On the other hand, we can consider how to exactly solve it or to upper bound the optimisation. Instead of exactly finding the solution for the inner maximisation, we can determine whether adversarial examples exist within a specific region. This concept is also referred as verification. A neural network verification algorithm aims to guarantee a safe region, where any perturbed input cannot defer the prediction of the model. Given the input x and a region

$$\mathbb{B}_p(x', \epsilon) := \{x' | \|x' - x\|_p \leq \epsilon\},$$

which is defined as a l_p norm ball around the input sample, suppose $\operatorname{argmax} f(x) = c$, the robust verification aims to verify $\operatorname{argmax} f(x') = c$ for all $x' \in \mathbb{B}_p(x', \epsilon)$. For the

classification model, the objective can be formulated as the following equation:

$$\min_{x' \in \mathbb{B}_p(x', \epsilon)} f_c(x') - f_t(x'), \forall t \neq c, \tag{2.8}$$

By ensuring that the optimisation function's minimisation is greater than zero,we can guarantee that the logit output of class c for any input x' within the region $\mathbb{B}$ is always greater than any other class. As this is a nonconvex optimisation problem, exactly solving the problem is a challenging task. To compute the robustness guarantees, two types of verifiers are proposed: incomplete and complete verifiers.

Complete Verifiers

The complete verifier aims to ensure there are no adversarial examples in a certain region or find an adversarial example within the region to mislead the verifier. Some works have explored in this direction, such as the algorithm based on satisfiability modulo theories (SMT) [63], and mixed integer programming (MIP) [10], [138]. The Reluplex algorithm proposed by [63] encodes the single ReLU function into a pair of variables to split the non-linear function into two linear programming functions and then formulate a set of constraints to justify whether the assumption is satisfied. The maximisation problem can be formulated as a combinatorial optimisation problem. As for the methods based on MIP, it can be solved by converting the problem into a mixed integer linear program (MILP), which compromises a linear objective, linear inequality, and equality constraint for each neuron using binary variables. However, these methods are shown NP-hard [175], thus they can only be employed for the small model to find the exact solution.

Incomplete Verifiers

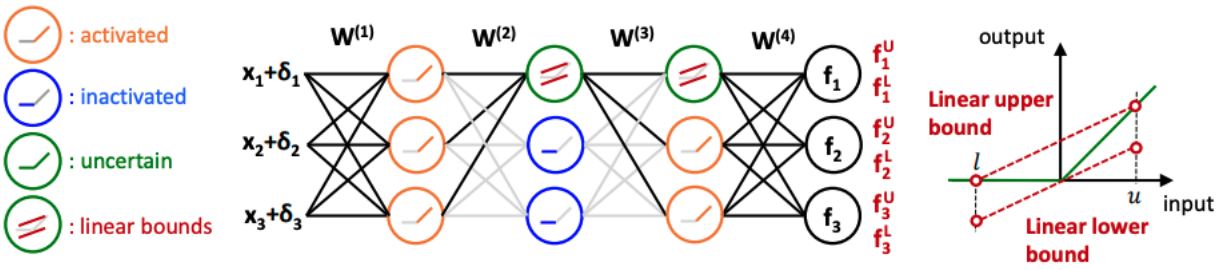


Figure 2.6: Demonstration of the process to compute the bounds for the output of neural network with ReLU active function [175].

The incomplete verifier aims to approximate the bounds of the model's output, and there are various methods to certify the robustness of neural networks by bounding the

output. As the main challenge to compute the output of the neural network is the non-linear activation function [175] proposed the algorithm, Fast-Lin to use a linear approximation for the non-linear activation function to derive the certified lower bound of minimum adversarial perturbation.

Given the perturbed input x within l_p norm ball $\mathbb{B}_p(x', \epsilon)$, suppose the lower bound and upper bound of the neuron pre-ReLU layer is l and u, following linear function can be used to bound the output of ReLU activation $\Gamma(y)$:

$$\frac{u}{u-l}y \leq \Gamma(y) \leq \frac{u}{u-l}(y-l),$$

In Figure 2.6, we illustrate the utilisation of a linear function for constraining the output of the ReLU activation. Therefore, to find the maximum possible lower bound of the perturbation ϵ, as Equation 2.8 suggested, the output of lower bound of logit output of original class $f_c^L(x+\epsilon)$ and upper bound for any other class $f_t^U(x+\epsilon)$ should be minimised by adjusting the value of ϵ, under the condition $f_c^L(x+\epsilon) > f_t^U(x+\epsilon)$.

As Fast-Lin can only be used to verify the basic neural networks with ReLU activation function, [178] then proposed a more general framework, CROWN, to improve it by allowing it to handle various active functions. In CROWN, instead of using two parallel linear bounds to estimate the output of ReLU, it applies two linear bounds with different slopes which makes it more precise and efficient. However, both these frameworks can only be used to verify the fully connected layers, [7] then proposed a more general framework, CNN-Cert to handle more complex convolutional neural networks. On the other hand, [118] proposed to use the convex relaxation to approximate the bounds for the non-linearity operations. Following this direction, [156] and [28] introduced a dual approach to form the relaxation. Additionally, several studies computed the bounds via abstract domain interpretation [39], [126], symbolic interval analysis [145], and semi-definite approximation [29], [112]. Above mentioned verification algorithms are based on the image domain, and they have been extended to various scenes like energy conversation [111], shape variants [126], Transformer [123], and 3D point clouds [87].

Randomized smoothing is also an incomplete verifier, which is a probabilistic method based on Monte Carlo sampling. Below we give a more detailed knowledge of it.

Randomized Smoothing

Randomized smoothing [24] was developed to evaluate probabilistic certified robustness for classification tasks. It aims to construct a smoothed model $g(x)$, which can produce the most probable prediction of the base classifier $f(x)$ over perturbed inputs from Gaussian noise in a test instance. The smoothed classifier $g(x)$ is supposed to be provably robust to l_2-norm bounded perturbations within a certain radius:

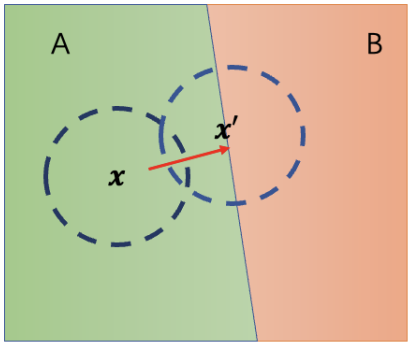


Figure 2.7: Demonstration of the proof of Theorem 1. The set in green represents class A and orange represents class B. The circle is the distribution sampled from $\mathcal{N}(x, \sigma^2 I)$ and $\mathcal{N}(x', \sigma^2 I)$. As the perturbation increases in the direction perpendicular to the boundary plane, the centre of distribution moves from x to x', resulting in the maximum distortion.

Theorem 1. *[24] For a classifier $f : \mathbb{R} \to \mathcal{Y}$, suppose $c \in \mathcal{Y}$, let $\delta \sim \mathcal{N}(0, \sigma^2 I)$, the smoothed classifier be $g(x) := \arg\max_c \mathbb{P}(f(x+\delta) = c)$, suppose $\underline{p_A}, \overline{p_B} \in [0, 1]$, if*

$$\mathbb{P}(f(x+\delta) = c_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq c_A} \mathbb{P}(f(x+\delta) = c), \tag{2.9}$$

then $g(x + \epsilon) = c_A$ for all $||\epsilon||_2 \leq R$, where

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})). \tag{2.10}$$

Here Φ^{-1} is the inverse cumulative distribution function (CDF) of the normal distribution. To be noticed, it cannot guarantee that a given model is robust against all possible adversarial attacks but rather provides a probabilistic guarantee that the model is robust with high probability. The level of confidence in the guarantee depends on the number of times the smoothing procedure is repeated and the variance of the noise distribution. The Theorem 1 is derived based on Neyman-Pearson [99] lemma, and also can be proved by the modified version.

Lemma 1. *[99] Let X and Y be random variables in $\mathbb{R}^d$ with densities μ_X and μ_Y. Suppose $h\ \mathbb{R}^d \to 0, 1$ is a random or deterministic function. Then: If S:$\{z \in \mathbb{R}^d : \frac{\mu_Y(z)}{\mu_X(z)} \leq t\}$ for some $t > 0$ and $\mathbb{P}(h(X) = 1) \geq \mathbb{P}(X \in S)$, then $\mathbb{P}(h(Y) = 1) \geq \mathbb{P}(Y \in S)$. On the other hand, if $\{z \in \mathbb{R}^d : \frac{\mu_Y(z)}{\mu_X(z)} \geq t\}$ for some $t > 0$ and $\mathbb{P}(h(X) = 1) \leq \mathbb{P}(X \in S)$, then $\mathbb{P}(h(Y) = 1) \leq \mathbb{P}(Y \in S)$.*

Lemma 2. *(Neyman-Pearson for Gaussians with different means)[24] Let X and Y be random variables from Gaussian distributions, $X \sim \mathcal{N}(x, \sigma^2 I)$ and $Y \sim \mathcal{N}(x + \delta, \sigma^2 I)$.*

Algorithm 2 Randomized Sampling [24]

Input: Pre-trained classifier f; input sample x;

Parameter: sampling size M; Gaussian distribution parameter σ; confidence parameter α

function PREDICTION(f, x, M, σ, α)
  $Cnts \leftarrow$ SAMPLE(f, x, M, σ)
  $c_A^*, c_B^* \leftarrow$ Top two indices in $cnts$
  $n_A, n_B \leftarrow cnts[c_A^*], cnts[c_B^*]$
  if BINPVALUE($n_A, n_A + n_B, =, 0.5$) $\leq \alpha$ **then**
    return c_A^*
  else
    return $None$
function CERTIFY($f, x, \sigma, M_0, M, 1-\alpha$)
  $cnts^0 \leftarrow$ SAMPLE(f, x, M_0, σ)
  $c_A^* \leftarrow$ top index in $cnts^0$
  $cnts \leftarrow$ SAMPLE(f, x, M, σ)
  $\underline{p_A} \leftarrow$ LOWERCONFBOND($cnts[c_A^*], n, 1-\alpha$)
  if $\underline{p_A} > \frac{1}{2}$ **then**
    return prediction c_A^* and radius $\sigma\Phi^{-1}(\underline{p_A})$
  else
    return None

Suppose h $\mathbb{R}^d \to 0, 1$ *is a random or deterministic function. Then we can have:*
a. If $S = \left\{z \in \mathbb{R}^d : \delta^T z \leq \beta\right\}$ *for some* β *and* $\mathbb{P}(h(X) = 1) \geq \mathbb{P}(X \in S)$*, then* $\mathbb{P}(h(Y) = 1) \geq \mathbb{P}(Y \in S)$.
b. if $S = \left\{z \in \mathbb{R}^d : \delta^T z \geq \beta\right\}$ *for some* $t > 0$ *and* $\mathbb{P}(h(X) = 1) \leq \mathbb{P}(X \in S)$*, then* $\mathbb{P}(h(Y) = 1) \leq \mathbb{P}(Y \in S)$.

Proof. Proofs are provided in Appendix A in [24]. I suggest you read more details in [24] about randomized smoothing to gain a better understanding. □

As a result, the region of each class can be defined as a hyperplane that is orthogonal to the direction of the distortion, as illustrated in Figure 2.7. To demonstrate the algorithm, we briefly present the algorithm for certification in a binary case in the Algorithm 2. The SAMPLE in the algorithm used the Monte-Carlo sampling to sample M noises ϵ from Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$ and obtains $f(x+\epsilon)$. The procedure, PREDICTION, involves gathering the counts of each class and utilising the top two classes in a two-sided binomial p-value test to assess whether the most frequently selected class should be returned. In the function CERTIFY, the Clopper-Pearson confidence interval [23] is applied to compute the lower bound of the probability p_A. As indicated in Theorem 1, by replacing the $\overline{p_B}$ with $1 - \underline{p_A}$, the certified radius R can be calculated. If the function CERTIFY returns a class and

radius R instead of $None$, we can guarantee that the Theorem 1 holds with confidence $1-\alpha$ on the radius R.

2.2.4 Structural Similarity Index Measure (SSIM)

The SSIM was first proposed by Wang and Bovik [146] and is detailed in Wang, Bovik, Sheikh, *et al.* [147]. Given x and $\hat{x}$ as the local pixels taken from the same location of the same frame in the clean video and adversarial video, respectively, the local similarity between them can be computed on three aspects: structures ($s(x,\hat{x})$), contrasts ($c(x,\hat{x})$), and brightness values ($b(x,\hat{x})$). The local SSIM is formed by these terms [147]:

$$\begin{aligned} S(x,\hat{x}) &= s(x,\hat{x})^{\alpha}\cdot c(x,\hat{x})^{\beta}\cdot b(x,\hat{x})^{\gamma} = \\ &\left(\frac{\sigma_{x\hat{x}}+D_1}{\sigma_x\sigma_{\hat{x}}+D_1}\right)^{\alpha}\cdot\left(\frac{2\sigma_x\sigma_{\hat{x}}+D_2}{\sigma_x^2+\sigma_{\hat{x}}^2+D_2}\right)^{\beta}\cdot\left(\frac{2\mu_x\mu_{\hat{x}}+D_3}{\mu_x^2+\mu_{\hat{x}}^2+D_3}\right)^{\gamma}, \end{aligned} \tag{2.11}$$

where the parameters α, β, and γ are positive values that govern the relative significance of the three components; μ_x and $\mu_{\hat{x}}$ denote means, σ_x and $\sigma_{\hat{x}}$ are standard deviations of x and $\hat{x}$, respectively; $\sigma_{x\hat{x}}$ represents the cross-correlation of x and $\hat{x}$ after deleting means; D_1, D_2, and D_3 are weight parameters. Following Wang, Bovik, Sheikh, *et al.* [147], we set $D_1 = 0.01, D_2 = 0.03, D_3 = 0.015$ and $\alpha = \beta = \gamma = 1$. For the SSIM metric, a value of 1 means that the two images compared are the same.

As we mentioned before, the SSIM is less sensitive to the combination of additive and spatial perturbations [1] and more similar to human perception than l_p-norms [185]. Because the SSIM is differentiable with respect to the input variable, in this chapter we apply SSIM to calculate the similarity loss to constrain the perturbation during the optimisation process. The overall score of SSIM of the video is calculated by summing up SSIM loss over all frames of the video.

[1] For convenience, we use *combined perturbation* for short in this chapter.

Chapter 3

Sparse Adversarial Video Attacks and Defences

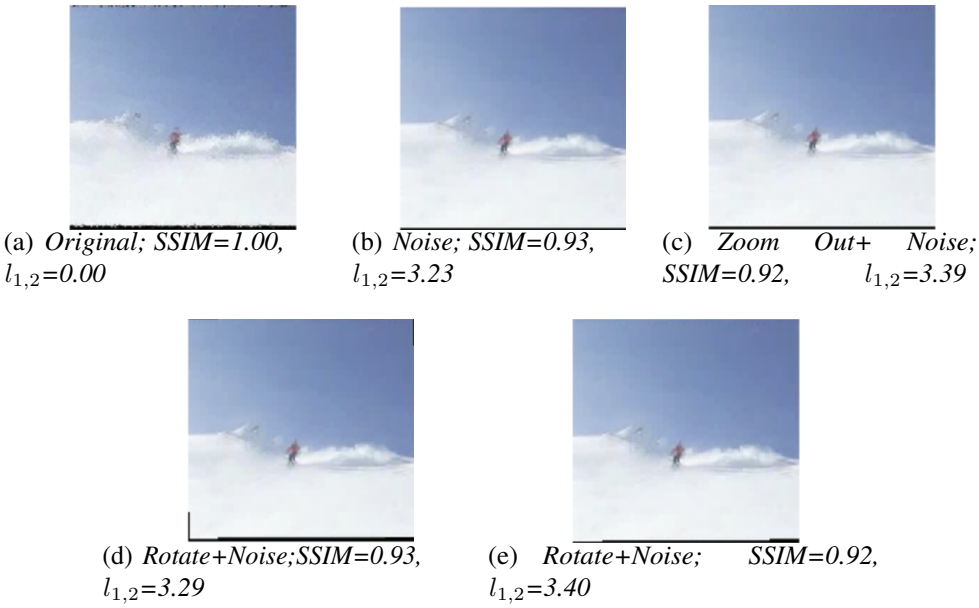
"A single idea from the human mind can build cities. An idea can transform the world and rewrite all the rules."
(*"Inception"*)

3.1 Introduction

As Deep Neural Networks (DNNs) are shown to be vulnerable to adversarial attacks, to improve the robustness of the model in the adversary environment, various adversarial defensive methods have been proposed recently. These attack strategies primarily concentrate on image-related tasks, yet the adversarial robustness of deep learning models in videos has not been thoroughly explored. Recently, a number of works [53], [98], [148], [167] are aware of the values of the adversarial attacks on videos. In practice, DNNs that process videos are widely applied in real systems such as video surveillance [103], action recognition [60], and autonomous driving [160]. In particular, most of these applications directly relate to decisions about property security or human health and safety. As a result, investigating adversarial samples on videos is urgently needed.

Theoretically, attacking videos is more challenging compared with images because videos contain temporal information. So video attack not only requires achieving minimal adversarial distance but also needs to perturb as few frames as possible, which is referred to as sparse attacks on videos. As such, identifying the most *effective* frame(s) and generating *competitive* perturbation upon those frame(s) are of huge importance to the success rate of the attack. Another important consideration is efficiency. As perturbing each frame of the video is time-consuming, we expect to perform the influential frame identification and adversarial perturbation simultaneously so that we can maintain human imperceptibility and achieve a high attacking success rate. In Table 3.1, we compare our method with existing related work on video attacks in six aspects in Table 3.1.

To achieve such *sparse* adversarial attack, one important criterion is that the perturbed example should resemble a real-world instance as close as possible. Current video attack strategies all adopt the l_p-norm metric to measure the fidelity of the perturbed examples. Although the l_p norm is effective in capturing noise contamination, it is sensitive to natural-occurring transformations such as rotation, spatial shift, and scaling [185]. Taking Figure 3.1 as an example, where the original video frame is modified by different types of perturbation: additive Gaussian noise, spatial scaling, and slight rotation, and both $l_{1,2}$ and structural similarity (SSIM) are given. In Figure 3.1(b) and (d), one frame has only noise added, while the other has a small rotation and noise added. The results demonstrate that the SSIM values of the two frames are identical, whereas the $l_{1,2}$ norm varies. This indicates that *SSIM exhibits lower sensitivity towards small modifications*, such as slight rotation or scaling of pixels, while the l_p-norm distance shows a noticeable difference. The use of the $l_{1,2}$-norm exacerbates this limitation, as even a negligible spatial transformation perturbation can significantly amplify the adversarial distance. Consequently, attacks constrained by the l_p-norm are unable to capture certain spatial transformations that naturally occur in real-world scenarios, such as camera shaking, vibration, or rotation, thereby limiting the effectiveness of the attack. Furthermore, the Image Quality Assessment community has demonstrated that SSIM is a superior alternative signal fidelity measure compared to the l_p-norm in applications where human perceptual criteria matter [185]. Overall, SSIM is



(a) *Original; SSIM=1.00,* $l_{1,2}$*=0.00*

(b) *Noise; SSIM=0.93,* $l_{1,2}$*=3.23*

(c) *Zoom Out+ Noise; SSIM=0.92,* $l_{1,2}$*=3.39*

(d) *Rotate+Noise;SSIM=0.93,* $l_{1,2}$*=3.29*

(e) *Rotate+Noise; SSIM=0.92,* $l_{1,2}$*=3.40*

Figure 3.1: Comparison of SSIM and $l_{1,2}$ norm distance for: (a) Original image. [(b)-(e)] Perturbed images: (b) Noise. (c) Zooming-out spatial scaling + noise. (d) Counterclockwise rotation 5 ° + noise. (e) Counter-clockwise rotation 5 ° + spatial scaling with zooming + noise. SSIM values for (b) and (d) are identical, whereas $l_{1,2}$ increases when an imperceptible rotation is introduced. This indicates that SSIM is less sensitive to rotation and can potentially result in a stronger adversarial attack with spatial transformation perturbation.

less sensitive to both noise and spatial transformations, such as rotation and scaling, thus better aligning with human perception. In this chapter, we employ spatial transformation perturbations, making the SSIM-based loss function more suitable for constraining the distance between adversarial and clean videos, enhancing the effectiveness and efficiency of DeepSAVA.

Moreover, in a real-world scenario, we may not be able to access the parameters, structures, or even datasets of a pre-trained DNN. Thus, similar to adversarial attacks on images, a strong adversarial transferability that can work across diverse unseen models is desirable. However, unlike DNNs for images that are without temporal structure, video models are more complicated and include diverse neural units for recurrent operations. Hence, achieving satisfying adversarial transferability on adversarial video attacks is also very challenging.

In this chapter, we demonstrate our methodology from two aspects: sparse adversarial attacks and adversarial training. The flow chart of our sparse attack method, DeepSAVA, is illustrated in Figure 3.2. In order to demonstrate the effectiveness of DeepSAVA, we compare its performance with the baseline [148] and [98] on three different models. To ensure a fair comparison, we first evaluate their performance under the constraint of limited iterations and then compare them by fixing the l_p-norm and SSIM of the added perturbation. Additionally, an ablation study is conducted to demonstrate the effectiveness of spatial transformation and Bayesian optimisation. Then we show the model accuracy and fooling rate of the model trained by different numbers of adversarial examples. Finally, we perform the transferability experiments on the UCF101 dataset for different recurrent models to demonstrate the transferability of our method across different models.

As for existing works of Adversarial attack on videos, [148] claimed that they are the first to attack videos. Instead of attacking each frame of a video, they apply additive perturbations on randomly selected frames and use $l_{2,1}$ norm to guide the gradient-based optimisation and evaluated the performance on the CNN+LSTM model. [81] used a GAN network to generate offline universal perturbations for each frame. [19] proposed to append a noise frame to the end of videos, which is obtained based on all videos. [98] applied flickering temporal perturbations on each frame to generate universal perturbations for the I3D model. [53] first proposed a black-box approach to attack videos. [149] proposed to use a heuristic method and [167] used a reinforcement learning algorithm to select the keyframes to perform a black-box attack. However, these works only applied additive perturbation based on l_p-norm distance. Our work applies the SSIM-guided non-additive perturbation on selected frames to generate adversarial videos efficiently. We also propose a novel alternating optimisation strategy to select the keyframes.

In Table 3.1, we compare our method with existing related works on video attacks in six aspects. Our work applies the SSIM-guided non-additive perturbation on selected frames to generate adversarial videos efficiently. In this chapter, we propose a novel alternating optimisation strategy to select the keyframes. Additionally, as existing adversarial defences

Table 3.1: Comparison with related works (Flickering [98], RL [167], Heuristic [149], Append [19], BlackBox [53], GAN-based [81], and Sparse Attack [148]) in different aspects.

	Flicker	RL	Heuristic	Append	BlackBox	GAN-based	Sparse	Deep-SAVA
Similarity metric	l_p	l_p	l_1	l_∞	l_∞	l_p	$l_{2,1}$	SSIM
Spatial-transformed perturbation	✗	✗	✗	✗	✗	✗	✗	✓
Additive Perturbation	✓	✓	✓	✓	✓	✓	✓	✓
Identify Key Frames	✗	✓	✓	✗	✗	✗	✗	✓
Transferability Study	✗	✗	✗	✓	✓	✗	✓	✓
Sparse Attack	✗	✓	✓	✗	✗	✗	✓	✓
Adversarial Training	✗	✗	✗	✗	✗	✗	✗	✓

mainly focused on images, and there is no work to improve the robustness of the video classification model against spatial-transformed perturbations, thus, we adapt the existing adversarial training methods on images to videos to improve its robustness on both spatial transformation and additive perturbation.

3.2 Methodology

3.2.1 Attack Problem Definition

The video classifier is defined as $J(\cdot;\boldsymbol{\theta})$ with pretrained weights $\boldsymbol{\theta}$. The input clean video is defined as $\mathbf{X} = (x_1, x_2, ..., x_T) \in \mathbb{R}^{T\times W\times H\times C}$, where T is the length of the video (number of frames), and W, H, C represents the width, height, and the number of channels of each frame; its adversarial video generated is represented as $\hat{\mathbf{X}}$. In order to obtain the adversarial example, the original video is perturbed by a spatial transformer $\mathcal{S}$, and additive noise $\mathcal{D}$. Given that the ground truth label of input video $\mathbf{X}$ is y, the objective function is:

$$\arg\min \lambda \ell_{similar}(\hat{\mathbf{X}}, \mathbf{X}) - \ell_{adv}\left(\mathbf{1}_y, J(\hat{\mathbf{X}};\boldsymbol{\theta})\right), \tag{3.1}$$

where $\mathbf{1}_y$ is the one-hot encoding of y; $\ell_{similar}$ is the similarity loss function to measure the distance between generated adversarial and original video; ℓ_{adv} is the loss function to

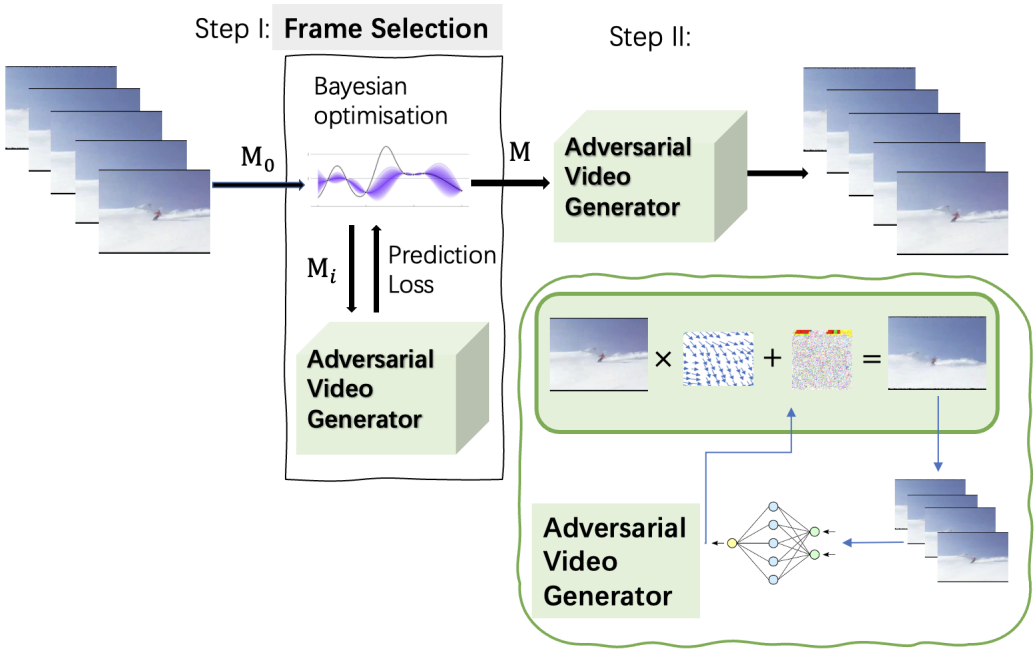


Figure 3.2: Overview of DeepSAVA. Step I is to select the most critical frame: the key frame mask indicator M is alternately identified using Bayesian Optimization (BO), which takes the initial mask indicator M_0 as input and iteratively updates M_i by interacting with the adversarial generator to obtain the prediction loss ($\mathcal{L}$). Step II is to generate the adversarial example using the adversarial generator which incorporates additive and spatial transformation perturbations to generate adversarial examples for the selected frame.

measure the difference between ground truth and prediction label. The parameter λ is set to balance these two loss terms. Additionally, the cross-entropy is used to calculate the ℓ_{adv}, which is proved to be effective in [148].

3.2.2 Sparse Spatial Transform Adversarial Attack

We will now introduce our algorithm for performing attacks with a combined perturbation. The main idea of the algorithm is to apply the combined perturbation on selected frames to achieve a high fooling rate.

Sparse Attack

Formally, the mask indicator $M = (m_1, m_2, .., m_T) \in \mathbb{R}^T$ is used to choose the keyframes in the video, where $m_t \in \{\mathbf{0}, \mathbf{1}\}$ indicates whether the t-th frame is masked to be perturbed. The masked video $\mathbf{X}_m$ is formed through the map function $\mathcal{M}(M, \mathbf{X})$, and then fed into the spatial transformer $\mathcal{S}$.

Spatial Transformed Perturbation

In this chapter, we consider the spacial transformation as a learnable component. Given the t-th frame $x^t \in \mathbb{R}^{W \times H \times C}$ of input video $\mathbf{X}$, x_n^t denotes the n-th pixel of x^t and its location in the frame can be represented by a 2D coordinate (h_n^t, v_n^t). The spatial transformer [52], $\mathcal{S}$, is a differentiable model composed flow displacement vectors

$$\mathbf{U} = ((\Delta\mathbf{H}^1, \Delta\mathbf{V}^1), (\Delta\mathbf{H}^2, \Delta\mathbf{V}^2), ..., (\Delta\mathbf{H}^T, \Delta\mathbf{V}^T)) \in \mathbb{R}^{T \times 2 \times H \times W}$$

where $\mathbf{H}^t = (h_0^t, h_1^t, ..., h_n^t)$, $\mathbf{V}^t = (v_0^t, v_1^t, ..., v_n^t) \in \mathbb{R}^{H \times W}$, which is used to synthesise the 2D coordinate of adversarial videos. Suppose $\hat{x}_n^t$ with location $(\hat{h}_n^t, \hat{v}_n^t)$ is the adversarial example transformed from x_n^t, given its corresponding spatial displacement flow vector $(\Delta h_n^t, \Delta v_n^t)$. As the flow vector moves from the pixel $\hat{x}_n^t$ in the adversarial frame to its corresponding pixel x_n^t in the input frame, therefore, the location of the original pixel x_n^t can be drawn from $\hat{x}_n^t$ by

$$(h_n^t, v_n^t) = (\hat{h}_n^t + \Delta h_n^t, \hat{v}_n^t + \Delta v_n^t).$$

Considering the sparse attack mask indicator M, we can represent the transformed adversarial video as

$$\hat{\mathbf{X}}_S = \mathcal{S}(\mathbf{U}, \mathbf{X}, M).$$

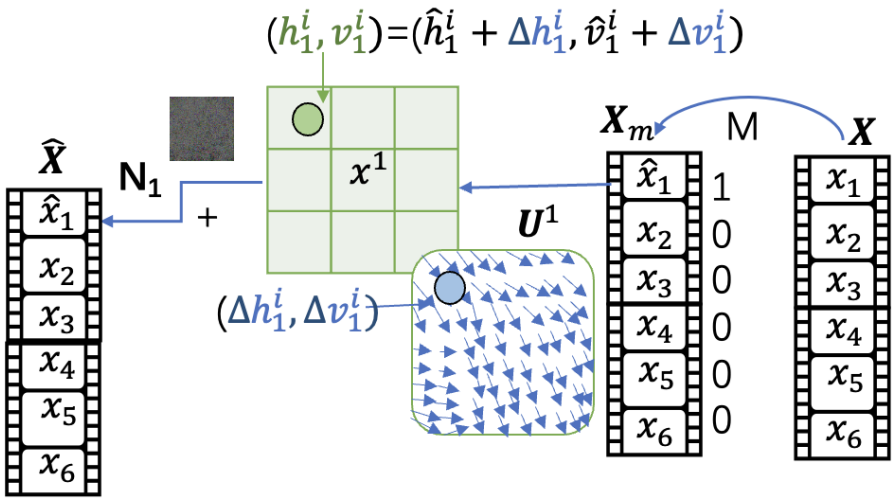


Figure 3.3: The procedure for perturbing a single frame within a video. Initially, the mask indicator M is used to pinpoint the target frame for manipulation. Subsequently, a spatial transformation perturbation is applied, followed by the addition of noise. Finally, the resulting adversarial example $\hat{\mathbf{X}}$ is generated.

Additive Perturbation

The additive perturbation is the most common way to generate adversarial examples [13], [41]. We define the additive model as $\mathcal{D}$ with parameter $\mathbf{N} \in \mathbb{R}^{T\times W\times H\times C}$. We combine spatial transformation and additive perturbation to generate adversarial videos as (illustrated in Figure 3.3):

$$\hat{\mathbf{X}} = \mathcal{D}(\mathbf{N}, \hat{\mathbf{X}}_S, M) = \mathbf{N} \cdot M + \hat{\mathbf{X}}_S \tag{3.2}$$

3.2.3 Novel Alternating Optimisation Strategy

In this work, we utilise Bayesian Optimisation (BO) to select the most critical frame(s). Here, the term "critical frame(s)" refers to the frame for which applying the specified combination results in the greatest reduction in prediction loss compared to other frames. It's important to note that the user can define the number of critical frames to be targeted, and this number may vary for different models in order to achieve optimal attack performance. As the frame selection is a discrete variable optimisation problem, we also tried other discrete optimisation techniques such as simulated annealing (SA) [38] and genetic algorithms (GA) [153], but both spent about 200s to find one critical frame which is much longer than about 16s taken

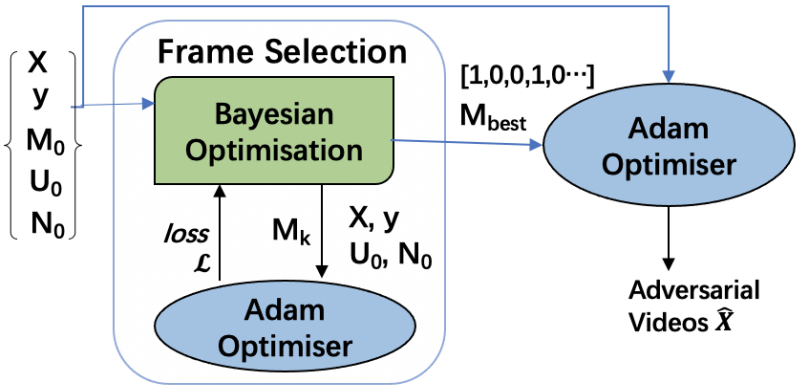


Figure 3.4: The systematic optimisation process by using Bayesian optimisation and Adam Optimiser.

by BO.

The generated adversarial video is formed as

$$\hat{\mathbf{X}} = \mathbf{N} \cdot M + \mathcal{S}(\mathbf{U}, \mathbf{X}, M).$$

In this chapter, the similarity loss $\ell_{similar}$ and adversarial loss ℓ_{adv} in Equation 3.1 can be expressed as

$$\ell_{similar}(\hat{\mathbf{X}}, \mathbf{X}) = 1 - SSIM(\hat{\mathbf{X}}, \mathbf{X}) = \mathcal{L}_s(\mathbf{N}, \mathbf{U}, \mathbf{X}, M)$$

and

$$\ell_{adv}\left(\mathbf{1}_y, J(\hat{\mathbf{X}}; \boldsymbol{\theta})\right) = \mathcal{L}_a(\mathbf{N}, \mathbf{U}, \mathbf{X}, M).$$

Therefore, Equation 3.1 can be simplified as:

$$\arg \min_{M,\mathbf{N},\mathbf{U}} \lambda \mathcal{L}_s(\mathbf{N}, \mathbf{U}, \mathbf{X}, M) - \mathcal{L}_a(\mathbf{N}, \mathbf{U}, \mathbf{X}, M). \tag{3.3}$$

As M is a discrete binary vector, which makes problem (4) non-differentiable, the Bayesian optimization (BO) is then utilised to optimise the binary vector M by identifying the critical frame that should be perturbed. It can be solved systematically by a novel alternating optimisation strategy. We initially provide M as input to the Bayesian optimization process. During the search process, BO generates different configurations of M to explore. Importantly, at each iteration, the BO finds a configuration of M and then queries the model output to obtain an evaluation score, which is used to guide the BO search for the next optimal M. Hence, when interacting with the model, the value of M remains fixed, transforming the problem into a differentiable one that can be solved using Stochastic Gradient Descent

(SGD)-based optimisation. In this work, we choose the Adam optimiser [65] because of its robust and fast convergence performance. This process repeats for a fixed number of iterations, continuously improving the solution via both techniques alternatively.

BO proposes sampling points from the search space through acquisition functions to obtain the reward of previous points. We apply expected improvement (EI) as our acquisition function F, which is a widely used function:

$$F = \mathrm{EI}(M) = \mathbb{E}[\max\left(\mathcal{L}(M) - \mathcal{L}\left(M^{+}\right), 0\right)], \tag{3.4}$$

where $\mathcal{L}(M)$ is the loss feedback from Adam optimiser by fixing M; $\mathcal{L}\left(M^{+}\right)$ is the best value obtained so far, and M^{+} is its location.

During the BO process, we will find the best mask indicator through several iterations. In the k-th iteration of BO, we will first sample a candidate M^{k} according to the acquisition function F. Then, the corresponding loss $\mathcal{L}_k$ will be computed by the Adam, which will then affect the next sampled point M^{k+1} for the next iteration. When the BO reaches the maximum exploration number, the M with minimum loss will be fed into the Adam optimiser to generate the final adversarial video. The process is illustrated in Figure 3.4.

Algorithm 3 Bayesian Optimisation for video frame selection

Input: $\mathbf{X}^{T\times W\times H\times C}$; y; G; λ; F; Number of steps to explore K; Number of critical frames to be chosen n.

Output: The frame selection mask indicator is represented by a one-hot matrix M with n '1' entries.

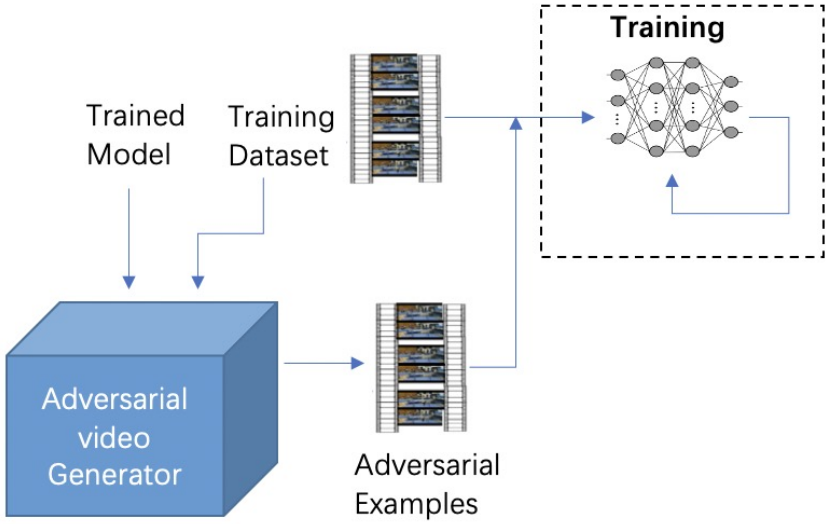
1: Initialise flow network parameter $\mathbf{U}_0$ and additive noise $\mathbf{N}_0$ as random variable from stand norm distribution;
2: **for** $k \leftarrow 1, K$ **do**
3: Find M=$\mathrm{argmax}_M\, F\left(M \mid D_{1:k-1}\right)$
4: Train $G(\mathbf{X}, y, M, \lambda)$ using Adam to obtain $\mathcal{L}$
5: Add $M, \mathcal{L}$ to the dataset $D_{1:k-1}$
6: Return the best M with lowest $\mathcal{L}$.

Algorithm 3 and Algorithm 4 detail the BO selection and adversarial video generation algorithms, respectively. In Algorithm 3, the next sampling point M is obtained by maximising the acquisition function F based on the previous sampling data set $D_{1:k-1}$ (Line 3). After the Adversarial Generator (G) is optimised, the loss $\mathcal{L}$ for M is calculated. Then the M with its corresponding $\mathcal{L}$ are appended to the sampling pool D to propose the next sampling point. In Algorithm 4, according to the optimised mask indicator M, the final flow vector $\mathbf{U}$ and additive noise $\mathbf{N}$ are optimised via Adam.

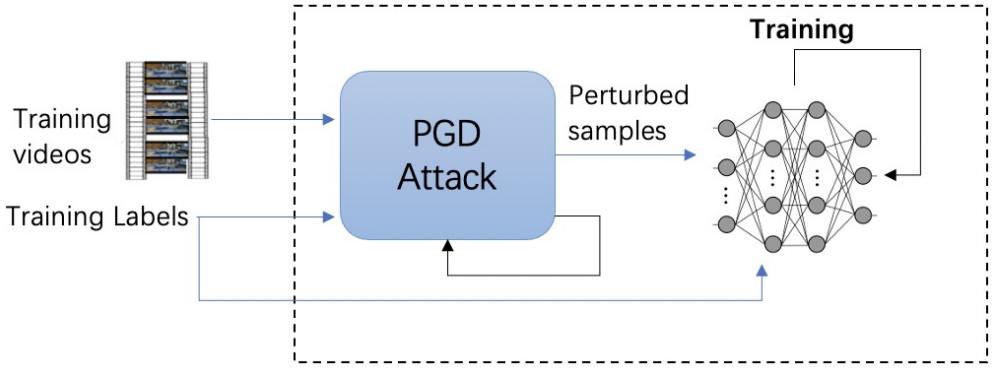
Algorithm 4 DeepSAVA adversarial generator (G)

Input: $\mathbf{X}^{T\times W\times H\times C}$; M; y; λ

1: Initialise flow vector $\mathbf{U}_0$, and additive noise $\mathbf{N}_0$;
2: **for** $step \leftarrow 1, maxStep$ **do**
3: $\quad \hat{\mathbf{X}} = \mathbf{N} \cdot M + \mathcal{S}(\mathbf{U}, \mathbf{X}, M)$
4: $\quad \mathcal{L} = \lambda(1 - SSIM(\hat{\mathbf{X}}, \mathbf{X})) - \ell_{adv}\left(\mathbf{1}_y, J(\hat{\mathbf{X}}; \boldsymbol{\theta})\right)$
5: $\quad$ Apply Adam to optimise $\mathbf{U}$ and $\mathbf{N}$ to minimise $\mathcal{L}$



(a) Adversarial training via injecting adversarial examples



(b) Adversarial training with PGD attack

Figure 3.5: Adversarial Training Overview.

3.2.4 Adversarial Training

Adversarial training is designed as the most intuitive defence mechanism against various adversarial attacks, which incorporates adversarial examples during the training process to improve the robustness of the model. Formally, this goal can be constructed as a min-max optimisation problem. Given the video classifier $J(\cdot;\boldsymbol{\theta})$ parameterised by $\boldsymbol{\theta}$ and the adversarial video input $(\hat{\mathbf{X}}_i, y_i)$ with ground truth label y_i, the objective function can be formulated as follows:

$$\min_{\boldsymbol{\theta}} \sum_i \max \ell_{adv}(J(\boldsymbol{\theta}, \hat{\mathbf{X}}_i), \mathbf{1}_{y_i}), \tag{3.5}$$

where ℓ_{adv} is the adversarial loss. The inner maximisation term aims to find the optimal adversarial examples and it can be approximated by some well-developed adversarial attack algorithms such as PGD [90] and FGSM [172]. The following outer minimisation term represents the traditional training process which aims to minimise the training loss by applying a gradient descent algorithm to optimise the model parameters $\boldsymbol{\theta}$. The generated re-trained model is expected to be more robust against the adversarial attack that is employed in the training process for generating adversarial examples.

The most intuitive way is to inject adversarial examples generated by the adversarial attack to re-train the model, as represented in Figure 3.5 (a). However, this method is time expensive, as we need to spend much more time obtaining thousands of adversarial examples from the training dataset. Goodfellow, Shlens, and Szegedy [41] first proposed to use the Fast Gradient Sign Method (FGSM) to solve the inner maximisation problem. The objective function to approximate the inner maximisation for the FGSM adversarial training can be formed as follows:

$$\hat{\mathbf{X}}_i = \mathbf{X}_i + \alpha \cdot sign\left(\nabla_{\mathbf{X}_i} \ell_{adv}\left(\mathbf{1}_{y_i}, J(\mathbf{X}_i;\boldsymbol{\theta})\right)\right).$$

However, the model trained by the FGSM adversarial training algorithm is still vulnerable to stronger adversarial attacks, such as the PGD attack, which is based on iterative adversarial attacks. Thus, in this chapter, we adapted the PGD optimisation method to solve the inner maximisation problem, which is proved to be effective [90]. As our DeepSAVA framework is a stronger and more effective adversarial attack method on videos, we design a novel adversarial training approach based on traditional PGD adversarial training to defend our adversarial attack with a combined perturbation, as shown in Algorithm 5.

Compared with the PGD adversarial training [90], we identify the main difference between the two methods is that our approach takes the spatial transformation perturbation combined with additive noise into account, which is achieved via the operation presented in Lines 5-6 of Algorithm 5. Despite the adversarial training procedures of the two algorithms being similar, our defence method is more empirically robust against Sparse attack [148] and DeepSAVA attack. We conjecture that the advantage mainly comes from the added combined perturbation term that can perform a more effective attack than additive perturbation only.

Algorithm 5 PGD adversarial training with a combined perturbation

Input: $\mathbf{X}^{T\times W\times H\times C}$; y; M; training epochs T; dataset batch size N; PGD steps K; step size α

Randomly initialise flow network parameter $\mathbf{U}$ and additive noise $\mathbf{N}$;
for $t \leftarrow 1, T$ **do**
  for $i \leftarrow 1, N$ **do**
    for $k \leftarrow 1, K$ **do**
      //Perform Adversarial Attack
      $\hat{\mathbf{X}}_i = \mathbf{N} \cdot M + \mathcal{S}(\mathbf{U}, \hat{\mathbf{X}}_i, M)$
      $\hat{\mathbf{X}}_i = \hat{\mathbf{X}}_i + \alpha \cdot$
      $sign\left(\nabla_{\hat{\mathbf{X}}_i}\left(\ell_{adv}\left(\mathbf{1}_{y_i}, J(\hat{\mathbf{X}}_i; \boldsymbol{\theta})\right)\right)\right)$
    //Update the model weights
    $\boldsymbol{\theta} = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}}\ell\left(J(\boldsymbol{\theta}; \hat{\mathbf{X}}), \mathbf{1}_{y_i}\right)$

3.3 Experiments

3.3.1 Experimental Setup

Dataset

As action recognition video datasets are widely used in adversarial video attack studies, we choose two popular benchmark action recognition datasets to evaluate the performance of our method: UCF101 [129] and HMDB51 [72]. Both datasets are realistic action recognition datasets. The UCF101 contains 13,320 videos with 101 categories such as playing instruments, body movements, and human-object interaction. Similarly, the HMDB51 has around 7,000 videos within 51 categories related to body motion and facial actions.

Action Recognition Models

We evaluate DeepSAVA on three classifiers: *Inception-v3*, a 2D-CNN based model [135], which is widely used in the image recognition task with high accuracy; *I3D*, a 3D-CNN based model, pre-trained on Kinetics [16]; *CNN + LSTM*, which is pre-trained on ImageNet to extract features from videos and then input these features to train the LSTM network. The training accuracy of all classifiers is shown in Table 3.2.

Baseline methods

Two baseline methods are used for comparison, the Sparse [148] and Sparse Flickering [98]. For the works shown in Table 3.1, only [148] is the white-box sparse attack; [149][167] are black-box sparse attack methods. As our work is a white-box sparse attack, we choose the most related one, Sparse [148], as the main baseline. We perform perturbation directly on the frame, while [19] appended an additional frame at the end of the video, which is more visible to humans. So we did not include it as a baseline due to its compromise on the similarity of human perception. In [81], GANs are used to attack real-time video, which is not comparable to our method. We modified Flickering [98], which perturbs all frames, into a sparse one as the Sparse Flickering baseline, but we still show the performance of perturbing all frames.

Experiments Setting

The length of all input videos is crafted to be the same (40 frames). We randomly selected 200 videos from different categories in the test dataset. For those experiments without saying the specific constraint, the maximum allowed search iteration (100 iterations) is applied; all experiments use Adam optimiser with a *0.01* learning rate. The parameter λ is set to 1.5 for the CNN+LSTM model, and 1.0 for the I3D and Inception-v3 models. For λ, values that can balance the fooling rate and perturbation strength are used. As for the step size of the adversarial attack used in the adversarial training, we use the alpha as $\frac{1}{255}$, where 255 is the re-scale image size, which is a commonly used method to determine the step size. It is a heuristic and the optimal value that is a small fraction of the range of pixel values in the image

Metrics

- *Fooling Rate (FR)*: the percentage of generated adversarial videos that are misclassified successfully.
- *Average Number of Iterations (ANI)*: the average number of iterations taken to generate adversarial examples successfully based on the same original videos, which is used to measure the efficiency when we set a constraint on the maximum allowed iteration.

3.3.2 DeepSAVA Attack: Comparison with Baseline Methods

In this section, we will show the comparison results between DeepSAVA and baselines. Since running BO will add extra time to choose the frame, to make the comparison more complete, we also take the DeepSAVA without BO selection into account.

Table 3.2: Training accuracy of the classifiers to be attacked.

Models	UCF101	HMDB51
CNN+LSTM	74%	43%
I3D	94.9%	80%
Inception-v3	71.2%	47%

Limited Iterations:

Since each method uses a different metric, in order to control the maximum allowed perturbation, we limit the number of search iterations for all methods. Each iteration only allows a small amount of perturbation (controlled by the learning rate of Adam optimiser), following the same setup used by the baselines. The results in Table 3.3 show that the ANIs are much below the maximum allowed iteration (100). In Table 3.4, we show the relationship between the iteration and the strength of perturbation. It can be seen that even when it reaches the maximum iteration, the lp-norm and $SSIM$ distances are still acceptable. Given that, setting a constraint on the maximum search number to 100 will not lead to large distortion. We run the experiments 10 times and show the average results with a 99% confidence interval. For the methods without frame selection, the first frame is perturbed. As shown in Table 3.3, BO selection is more efficient than the one without BO. This happens because it is able to select the most critical frame, which can improve efficiency in most cases. For the CNN+LSTM model, DeepSAVA increases the FR slightly compared with the baselines; while for the I3D model, we can see that the FR grows significantly. The BO selection process is also essential for I3D. Without BO, only about half of the test videos can be attacked successfully; after applying BO, the FR increases to nearly 100%. As for the Inception-v3 model, the FR increases when applying DeepSAVA. It can be concluded that the CNN+LSTM is the most robust classification model among the three classifiers. Although the I3D has the highest classification accuracy, it is more vulnerable to attacks, even when only one frame is modified. That might happen because the I3D model relies heavily on the integral structure of the video itself and some frames may be more important.

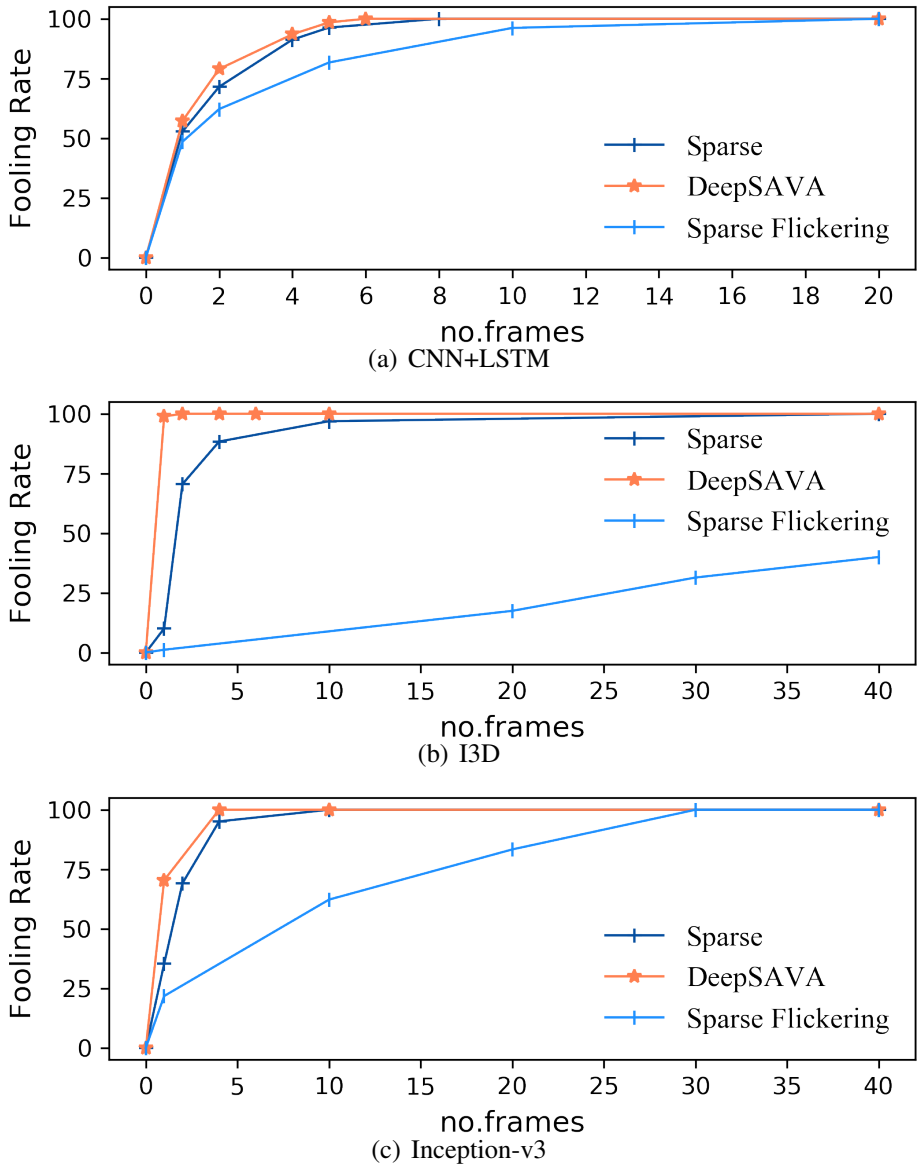
We find that the position of keyframes is related to the classifiers evaluated: for CNN+LSTM, the frames in the front are more often selected, and for other CNN networks, the position is variant. Thus, it is reasonable that the BO cannot improve the FR for the CNN+LSTM model as much as the I3D, as we attacked the first frame when not selecting it. We also show the results in Figure 3.6 for attacking a different number of frames across I3D, CNN+LSTM, and inception-v3 models. It can be seen that the more frames attacked, the higher the fooling rate obtained.

Table 3.3: Comparison with baselines, DeepSAVA without BO and with BO on different models by only perturbing one frame. '-' means that there is no successful attack. Gray cell shows the best results.

Models	Attack Method	UCF101		HMDB51	
		FR	ANI	FR	ANI
CNN+LSTM	Sparse	$52.77\% \pm 2.44\%$	16.45	$95.2\% \pm 1.8\%$	16.4
	Sparse Flickering	$48.48\% \pm 1.67\%$	23.55	$91.94\% \pm 2.93\%$	8.4
	DeepSAVA (without BO)	$56.22\% \pm 1.65\%$	8.32	$99.27\% \pm 0.34\%$	8.42
	DeepSAVA(BO)	$57.22\% \pm 1.36\%$	8.77	100%	6.6
I3D	Sparse	$10.12\% \pm 1.19\%$	44	$5.74\% \pm 1.25\%$	25.1
	Sparse Flickering	$1.15\% \pm 0.68\%$	13	0%	-
	DeepSAVA (without BO)	$47.57\% \pm 2.64\%$	12.15	$46.39\% \pm 3.86\%$	12.2
	DeepSAVA(BO)	$99.89\% \pm 0.11\%$	6.47	$99.92\% \pm 0.08\%$	5.35
Inception-v3	Sparse	$42.25\% \pm 4.30\%$	33.70	$45.82\% \pm 1.56\%$	22.06
	Sparse Flickering	$21.73\% \pm 1.39\%$	35.4	$27.55\% \pm 0.98\%$	27.25
	DeepSAVA (without BO)	$68.86\% \pm 1.83\%$	13.29	$68.98\% \pm 3.19\%$	11.84
	DeepSAVA(BO)	$70.39\% \pm 2.78\%$	10.52	$74.74\% \pm 0.82\%$	9.07

Table 3.4: The relationship between the iteration, $l_{1,2}$, SSIM, and Fooling Rate for the I3D model with combined perturbation on UCF101.

max iter	FR	max(lp)	max(ssim)	ave(lp)	ave(ssim)
30	0.5	0.11	0.094	0.052	0.069
50	0.5	0.135	0.094	0.059	0.099
80	0.529	0.131	0.0959	0.0595	0.081
100	0.529	0.131	0.095	0.052	0.067



(a) CNN+LSTM

(b) I3D

(c) Inception-v3

Figure 3.6: Fooling Rate of attacking the different number of frames across three classifiers.

$l_{2,1}$-norm						
Constraint	$l_{2,1}$ budget = 0.08			$l_{2,1}$ budget = 0.09		
Method	Sparse	DeepSAVA (no BO)	DeepSAVA	Sparse	DeepSAVA (no BO)	DeepSAVA
FR	40.51%	48.1%	88.61%	41.77%	54.43%	93.67%
Time (s)	8018.9	2629	1535.8	14001	3729	1573.82
$SSIM$						
Constraint	$SSIM$ budget = 0.98			$SSIM$ budget = 0.96		
Method	Sparse	DeepSAVA (no BO)	DeepSAVA	Sparse	DeepSAVA (no BO)	DeepSAVA
FR	8.06%	16.56%	35.44%	10.1%	51.9%	96.20%
Time (s)	5842.32	1285.1	1424.4	13789.23	5633.28	1545.5

Table 3.5: Attack I3D model on UCF101 dataset under $l_{2,1}$ and $SSIM$ constraint separately.

Fixed $l_{2,1}$ Norm and SSIM:

For the purpose of a fair comparison, we also present the results under fixed $l_{2,1}$ and SSIM budgets for perturbing only one frame. The maximum allowed iteration is set to 500 to limit the time. As the baseline methods are based on l_p-norm and our method is on SSIM, we take experiments under the same l_p-norm constraint and SSIM constraint, respectively. Based on the results of fixed iterations, we randomly select 200 videos from different categories to attack the I3D model on the UCF101 dataset. During the experiments, the Sparse Flickering spent days to achieve the constraint, thus we will only compare it with the Sparse [148] attack. In [34], the SSIM budget for attacking image is set to 0.95, thus we choose the SSIM constraints above 0.95. In [169], it states that the difference between the images is imperceptible when the $l_{2,1}$ score is 4, given that, we also set the $l_{2,1}$-norm budget to below 0.1 (since $0.1 * 40 = 4$, as we have 40 frames). As we can see in Table 3.5, under small fixed budgets, DeepSAVA outperforms the Sparse [148] in both cases in terms of FR and total time.

3.3.3 DeepSAVA Attack: Effects of λ

To decide the value of λ, we applied the DeepSAVA without BO selection on 200 randomly selected videos of the UCF101 dataset to evaluate the effect of λ. The average success perturbation (ASP) is the average of the SSIM score of perturbation for the adversarial examples that could mislead the model successfully:

$$ASP(SSIM)) = avg(SSIM(V_{adv}, V_{original})),$$

Table 3.6: Comparison with Sparse baseline, DeepSAVA without BO and with BO on different models by only perturbing one frame. Gray cell shows the best results.

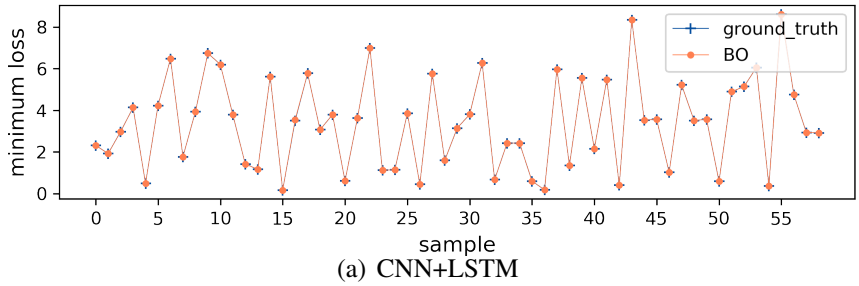
Models	Attack Method	UCF101			
		FR	ANI	AAP ($l_{1,2}$)	AAP (SSIM)
CNN+LSTM	Sparse	54.31%	15.31	0.054	0.043
	DeepSAVA (without BO)	56.94%	7.87	0.077	0.060
	DeepSAVA(BO)	57.11%	8.01	0.071	0.058
I3D	Sparse	11.22%	49	0.092	0.079
	DeepSAVA (without BO)	48.78%	11.34	0.0857	0.054
	DeepSAVA(BO)	99.89%	5.74	0.055	0.0233
Inception-v3	Sparse	41.84%	38.21	0.062	0.0512
	DeepSAVA (without BO)	65.14%	14.88	0.072	0.052
	DeepSAVA(BO)	77.49%	11.43	0.071	0.0508

where V_{adv} denotes the generated adversarial video that could successfully mislead the classifier and $V_{original}$ is the original video. The results of applying $\lambda = 0.8, 1.0, 1.5$ on three models are presented in Table 3.7. We can see that the bigger the λ, the lower the FR while the lower the perturbation. While, for the CNN+LSTM model, the fooling rate remains the same across all tested λ values, but the perturbation level is the lowest at λ = 1.5. Thus, we choose $\lambda = 1.5$ for the CNN+LSTM model and $\lambda = 1.0$ for I3D and Inception-v3 models to trade off the performance in terms of the fooling rate and average success perturbation.

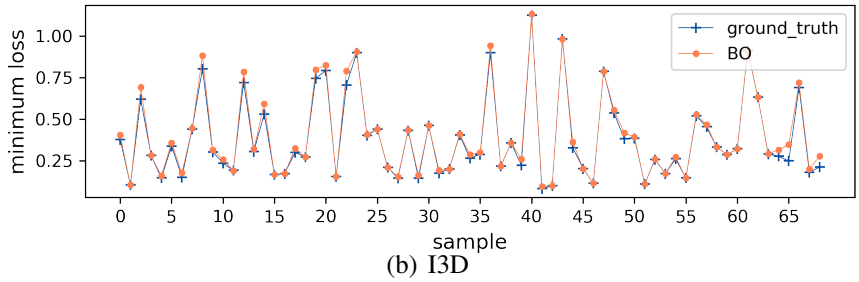
3.3.4 DeepSAVA Attack: Average Absolute Perturbation

Average Absolute Perturbation (AAP) is introduced to measure the perturbation level for each method. As mentioned previously, the sparse Flickering adds a small perturbation per frame, but cannot obtain comparable results to ours. Thus, we choose the pure sparse attack (Sparse) as the main baseline to show the average absolute perturbation. As the baseline is guided by $l_{1,2}$ norm and ours is based on SSIM loss, we will record the average perturbation of $l_{1,2}$ and SSIM separately. To achieve a fair comparison, we set the maximum $l_{1,2}$ norm ball constraint as 0.1 and the maximum SSIM constraint as 0.92. Suppose the fooling rate is f, and distant matrix is D, which can be set to (1-SSIM) or $l_{1,2}$ norm, thus the average absolute perturbation (AAP) can be represented as:

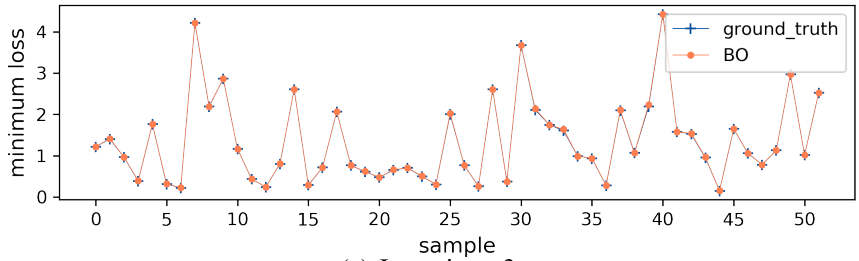
$$AAP(D) = \frac{\sum_N D(V_{adv}, V_{original})}{N} * f + D_{max} * (1 - f),$$



(a) CNN+LSTM



(b) I3D



(c) Inception-v3

Figure 3.7: Minimum loss selected by BO and brute force search along videos

where V_{adv} denotes the generated adversarial video that could successfully mislead the classifier and D_{max} is the maximum constraint; N is the number of adversarial samples achieving a successful attack. We run experiments on 200 randomly selected videos of the UCF101 dataset and record the results of FR, ANI, AAP ($l_{1,2}$), and AAP (SSIM) in Table 3.6. From the results, we can see that for the model I3D and Inception-v3, our method could achieve better performance in terms of efficiency, fooling rate, and AAP (SSIM). For the CNN+LSTM model, our method engages a higher fooling rate and spends less time, and the AAP ($l_{1,2}$) and AAP (SSIM) are also acceptable compared with the baseline model.

3.3.5 DeepSAVA Attack: Visualisation of Results

The generated adversarial frames by DeepSAVA are presented in Figure 3.8. Because of the spatial transformation, the frame looks a little bit shaky but not obvious to human eyes. In fact, in the real world, it is normal to see that there are a few frames with instabilities during

Table 3.7: The results of DeepSAVA (without BO) on UCF101 dataset for different λ values.

Models	λ value	FR	ASP(SSIM)
CNN+LSTM	0.8	57.4%	0.043
	1.0	57.4%	0.0418
	1.5	57.4%	0.0403
I3D	0.8	50.11%	0.0322
	1.0	47.33%	0.0274
	1.5	46.88%	0.0205
Inception-v3	0.8	66.57%	0.0545
	1.0	65.23%	0.0522
	1.5	64.42%	0.0441

video shooting and transmitting. That's why we apply spatial transformation in video attacks to improve the efficiency and fooling rate. In practice, a distortion in one frame of a video is less noticeable than a static image since this specific frame only appears for 0.047 seconds in human eyes [164]. We could also see that it does not lead to a noticeable perturbation as shown by our video demos. We also can find that different attack approaches lead to the same wrong class. This occurs because the second most likely classification is the wrong class, closely following the true class.

When transmitting the videos in the real world, the generated frames need to be compressed into videos first and then decompressed into frames. We found that the additive-only perturbed frames, may not remain adversarial examples after such transmission. Our experiments demonstrate that DeepSAVA can be immune to short video compression due to the fact that perturbation based on spatial transformation can be well preserved during compression while additive perturbation may disappear.

3.3.6 DeepSAVA Attack: Ablation study

We perform ablation experiments to study the effects of combined perturbation for a different number of attacked frames by comparing with additive noise-only and spatial transform-only perturbations, and the effects of BO selection. Table 3.8 shows the FR for three classifiers on the UCF101 dataset. Four approaches are taken to attack the model: 1) only noise ($\mathcal{D}$), 2) only spatial transformation ($\mathcal{S}$), 3) a combination of additive perturbation and spatial transformation ($\mathcal{D} + \mathcal{S}$), and 4) combined perturbation with BO selection. To make more comprehensive evaluations on the superiority of combination, we attack a different number of frames for the CNN+LSTM model as it has the lowest FR when only perturbing one frame. All experiments showed the combination power to increase the FR; using BO selection is

(a) BandMarching (b) Mixing (DeepSAVA) (c) Mixing (Sparse)

(d) BasketballDunk (e) BreastStroke (DeepSAVA) (f) BreastStroke (Sparse)

(g) BasketballDunk (h) Haircut (DeepSAVA) (i) Haircut (Sparse)

Figure 3.8: Original, and adversarial examples generated by DeepSAVA and Sparse [148] when only one frame in the video is perturbed. The red labels are the wrong predictions. The target model for (a)-(b) is CNN+LSTM; for (d)-(f) is Inception-v3; for (g)-(i) is I3D.

Table 3.8: Effects of combining noise ($\mathcal{D}$) and spatial transformation ($\mathcal{S}$) by modifying a different number of frames on UCF101; Fixed the frame(first n-th) , Using BO choose frame, Mask N means that N frames are modified.

Approach	CNN+LSTM			Inception-v3	I3D
	Mask1	Mask2	Mask4	Mask1	Mask1
Fix					
$\mathcal{D}$	$52.77\% \pm 2.24\%$	$71.51\% \pm 1.5\%$	$91.28\% \pm 1.95\%$	$42.45\% \pm 4.30\%$	$10.12\% \pm 1.19\%$
$\mathcal{S}$	$55.27\% \pm 1.82\%$	$74.33\% \pm 1.36\%$	$91.89\% \pm 1.45\%$	$63.91\% \pm 5.61\%$	$29.99\% \pm 2.36\%$
$\mathcal{D}+\mathcal{S}$	$56.22\% \pm 1.65\%$	$77.36\% \pm 1.86\%$	$92.99\% \pm 1.85\%$	$68.86\% \pm 1.83\%$	$47.57\% \pm 2.64\%$
BO					
$\mathcal{D}+\mathcal{S}$	$57.22\% \pm 1.36\%$	$78.95\% \pm 1.93\%$	$93.51\% \pm 1.33\%$	$70.39\% \pm 2.78\%$	$99.89\% \pm 0.11\%$

Table 3.9: Fooling Rate, average selected maximum loss and average time spent for one video of BO Selection and Brute Force Search.

Approach	CNN+LSTM		Inception-v3		I3D		time (s)
	FR	average loss	FR	average loss	FR	average loss	
BO Selection	55.81%	0.21	72.22%	3.39	100%	1.35	16.1
brute force search	55.81%	0.27	72.22%	3.39	100%	1.35	70.4

also useful, especially for the I3D model.

As shown in Table 3.8, using only spatial transformation perturbations results in a higher fooling rate compared to using only additive noise perturbations, highlighting the effectiveness of spatial transformations. The experimental results demonstrate a significant increase in the fooling rate of both the Inception-v3 and I3D models. Furthermore, combining spatial transformation and additive noise perturbations leads to an even higher fooling rate, indicating a stronger attack strategy. Notably, when attacking the I3D model, the combination perturbation increases the fooling rate by approximately four times compared to using only additive perturbation, revealing the significant impact of combination perturbations on I3D models. These findings collectively highlight the ability of combined perturbations to increase the fooling rate, with the additional effectiveness of using BO selection, particularly for the I3D model.

3.3.7 DeepSAVA Attack: The Accuracy of Bayesian Optimisation Selection

To justify whether the Bayesian Optimisation could select the most critical frames, we take the brute force search experiments to obtain the upper bound of the performance: when the selection frame is 1, we select the keyframe manually one by one of the video, and then record the maximum loss found by the search. We randomly select 100 videos from UCF101 in different categories. The fooling rates, average maximum loss, and average time spent for one video on three models are shown in Table 3.9. We also compare the selected maximum loss by BO and brute force search along the video samples in Figure 3.7. The results demonstrate that we can obtain the same results as the brute force search, but spend much less time, which confirms the effectiveness of BO optimisation.

3.3.8 Adversarial Training

In this section, we show the results of two adversarial training methods. The first method is the most intuitive one, as demonstrated in Figure 3.5 (a), which first obtains adversarial examples by performing DeepSAVA and then feeds these adversarial examples into the training stage. Another method is the algorithm demonstrated in Section 2.2.2, which modifies the training loop by injecting PGD adversarial attacks. To evaluate the power of adversarial defences, we randomly picked 200 video samples covering 101 classes from the test dataset of UCF101 and then perform the DeepSAVA or Sparse [148] attack on the first frame of the video to compare the fooling rate. The model recognition accuracy on clean data is also recorded.

For the first method, we present the adversarial training results for the CNN+LSTM model in Table 3.10. We randomly choose 8000 videos from the training dataset. To perform the adversarial training, we randomly generate 1000, 1500, and 3000 adversarial examples by applying the DeepSAVA attack, and feed these adversarial videos with the 8000 unmodified videos as input to the training stage. For the model without defence, we change the adversarial examples to unmodified videos as the input to train the model. As we can see from the results, comparing the defended model with the undefended model, we obtained comparable model recognition accuracy, and a lower fooling rate, which demonstrates that adversarial training can improve the robustness of the model. Additionally, as expected, the results also indicate that the more adversarial examples injected, the lower the fooling rate obtained by the defended model and the larger gap of reduced fooling rate compared with the undefended model. Thus, the more adversarial examples injected into the training stage, the more robust the model is against the DeepSAVA attack. However, generating loads of adversarial examples to train on is extremely expensive in terms of time and resources. Therefore, this encourages us to use a more effective adversarial training algorithm, as demonstrated earlier, to modify the training process by incorporating adversarial attacks.

For the PGD adversarial training, the model recognition accuracy and fooling rate results

are presented in Table 3.11. As the CNN+LSTM model implements the pre-trained CNN model to extract features first and then feeds these features to train the LSTM model, in order to perform defence, we perform adversarial training to train the CNN model first, and then implement the re-trained CNN to extract features. As we can see from the table, after applying the combined perturbation adversarial training, we can obtain lower fooling rates on both DeepSAVA and Sparse attacks. Looking at the results, we find that the more powerful perturbation added in the adversarial training process can lead to a more robust model against both the additive perturbation-only attack and combined perturbation attack, which confirms the effectiveness for a broader range of perturbation when performing adversarial training.

Table 3.10: Model Accuracy and fooling rate on different sizes of training datasets for the CNN +LSTM model.

Models	8000 +1000		8000 +1500		8000 +3000	
	Acc.	FR	Acc.	FR	Acc.	FR
CNN+LSTM With Defense	**59.77%**	**64.42%**	62.06%	**59.26%**	63.42%	**58.92%**
CNN+LSTM without Defense	59.19%	66.99%	**63.79%**	63.06%	**64.36%**	63.06%
defended vs. undefended	+0.98%	-3.8%	-2.7%	-6.0%	-1.46%	-6.6%

Table 3.11: Model Accuracy and fooling rate on different models with defence and without defence.

Defence	Model	Accuracy	DeepSAVA	Sparse
Combined Defence	inception-v3	63.96%	**37.62%**	**25.29%**
PGD Defence [90]	inception-v3	64.52%	38.53%	27.21%
Without Defence	inception-v3	**65.12%**	40.59%	29.6%
Combined Defence	CNN+LSTM	68.02%	**47.86%**	**46.15%**
PGD Defence [90]	CNN+LSTM	69.00%	52.89%	52.06%
Without Defence	CNN+LSTM	**70.22%**	55.93%	55.26%
Combined Defence	I3D	87.5%	**77.3%**	**42.3%**
PGD Defence [90]	I3D	88.2%	78.5%	46.2%
Without Defence	I3D	**89.1%**	80.2%	47.2%

Table 3.12: Fooling Rate across recurrent models on UCF101 dataset.

Models	Approach	LSTM	Vanilla RNN	GRU	Inception-v3	I3D
LSTM	Baseline	100%	34.42%	64.35%	50.0%	53.48 %
	DeepSAVA	100%	41.38%	85.34%	52.17%	54.62%
Vanilla RNN	Baseline	100%	100%	100%	71.74%	60.50 %
	DeepSAVA	100%	100%	100%	82.40%	64.02%
GRU	Baseline	79.34 %	40.70%	100%	50.0%	42.68%
	DeepSAVA	84.75%	56.03%	100%	51.08%	49.58%
Inception-v3	Baseline	22.95 %	22.80%	22.90%	100%	33.61%
	DeepSAVA	24.36%	26.72%	31.03%	100%	37.8%
I3D	Baseline	6.56 %	7.01%	7.64%	13.04%	100%
	DeepSAVA	10.08%	9.48%	8.62%	14.13%	100%

3.3.9 Transferability Across Models

The transferability across models is an important evaluation of adversarial attacks, which can be treated as a black-box problem without accessing the parameters of the target model. In our work, the I3D and Inception-v3 only contain CNN, while the recurrent neural networks (RNN) such as CNN+LSTM contain the time-related network. Due to the unique time-related structure of videos, I conduct extensive experiments to evaluate the transferability across various time-related networks. I perform the transferability experiments on the UCF101 dataset for different RNNs, Inception-v3, and I3D models. For the recurrent models, the features of original videos are extracted firstly by CNN (Inception-v3) model and then are fed into vanilla RNN [115], LSTM [48], and GRU [21] networks respectively. The training accuracy for vanilla RNN and GRU are 65.16% and 73.05% respectively.

As Figure 3.6 shows that the Sparse [148] performs better than the Sparse Flickering in terms of FR, I choose the Sparse [148] as the baseline method. The fooling rates (FR) of the generated videos across models are presented in Table 3.12. The models in rows are used to generate adversarial videos, and the models in columns are the target attack classifiers. Here we disturb seven frames of a video to enlarge the attacking success rate. I use the adversarial examples generated from the white-box attack for the transferability, which leads to the FR in the diagonal being 100%. These adversarial examples are then used to attack other models (like a black-box attack) as detailed in Table 3.12. Compared with the baseline, our approach has a higher FR which indicates better performance in terms of transferability. The difference between vanilla RNN and the other models is that vanilla RNN has no memory component, so it shows a weak performance on the video classification task. As we observed, adversarial videos generated from LSTM and GRU models can fool

the vanilla RNN successfully. Additionally, the FR across GRU and LSTM are around 85%, which shows good transferability between the recurrent models with memory. However, from the fooling rates shown in Table 3.12, we could see that the transferability from RNNs to CNNs is not as good as that from CNNs to RNNs. The fooling rates to attack RNN models are higher than those to attack the I3D and Inception-v3 models. While that happens maybe because the I3D model has the highest training accuracy, it engages the lowest fooling rate when performing the black-box attack on it. As we can also conclude that due to the lowest training accuracy the Vanilla RNN model obtains, it achieves the highest fooling rate on attacking the unseen Vanilla RNN model. Overall, compared with the Sparse baseline, our method could achieve better transferability.

3.4 Case Study I

Next-Frame Video Prediction

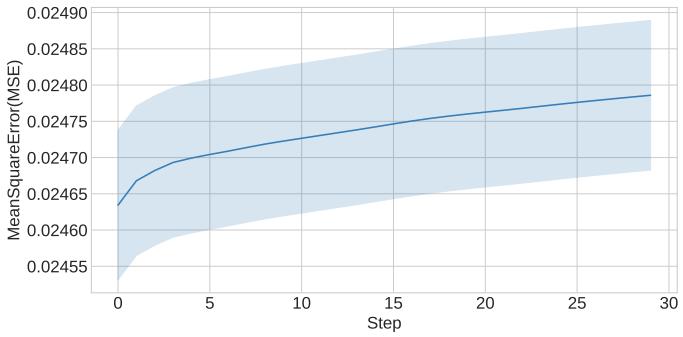
In this section, we explore our framework in the application of next-frame video prediction. Next-frame video prediction aims to predict the next frame based on a sequence of previous videos. Various models have been applied for the video prediction tasks, i.e., CNN and LSTM [59], PredNet [88], and Transformer [74]. As a case study, we will apply our DeepSAVA framework to the video prediction model based on CNN and LSTM models to see whether the sparse attack can also be effective on the frame prediction task.

In the CNN and LSTM architecture, the CNN acts as an encoder that extracts spatial features from the input video sequences, while the LSTM decodes the temporal connections between the frame and video sequence to forecast the next frame. Next-frame video prediction based on combining CNN and LSTM networks offers a wide range of applications in computer vision, including video reduction, editing, and creation, and also demonstrated significant potential to predict the next-frame of videos. In this chapter, we examined this model using the moving-MNIST dataset, which is a common benchmark dataset for videos.

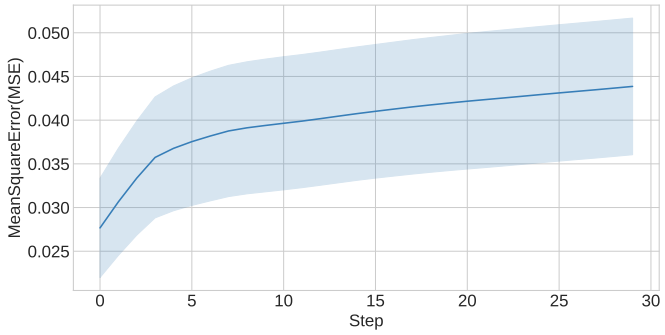
In next-frame prediction, the model inputs a sequence of the previous frame, f_n, to predict a new frame, $f_{(n+1)}$. Therefore, it takes a sequence of input frames (x_n) as input, to output the prediction frame $y_{(n+1)}$. To be noticed, in the video prediction task, the accuracy is measured by the Mean Square Error or SSIM similarity metric.

3.4.1 Problem Definition:

The video prediction model is defined as $J(\cdot;\boldsymbol{\theta})$ with pre-trained weights $\boldsymbol{\theta}$. The input clean video is defined as $\mathbf{X} = (x_1, x_2, ..., x_T)$ and the perturbed example is denoted as $\hat{\mathbf{X}} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_T)$. The video sequence containing the next frame of the input video is $y = (x_2, x_3, ..., x_{T+1}) \in \mathbb{R}^{T\times W\times H\times C}$, which is used as the ground truth label for the prediction output. Therefore, the objective function for perturbing the task of frame prediction can be



(a) First n-th frames



(b) With BO selection

Figure 3.9: Per-step Mean Square Error (MSE) between the predicted frame and ground truth frame, while introducing perturbations to the input frames. The amount of perturbation added to the input frames is constrained to a budget within the l_2=0.03 norm ball.

defined as:

$$\arg\min \left(\lambda \ell_{similar}(\hat{\mathbf{X}}, \mathbf{X}) - \ell_{adv} \left(y, J(\hat{\mathbf{X}}; \boldsymbol{\theta}) \right) \right). \tag{3.6}$$

Here the loss function l_{adv} we used is the mean square error. As a result, the attack goal is to enlarge the distance between prediction frames and the ground truth video, while maintaining the perturbation human imperceptible.

3.4.2 Experiments

We perform the DeepSAVA on the next-frame video prediction model. We randomly picked 100 samples of the moving-MNIST dataset and plotted the MeanSquareError (MSE) values against the steps under DeepSAVA in Figure 3.9. In Figure 3.9 (a), we conducted attacks on the first frame of the input video, measuring their respective MSEs. Subsequently, we apply the BO selection to identify the most crucial frame, and the results are illustrated in

Figure 3.9 (b). Notably, the BO selection consistently designated the last frame as the most critical. Furthermore, the results indicate that attacking the last frame chosen by BO resulted in significantly higher MSE values. Consequently, it can be inferred that the next-frame video prediction model relies heavily on the information contained in the last frame of the input video.

3.5 Case Study II

Evaluate the robust model using an explainable AI method

To evaluate the regular model and robust mode, we use the salience map to show what features are learned by each model. The salience map can effectively highlight the pixels within each frame that have the greatest impact on the model's predictions. It serves as a valuable tool in the field of explainable AI, commonly used to evaluate input images or videos in a neural network, revealing which region of the image or frame contributes the most to the model's decision-making process.

A gradient-based salience map is designed to visualize the gradients of the predicted outcome from the model with respect to the input pixel values [125], [130], [174]. The relative contribution of each pixel to the final prediction of the model can be calculated by applying tiny tweaks to pixel values across the image and catching the change in the predicted class.

In this section, we perform experiments to evaluate the performance of the Inception-V3 model with defence and without defence. In Figure 3.10, we display the salience map superimposed on the input frame. The background image represents the original video of a girl playing the flute, and the yellow pixels indicate areas with high gradients. We compare the salience maps learned by the regular model and the robust model, which was trained using the combination.

From the results, we observe that both models heavily rely on the central portion of the frame, particularly focusing on the flute, when making predictions. Additionally, numerous frames do not exhibit a salience map, suggesting that they do not contribute significantly to the model's final decision. However, when conducting an attack on a single frame of the video, it is observed that perturbing a frame without a salience map is more effective to result in a change in the model's predicted label compared to perturbing a frame highlighted by the salience map (highlighted in yellow). This finding raises important considerations, which we will discuss later. Furthermore, based on the experimental results, a notable difference between the regular model and the robust model is that there are more frames highlighted by the salience map. This observation suggests that the model with the defence method may be more robust due to its ability to learn more distinctive features at the individual frame level.

3.5.1 Discussion and Limitations

The experiment was taken by a Python package 'Kera-vis', which primarily focuses on generating salience maps for individual images [69]. While this approach may not capture the full temporal dynamics and interactions between frames, it can provide some useful information if we set the input to each individual frame of video. Visualizing the salience map for each frame allows us to analyze the frame-level attention and identify the regions that the model focuses on for making predictions.

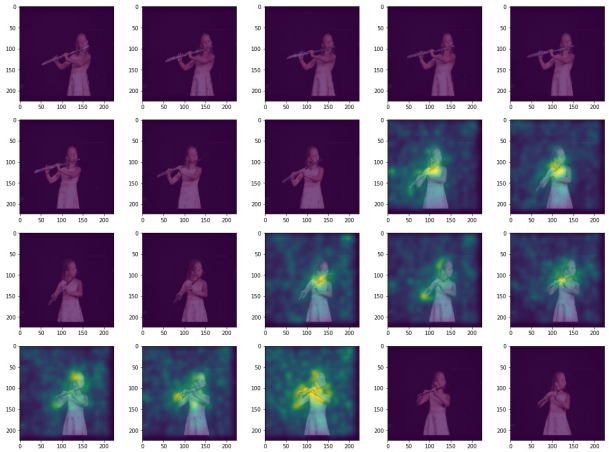
However, this approach is still limited for video model analysis. Since each frame is considered independently, the salience maps cannot find the temporal relationships present in the video. Some frames may not have prominent salience maps if they are less informative or contribute less to the model's predictions. Additionally, the salience maps might not provide a holistic view of the salient regions in the entire video.

Hence, to gain a more comprehensive understanding of the salience map and its temporal dependencies in the video, considering video-specific salience map generation techniques [31], [32], [168] would be more appropriate, which can be considered a future work. These techniques are designed to capture the dynamics and interactions across frames, providing a more accurate representation of the salient regions in the video.

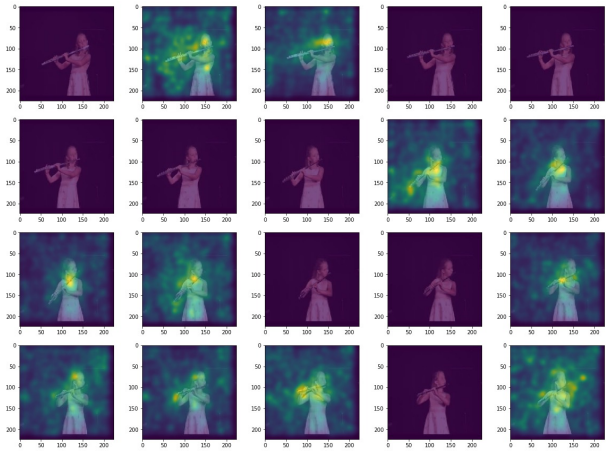
3.6 Discussion and Conclusion

In this chapter, we apply spatial transformed perturbation and additive noise to attack as few frames as possible to obtain sparse adversarial videos. The most influential frames to be attacked are selected by a joint optimisation strategy with Bayesian optimisation (BO) and SGD-based optimisation. We take sufficient experiments to examine the power of BO and show the effectiveness of BO selection in this task. Additionally, the quality of generated adversarial examples is measured by SSIM instead of l_p-norm, which can capture both additive noise and spatial transformation effectively. We propose the novel and effective video attack mechanism, DeepSAVA, and perform extensive experiments to evaluate its performance on the UCF101 and HMDB51 action dataset and three different classification models: Inception-3v, CNN+LSTM, and I3D. We obtain better results than state-of-the-art sparse baselines in terms of both fooling rate and transferability, which confirms the success of DeepSAVA. Our most significant results are for the I3D model, by only attacking one frame of the video to obtain a 99.5% to 100% attack success rate.

Additionally, in this chapter, we add adversarial training experiments to improve the robustness of video classification models. By now, adversarial training research focused on image classification models, thus, in this chapter, we choose to adopt the most effective defence method, PGD adversarial training, to retrain the video classifiers. We modify the adversarial training algorithm by adding a combination of spatial transformation and additive perturbation in light of our DeepSAVA framework. We also show the experimental



(a) Regular Model



(b) Robust Model

Figure 3.10: Saliency map overlaid on each frame. The regular model is the model that is not trained with adversarial examples, and the robust model is trained by combined perturbation.

results of the most intuitive adversarial training approach, which takes the clean training dataset and the generated adversarial examples as input to re-train the model. As a result, after applying our adversarial training with combined perturbation, we can obtain a more robust model compared to the PGD adversarial training, and more effective than injecting adversarial examples. Nonetheless, the use of BO for selecting critical frames is still a time-intensive process. In the future, we will explore the possibility of implementing the Gumbel-softmax technique, an enhanced and differentiable version of SemHash, to pinpoint significant features within input videos.

Chapter 4

Verification for 3D Point Cloud Models

"Large-scale pre-trained language models are a revolutionary development that has had a huge impact in many areas. They have made significant strides in machine translation, question-answering systems, natural language understanding, text-to-speech synthesis, and more, which will fundamentally change the way people interact with computers in the near future."
(Yann LeCun)

4.1 Introduction

Recent years have witnessed increasing interest in 3D object detection, and Deep Neural Networks (DNNs) have also demonstrated their remarkable performance in this area [109], [110]. For 3D object detectors, point clouds are used to represent 3D objects, which are usually the raw data obtained from LIDARs and depth cameras. Such 3D deep learning models have been widely used in multiple safety-critical applications such as motion planning [143], virtual reality [131], and autonomous driving [17], [82]. However, extensive research has shown that DNNs are vulnerable to adversarial attacks, appearing as adding a small amount of non-random, ideally human-invisible, perturbations on the input will cause DNNs to make abominable predictions [13], [57], [97], [136], [144], [166]. Therefore, there is an urgent need to address such safety concerns in DNNs caused by adversarial examples, especially in safety-critical 3D object detection scenarios.

Recently, most works on analysing the robustness of 3D models mainly focus on adversarial attacks with an aim to reveal the model's vulnerabilities under different types of perturbations. Xiang, Qi, and Li [162] claimed that they are the first to perform extensive adversarial attacks on 3D point cloud models by perturbing the positions of points or generating new points. Recent work extended adversarial attacks for images to 3D point clouds by perturbing points and generating new points [41], [79], [170]. Additionally, Cao, Xiao, Yang, *et al.* [12] proposed adversarial attacks on LiDAR systems.Wicker and Kwiatkowska [154] used occlusion attack and Zhao, Wu, Chen, *et al.* [182] proposed isometric transformations attack. Towards these adversarial attacks, corresponding defence techniques are developed [170], [184], which are more effective than adversarial training such as [85], [181].

However, as Tramer, Carlini, Brendel, *et al.* [139] and Athalye, Carlini, and Wagner [2] indicated, *even though these defenses are effective for some attacks, they can still be broken by other stronger attacks*. Thereby, we need a more solid solution, ideally with *provable guarantees*, to verify whether the model is robust to *any* adversarial attacks within an allowed perturbation budget[1]. This technique is also generally regarded as verification on (local) adversarial robustness[2] in the community. So far, various solutions have been proposed to tackle robustness verification, but they mostly focus on the image domain [7], [54], [126], [138]. In contrast, verification of the adversarial robustness of 3D point cloud models is barely explored by the community. As far as we know, 3DCertify, proposed by Lorenz, Ruoss, Balunović, *et al.* [87] is the first, and also the *only* work to verify the robustness of 3D models. Although 3DCertify is very inspiring, it has not yet completely resolved some critical challenges in robustness verification for 3D models, according to our empirical study.

Below we address three main challenges to be tackled in this chapter:

[1] In the community, we normally use a small predefined l_p-norm ball to quantify such perturbations, namely, within this small perturbing space the decision should remain the same from a perspective of a human observer.

[2] For convenience, we use *robustness verification* or *verification* for short in this chapter.

- *Existing 3D point clouds are time-consuming and need large memory.* So far 3DCertify [87] is the first and only verification tool for 3D models, the empirical experiments indicate that it is time-consuming and thus not computationally attainable on large neural networks. As 3DCertify is built on DeepPoly [126] when directly applying the relaxation algorithm that is specifically designed for images on high-dimensional point clouds, it will result in out-of-memory issues and cause the termination of the verification.

- *There is no tool to verify the JANet in the PointNet.* 3DCertify can only verify a simplified PointNet model without the Joint Alignment Network (JANet) consisting of matrix multiplication operations. Since the learned representations are expected to be invariant to spatial transformations, JANet is the key enabler in PointNet to achieve this geometric invariant functionality by adopting the T-Net and matrix multiplications. Recent research also shows that JANet is essential to improve PointNet performance [110] and therefore widely applied in some safety-critical tasks [1], [18], [102].

- *A 3D point clouds verifier for different l_p-norm metrics needs.* Existing verifier can only work on l_∞-norm metric, however, arguably, some researchers in the community regard other l_p-norm metrics such as l_1, l_2-norm metrics are equally (if not more) important in the study of adversarial robustness [7], [151]. Thus, a robustness verification tool that can work on a wide range of l_p-norm metrics is also worthy of a comprehensive exploration.

This motivates us to design a more efficient and general framework to verify various architectures of point cloud models. The key challenges in verifying the large-scale complete PointNet models are addressed as dealing with the cross-non-linearity operations in the multiplication layers and the high computational complexity of high-dimensional point cloud inputs and added layers. Thus, we propose an efficient verification framework, 3DVerifier, to tackle both challenges by adopting a linear relaxation function to bound the multiplication layer and combining forward and backward propagation to compute the certified bounds of the outputs of the point cloud models.

In this chapter, we begin by presenting a detailed overview of the general framework for the proposed method, and then provide a comprehensive and informative running example to help the reader understand the approach, which aims to provide a clear and concise explanation of the computation involved in the certification process. Furthermore, we present extensive experimental results in the experiments section to demonstrate the effectiveness and robustness of our certification method and provide a thorough analysis of the data obtained.

4.2 Methodology: 3DVerifier

4.2.1 Overview

The clean input point cloud $\mathbf{P}_0$ with n points can be defined as

$$\mathbf{P}_0 = \{\mathbf{p_0}^{(i)} \mid \mathbf{p_0}^{(i)} \in \mathbb{R}^3, i = 1, ..., n\},$$

where each point $\mathbf{p_0}^{(i)}$ is represented in a 3D space coordinate (x, y, z). To verify the robustness of C, our objective is to confirm that the predicted label remains unchanged within a defined perturbation budget. Consequently, we select the point clouds that are correctly recognised by C as input of the verifier.

Throughout this chapter, we will perturb the points by shifting their positions in the 3D space within a distance bounded by the l_p-norm ball. Given the perturbed input,

$$\mathbf{P} = \{\mathbf{p}^{(i)} \mid \mathbf{p}^{(i)} \in \mathbb{R}^3, i = 1, ..., n\},$$

that is in the l_p-norm ball

$$\mathbb{S}_p\left(\mathbf{p}_0^{(i)}, \epsilon\right) := \left\{\mathbf{p}^{(i)} \mid \left\|\mathbf{p}^{(i)} - \mathbf{p}_0^{(i)}\right\|_p \leq \epsilon, i = 1, ..., n\right\}.$$

we aim to verify whether the predicted label of the model is stable within the region $\mathbb{S}_p$. This can be solved by finding the ***minimum adversarial perturbation*** ϵ_{min} via binary search, such that

$$\exists \mathbf{P} \in \mathbb{S}_p\left(\mathbf{P}_0, \epsilon_{min}\right), \operatorname{argmax} C(\mathbf{P}) \neq c,$$

where $c = \operatorname{argmax} C(\mathbf{P_0})$. Such ϵ_{min} is also referred to as the *untargeted* robustness. As for the *targeted* robustness, it can be interpreted as the prediction output score for the true class being always higher than that for the target class.

Assuming that the target class is t, the objective function is:

$$\begin{aligned} &\min\left\{y_c(\mathbf{P}) - y_t(\mathbf{P})\right\} := \sigma_\epsilon, \\ &s.t \quad \|\mathbf{p}^k - \mathbf{p}_0^k\|_p \leq \epsilon, (k = 1, 2, ..., n), \end{aligned} \tag{4.1}$$

where y_c represents the logit output for class c and y_t is for the target class t. $\mathbf{P}$ is the set of points that is centred around the original set of points $\mathbf{P}_0$ within the ball of the norm l_p with radius ϵ. Thus, if $\sigma_\epsilon > 0$, the logit output of the true class c is always greater than the target class, which means that the predicted label cannot be t. Due to the fact that finding the exact output of σ_ϵ is an NP-hard problem [63], the objective function of our work can be alternatively altered as computing the lower bound of σ_ϵ. By applying binary search to update the perturbation ϵ, we can find the minimum adversarial perturbation. Equivalently, the *maximum* ϵ_{cert} that does not alter the predicted label can be attained. Thus, in this chapter, we aim to compute the ***certified lower bound*** of $\sigma_{\epsilon_{cert}}$.

4.2.2 Generic Framework

To obtain the lower bound of σ_ϵ for the l_p-norm ball bounded model, we propagate the bound of each neuron layer by layer. As mentioned previously, most of the structures employed by the 3D point cloud models are similar to the traditional image classifiers, such as the convolution layer, batch normalisation layer, and pooling layer. The most distinctive structure of the point clouds classifier, like the PointNet [110], is the JANet structure. Thus, to compute the logit outputs of the neural network, our verification algorithm adopt three types of formulas to handle different operations: 1) linear operations (e.g., convolution, batch normalization, and average pooling), 2) non-linear operation (e.g., ReLU activation function and max pooling), 3) multiplication operation.

Let $\Phi^l(\mathbf{P})$ be the output of neurons in the layer l with point clouds input $\mathbf{P}$. The input layer can be represented as $\Phi^0(\mathbf{P}) = \mathbf{P}$. Suppose that the total number of layers in the classifier is m, $\Phi^m(\mathbf{P})$ is defined as the output of the neural network. In order to verify the classifier, we aim to derive a linear function to obtain the global upper bound u and lower bound l for each layer output $\Phi^l(\mathbf{P})$ for $\mathbf{P} \in \mathbb{S}_p(\mathbf{P}_0, \epsilon)$.

The network takes input in the form of (x, y), where "x" represents the number of points, and "y" signifies the dimensionality of each point, which is set at 3. The bounds are derived layer by layer from the first layer to the final layer. We have full access to all parameters of the classifier, such as weights $\mathbf{W}$ and bias $\mathbf{b}$. To calculate the output for each neuron, we show below the linear functions to obtain the bounds of the l-th layer for the neuron in position (x, y) based on the previous l'-th layer:

$$\sum_{i,j} \mathbf{A}^{(l,l'),L}_{(x,y,i,j)} \Phi^{l'}_{(x+i,j)}(\mathbf{P}) + \mathbf{B}^{(l,l'),L}_{(x,y)} \leq \Phi^l_{(x,y)}(\mathbf{P}) \leq \sum_{i,j} \mathbf{A}^{(l,l'),U}_{(x,y,i,j)} \Phi^{l'}_{(x+i,j)}(\mathbf{P}) + \mathbf{B}^{(l,l'),U}_{(x,y)}, \quad (4.2)$$

where $\mathbf{A}^L, \mathbf{B}^L, \mathbf{A}^U, \mathbf{B}^U$ are weight and bias matrix parameters of the linear function for lower and upper bound calculations respectively. $\mathbf{A}$ and $\mathbf{B}$ are initially assigned as identity matrix ($\mathbf{I}$) and zeros matrix ($\mathbf{0}$), respectively, to keep the same output of $\Phi^l_{(x,y)}$ when $l = l'$. To calculate the bounds of the current layer, we take backward propagation to previous layers. The $\Phi^{l'}_{(x+i,j)}$ is substituted by the linear function of the previous layer recursively until it reaches the first layer ($l' = 0$). After that, the output of each layer can be formed by a linear function of the first layer ($\Phi^0(\mathbf{P}) = \mathbf{P}$), as:

$$\mathbf{A}^{(l,0),L} * \mathbf{P} + \mathbf{B}^{(l,0),L} \leq \Phi^l(\mathbf{P}) \leq \mathbf{A}^{(l,0),U} * \mathbf{P} + \mathbf{B}^{(l,0),U}. \quad (4.3)$$

Since the perturbation added to the point clouds input is bounded by the l_p-norm ball, $\mathbf{p} \in \mathbb{S}_p(\mathbf{p}_0, \epsilon)$, to compute the global bounds we need to minimise the lower bound and maximise the upper bound in Equation 4.3 over the input region. Thereby, the linear function to compute the global bounds for the l-th layer can be represented as:

$$\Phi^{l,U/L}_{x,y} = \pm\epsilon \|\mathbf{A}^{(l,0),U/L}_{(x,y,:,:)}\|_q + \sum_{i,j} \mathbf{A}^{(l,0),U/L}_{(x,y,i,j)} \mathbf{p}_0^{(x+i,j)} + \mathbf{B}^{(l,0),U/L}_{(x,y)}, \quad (4.4)$$

where $\|\mathbf{A}\|_q$ is the l_q-norm of $\mathbf{A}$ and $1/p + 1/q = 1$ with $p, q > 1$, "U/L" denotes that the equations are formulated for the upper bounds and lower bounds, respectively and "+" for "U", "-" for "L". This generic framework resembles CROWN [178], which is widely utilised in the verification works to verify feed-forward neural networks (e.g. CNN-Cert [7], Transformer Verification [123]). Unlike existing frameworks based on CROWN, we further extend the algorithm to verify point cloud classifiers.

4.2.3 Functions for linear and non-linear operation

As the linear and nonlinear functions are basic operations in the neural network, we first adapt the framework given in [7] to 3D point cloud models. In Section 4.2.4, we will present our novel technique for JANet.

Functions for linear operation

Suppose that the output of the l'-th layer, $\Phi^{l'}(\mathbf{P})$, can be computed by the output of the (l'-1)-th layer, $\Phi^{l'-1}(\mathbf{P})$, by the linear function

$$\Phi^{l'}(\mathbf{P}) = \mathbf{W}^{l'} * \Phi^{l'-1}(\mathbf{P}) + \mathbf{b}^{l'},$$

where $\mathbf{W}^{l'}$ and $\mathbf{b}^{l'}$ are parameters of the function in the layer l'. Thus the Equation 4.2 can be propagated from the layer l to the layer $l' - 1$ by substituting $\Phi^{l'}(\mathbf{P})$.

$$\begin{aligned}
&\Phi^{l,U/L}_{(x,y)}(\mathbf{P}) = \sum_{i,j} \mathbf{A}^{(l,l'),U/L}_{(x,y,i,j)} \Phi^{l'}_{(x+i,j)}(\mathbf{P}) + \mathbf{B}^{(l,l'),U/L}_{(x,y)} \\
&= \sum_{i,j} \mathbf{A}^{(l,l'),U/L}_{(x,y,i,j)} (\sum_{k} \mathbf{W}^{l'}_{(k,j)} \Phi^{l'-1}_{(x+i,k)}(\mathbf{P}) + \mathbf{b}^{l'}_{x+i,j}) + \mathbf{B}^{(l,l'),U/L}_{(x,y)} \\
&= \sum_{i,k} \mathbf{A}^{(l,l'-1),U/L}_{(x,y,i,k)} \Phi^{l'-1}_{(x+i,k)}(\mathbf{P}) + \mathbf{B}^{(l,l'-1),U/L}_{(x,y)}
\end{aligned} \tag{4.5}$$

Functions for basic non-linear operation

For the l'-th layer with non-linear operations, we apply two linear functions to bound $\Phi^{(l')}(\mathbf{P})$:

$$\alpha^{l',L}\Phi^{l'-1}(\mathbf{P}) + \beta^{l',L} \leq \Phi^{(l')}(\mathbf{P}) \leq \alpha^{l',U}\Phi^{l'-1}(\mathbf{P}) + \beta^{l',U}.$$

Given the bounds of $\Phi^{(l'-1)}(\mathbf{P})$, the corresponding parameter $\alpha^L, \alpha^U, \beta^L, \beta^U$ can be chosen appropriately. Suppose that the non-linear function is $f(y)$, where y is the output from previous layer $\Phi(\mathbf{P})^{l'-1}$. The bound obtained from the previous layer is $[l, u]$. Therefore, our goal is to bound $f(y)$ by the following constraints:

$$\alpha^{l',L}y + \beta^{l',L} \leq f(y) \leq \alpha^{l',U}y + \beta^{l',U},$$

where parameters $\alpha^{l',L}$, $\beta^{l',L}$, $\alpha^{l',U}$, and $\beta^{l',U}$ are chosen depending on the lower and upper bound, l and u, of the previous layer, which follows different rules for different functions.

The way to bound the non-linear functions has been thoroughly discussed in previous works on images (e.g., [7], [126], [178]). Thus, by reviewing their methods, we synthesise our relaxations for the nonlinear functions to determine the parameters according to l and u.

The most common activation function is the ReLU function, which is represented as $f(y) = max(0, y)$. Thus, if $u \leq 0$, the output of $f(y)$ is 0; if $l \geq 0$, the output will be exactly y. As for the situation that $l < 0$ and $u > 0$, we set the upper bound u to be the line cross two endpoints: $(l, 0)$ and $(u, f(y))$, which can be represented as $f^U(y) = \frac{u(y-l)}{(u-l)}$. As for the expression for the lower bound, to make the bounds tighter, we consider two cases: if $u > |l|$, $f^L(y) = y$ and otherwise $f^L(y) = y$. Therefore, we can choose the $\alpha^U = \frac{u}{(u-l)}, \beta^U = \frac{-ul}{(u-l)}, \beta^L = 0$; $\alpha^L = 0$ when $u < |l|$, and otherwise $\alpha^L = 1$.

After computing the corresponding $\alpha^L, \alpha^U, \beta^L, \beta^U$, we back propagate the Equation 4.2 to the (l'-1)-th layer:

$$\begin{aligned}
&\Phi^{l,U/L}_{(x,y)}(\mathbf{P}) = \sum_{i,j} \mathbf{A}^{(l,l'),U/L}_{(x,y,i,j)} \Phi^{l'}_{(x+i,j)}(\mathbf{P}) + \mathbf{B}^{(l,l'),U/L}_{(x,y)} \\
&= \sum_{i,j} \mathbf{A}^{(l,l'),U/L,+}_{(x,y,i,j)} * (\alpha^{l',U/L}_{(x+i,j)} \Phi^{l'-1}(\mathbf{P})_{(x+i,j)} + \beta^{l',U/L}_{(x+i,j)}) \\
&+ \sum_{i,j} \mathbf{A}^{(l,l'),U/L,-}_{(x,y,i,j)} * (\alpha^{l',L/U}_{(x+i,j)} \Phi^{l'-1}(\mathbf{P})_{(x+i,j)} + \beta^{l',L/U}_{(x+i,j)}) \\
&= \sum_{i,j} \mathbf{A}^{(l,l'-1),U/L}_{(x,y,i,j)} \Phi^{l'}_{(x+i,j)}(\mathbf{P}) + \mathbf{B}^{(l,l'-1),U/L}_{(x,y)},
\end{aligned}$$

where if $\mathbf{A}^{(l,l'),U/L}_{(x,y,i,j)}$ is a positive element of $\mathbf{A}^{(l,l'),U/L}_{(:,:,:,j)}$, then $\mathbf{A}^{(l,l'),U/L,+}_{(x,y,i,j)} = \mathbf{A}^{(l,l'),U/L}_{(x,y,i,j)}$ and $\mathbf{A}^{(l,l'),U/L,-}_{(x,y,i,j)} = 0$; otherwise, $\mathbf{A}^{(l,l'),U/L,-}_{(x,y,i,j)} = \mathbf{A}^{(l,l'),U/L}_{(x,y,i,j)}$ and $\mathbf{A}^{(l,l'),U/L,+}_{(x,y,i,j)} = 0$.

4.2.4 Functions for Multiplication Layer

The most critical structure in the PointNet model is the JANet, which contains the multiplication layers. For the multiplication, assume that it takes the output of the previous layer ($\Phi^{l'-1}(\mathbf{P})$) and (l'-r)-th layer ($\Phi^{l'-r}(\mathbf{P})$, r $\in [1, l']$) as inputs, the output of $\Phi^{l'}(\mathbf{P})$ can be calculated by:

$$\Phi^{l'}_{(x,y)}(\mathbf{P}) = \sum_{k=1}^{d_k} \Phi^{l'-r}_{(x,k)}(\mathbf{P}) * \Phi^{l'-1}_{(k,y)}(\mathbf{P}),$$

where d_k is the dimension of $\Phi^{l'-r}_{(x,:)}(\mathbf{P})$ and $\Phi^{l'-1}_{(:,y)}(\mathbf{P})$. To get the linear relaxation for the multiplication operation, we will adapt the McCormick envelope [64].

In the JANet, before the multiplication layer, there is one reshape layer and one pooling layer. To simplify the calculation, we choose $\Phi^{l'-1}$ as the output of the pooling layer, using $h = d_k * (k-1) + y$ to represent the transformation in the reshape layer. The equation to

compute $\Phi^{l'-1}(\mathbf{P})$ can be rewritten as:

$$\Phi^{l'}_{(x,y)}(\mathbf{P}) = \sum_{k=1}^{d_k} \Phi^{l'-r}_{(x,k)}(\mathbf{P}) * \Phi^{l'-1}_{(0,h)}(\mathbf{P}),$$

where $h = d_k * (k-1) + y$.

To obtain the bounds of the multiplication layer, we utilise two linear functions of the input $\mathbf{P}$ to bound $\Phi^{l'}(\mathbf{P})$:

$$\mathbf{\Lambda}^{(l',0),L}_{(x,y,i,:)}\mathbf{p}^{(x+i)} + \Theta^{(l',0),L}_{(x,y)} \leq \Phi^{l'}_{(x,y)}(\mathbf{P}) \leq \mathbf{\Lambda}^{(l',0),U}_{(x,y,i,:)}\mathbf{p}^{(x+i)} + \Theta^{(l',0),U}_{(x,y)}, \tag{4.6}$$

where λ and Θ are new parameters of linear functions to bound the multiplication.

Theorem 2. *(McCormick envelope [64])Let l_r and u_r be the lower and upper bounds of the (l'-r)-th layer output ($\Phi^{l'-r}(\mathbf{P})$, $r \in [1, l']$); l_1 and u_1 be the lower and upper bounds of the (l'-1)-th layer output ($\Phi^{l'-1}(\mathbf{P})$). Suppose $a^L = l_1, a^U = u_1, b^L = b^U = l_r, c^L = -l_r * l_1$, and $c^U = -l_r * u_1$, Then, for any point clouds input $\mathbf{P} \in \mathbb{S}_p(\mathbf{P}_0, \epsilon)$:*

$$a^L * \Phi^{l'-r}_{(x,k)}(\mathbf{P}) + b^L * \Phi^{l'-1}(\mathbf{P}) + c^L \leq \Phi^{l'-r}_{(x,k)}(\mathbf{P}) * \Phi^{l'-1}_{(k,y)}(\mathbf{P}) \leq a^U * \Phi^{l'-r}_{(x,k)}(\mathbf{P}) + b^U * \Phi^{l'-1}_{(0,h)}(\mathbf{P}) + c^U,$$

The bounds and corresponding bounds matrix of $\Phi^{l'-r}(\mathbf{P})$ and $\Phi^{l'-1}(\mathbf{P})$ can be calculated via back propagation. Given Theorem 2, the functions can be formed to compute the $\mathbf{\Lambda}^{(l',0),U/L}$ and $\Theta^{(l',0),U/L}$ for $\Phi^{l'}_{(x,y)}(\mathbf{P})$ in Equation 4.6 as:

$$\mathbf{\Lambda}^{(l',0),U/L}_{(x,y,i,:)} = \sum_k (a^{U/L,+}_{(0,h)}\mathbf{A}^{(l'-r,0),U/L}_{(x,k,i,:)} + a^{U/L,-}_{(0,h)}\mathbf{A}^{(l'-r,0),L/U}_{(x,k,i,:)}) +$$

$$\sum_k (b^{U/L,+}_{(x,k)}\mathbf{A}^{(l'-1,0),U/L}_{(0,h,i,:)} + b^{U/L,-}_{(x,k)}\mathbf{A}^{(l'-1,0),L/U}_{(0,h,i,:)}),$$

$$\Theta^{(l',0),U/L}_{(x,y)} = \sum_k (a^{U/L,+}_{(0,h)}\mathbf{B}^{(l'-r,0),U/L}_{(x,k)} + a^{U/L,-}_{(0,h)}\mathbf{B}^{(l'-r,0),L/U}_{(x,k)}) +$$

$$\sum_k (b^{U/L,+}_{(x,k)}\mathbf{B}^{(l'-1,0),U/L}_{(k,y)} + b^{U/L,-}_{(x,k)}\mathbf{B}^{(l'-1,0),U/L}_{(0,h)}) + c^{U/L}_{(x,y)}.$$

Thereby, it is a forward propagation process by employing the computed bounds metrics of $\Phi^{(l'-r)}(\mathbf{P})$ and $\Phi^{(l'-1)}(\mathbf{P})$ to obtain the bounds of $\Phi^{(l')}(\mathbf{P})$. When it comes to the later layer, the l-th layer, we use the backward process to propagate the bounds to the multiplication layer, which can be referred to as the l'-th layer. Next, at the multiplication layer, we propagate the bounds to the input layer directly by skipping previous layers. The bounds propagating to the multiplication layer (l'-th layer) can be represented by Equation 4.2. Therefore, $\Phi^{l'}(\mathbf{P})$ can be substituted by the linear functions in Equation 4.6 to obtain $\mathbf{A}^{(l,0),U/L}$ and $\mathbf{B}^{(l,0),U/L}$:

$$\mathbf{A}^{(l,0),U/L}_{(x,y,i,:)} = \sum \left(\mathbf{A}^{(l,l'),U/L,+}_{(x,y,i,:)}\mathbf{\Lambda}^{(l',0),U/L}_{(x+i,y,:,:)} + \mathbf{A}^{(l,l'),U/L,-}_{(x,y,i,:)}\mathbf{\Lambda}^{(l',0),L/U}_{(x+i,y,:,:)} \right),$$

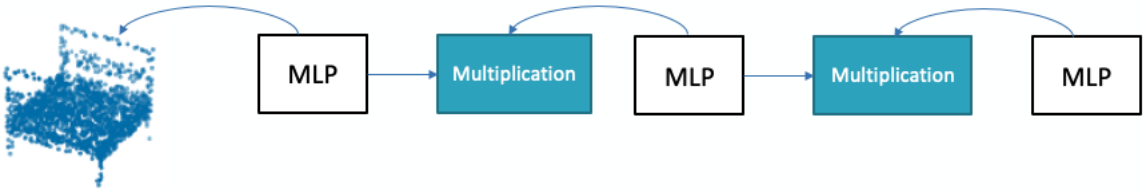


Figure 4.1: Illustration of the combining forward and backward propagation process. The architecture in the MLP block contains convolution with ReLU activation function, batch normalisation, and pooling. Inside the MLP block, the bounds are computed by backward propagation.

$$\mathbf{B}_{(x,y)}^{(l,0),U/L} = \mathbf{B}_{(x,y)}^{(l,l'),U/L} + \sum (\mathbf{A}_{(x,y,i,:)}^{(l,l'),U/L,+} \mathbf{\Theta}_{(x+i,y)}^{(l',0),U/L} + \mathbf{A}_{(x,y,i,:)}^{(l,l'),U/L,-} \mathbf{\Theta}_{(x+i,y)}^{(l',0),L/U}).$$

Lastly, the global bounds of the l-th layer can be computed using the linear functions in Equation 4.4. The combining forward and backward propagation process is demonstrated in Figure 4.1.

4.3 Running Numerical Example

To demonstrate the verification algorithm, we present a simple example network in Figure 4.2 with two input points, p_1 and p_2. Suppose that the input points are bounded by a l_∞-norm ball with radius ϵ, our goal is to compute the lower and upper bounds of the output (p_{11}, p_{12}) based on the input intervals. As Figure 4.2 shows, the neural network contains 12 nodes, and each node is assigned a weight variable. There are three types of operations in the example network: linear operation, non-linear operation (ReLU activation function), and multiplication.

Given the input points $\mathbf{P} = [p_1 = 1, p_2 = 0]$, to obtain the bounds for p_3 and p_4, according to the Equation 4.2, $\mathbf{A}^{(1,0)}$ can be assigned as

$$\mathbf{A}_{(0,0,:,:)}^{(1,0)} = \begin{bmatrix} 1 & 1 \end{bmatrix} \text{ and } \quad \mathbf{A}_{(0,1,:,:)}^{(1,0)} = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

to compute the output for the 1-st layer $[p_3, p_4]$:

$$\begin{aligned} p_3 &= p_1 \cdot \mathbf{A}_{(0,0,0,0)}^{(1,0)} + p_2 \cdot \mathbf{A}_{(0,0,0,1)}^{(1,0)} = 1, \\ p_4 &= p_1 \cdot \mathbf{A}_{(0,1,0,0)}^{(1,0)} + p_2 \cdot \mathbf{A}_{(0,1,0,1)}^{(1,0)} = -1. \end{aligned} \tag{4.7}$$

Based on Equation 4.4, we can obtain the lower bound and upper bound for the perturbed input ($p = \infty$):

$$l_3 = 1 - 2\epsilon = -1, u_3 = 1 + 2\epsilon = 3, l_4 = -1 - 2\epsilon = -3, u_4 = -1 + 2\epsilon = 1.$$

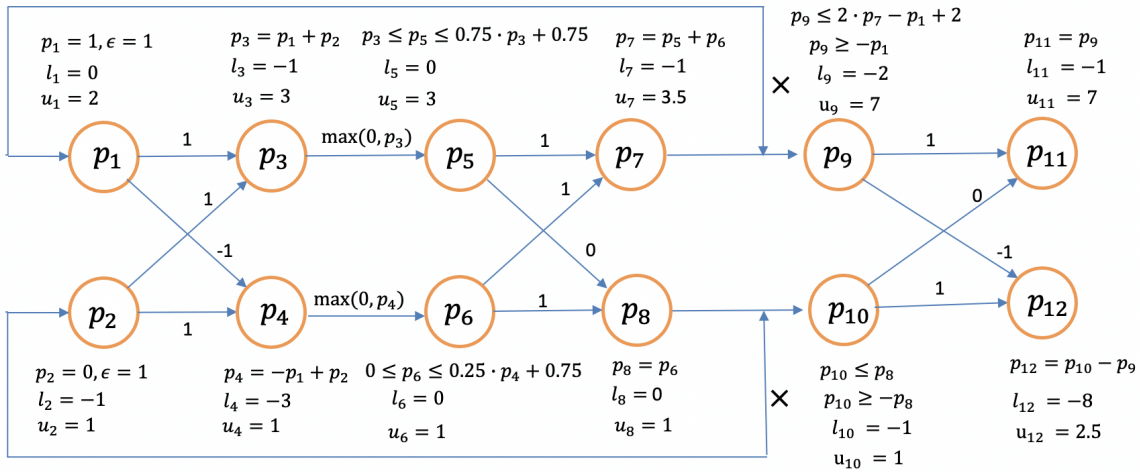


Figure 4.2: A running example for a simple neural network with multiplication. The inputs (p_1, p_2) are bounded by a l_∞-norm ball with radius ϵ.

For the non-linear ReLU activation layer, to compute the bounds of p_5, if $l_3 \geq 0$, p_5 will exactly be the input, p_3; if $u_3 \leq 0$, the p_5 will hold as 0. When $l_3 \leq 0$ and $u_3 \geq 0$, we set the bounds for p_5 as

$$p_5 = \begin{cases} \left[p_3, \frac{u_3(p_3 - l_3)}{u_3 - l_3}\right], & \text{if } u_3 > |l_3| \\ \left[0, \frac{u_3(p_3 - l_3)}{u_3 - l_3}\right], & \text{otherwise} \end{cases} \tag{4.8}$$

according to the relaxation rule, in our example, for p_5, we obtain $\alpha_L = 1$, $\beta_L = 0$, $\alpha_U = 0.75$, and $\beta_U = 0.75$; for p_6, we choose $\alpha_L = \beta_L = 0$, $\alpha_U = 0.25$, and $\beta_U = 0.75$. Then we can obtain the bounds for p_5 and p_6 as shown in Figure 4.2.

Next, for the 3-rd layer output

$$[p_7, p_8] : p_7 = p_5 + p_6, p_8 = p_6,$$

we assign

$$\mathbf{A}^{(3,2),L}_{(0,0,:,:)} = \mathbf{A}^{(3,2),U}_{(0,0,:,:)} = [1, 1], \mathbf{A}^{(3,2),L}_{(0,1,:,:)} = \mathbf{A}^{(3,2),U}_{(0,1,:,:)} = [0, 1]$$

to form:

$$p_5 \cdot \mathbf{A}^{(3,2),L}_{(0,0,0,0)} + p_6 \cdot \mathbf{A}^{(3,2),L}_{(0,0,0,1)} \leq p_7 \leq p_5 \cdot \mathbf{A}^{(3,2),U}_{(0,0,0,0)} + p_6 \cdot \mathbf{A}^{(3,2),U}_{(0,0,0,1)}$$
$$p_5 \cdot \mathbf{A}^{(3,2),L}_{(0,1,0,0)} + p_6 \cdot \mathbf{A}^{(3,2),L}_{(0,1,0,1)} \leq p_8 \leq p_5 \cdot \mathbf{A}^{(3,2),U}_{(0,1,0,0)} + p_6 \cdot \mathbf{A}^{(3,2),U}_{(0,1,0,1)}.$$

We can backpropagate constraints to the first layer to gain final global bounds of p_7 and p_8:

$$\begin{aligned} p_1 + p_2 \leq p_7 \leq 0.5 \cdot p_1 + p_2 + 1.5 \\ 0 \leq p_8 \leq -0.25 \cdot p_1 + 0.25 \cdot p_2 + 0.75. \end{aligned} \tag{4.9}$$

Similarly, we compute the global bounds for p_7 and p_8.

For the multiplication layer, according to the formulations presented in section 3.4, to compute the bounds of p_9 in our example, we set $a^L = l_1$, $a^U = u_1$, $b^L = b^U = l_7$, $c^L = -l_7 \cdot l_1$, and $c^U = -l_7 \cdot u_1$. Similarly, for p_9, we choose $a^L = l_2$, $a^U = u_2$, $b^L = b^U = l_8$, $c^L = -l_8 \cdot l_2$, and $c^U = -l_8 \cdot u_2$. Thus, we get

$$\begin{aligned} -p_1 \leq p_9 \leq 2 \cdot p_7 - p_1 + 2, \\ -p_8 \leq p_{10} \leq p_8. \end{aligned} \tag{4.10}$$

Instead of performing back-propagation to the input layer, we calculate the bounds for p_9 and p_{10} by directly replacing p_7 and p_8 with Equation 4.9:

$$\begin{aligned} -p_1 \leq p_9 \leq 2 \cdot p_2 + 5, \\ 0.25 \cdot p_1 - 0.25 \cdot p_2 - 0.75 \leq p_{10} \leq -0.25 \cdot p_1 + 0.25 \cdot p_2 + 0.75. \end{aligned} \tag{4.11}$$

Again, according to Equation 4.4, we obtain

$$l_9 = -2, u_9 = 7, l_{10} = -1, u_{10} = 1.$$

Finally, in our example, $p_{11} = p_9$ and $p_{12} = p_{10} - p_9$. After propagating the linear bounds to the previous layer by replacing the p_9 and p_{10} with p_7 and p_8 in Equation 4.10, we construct the constraints of the last layer as:

$$\begin{aligned} -p_1 \leq p_{11} \leq 2 \cdot p_7 - p_1 + 2, \\ -p_8 - 2 \cdot p_7 + p_1 - 2 \leq p_{12} \leq p_8 + p_1. \end{aligned} \tag{4.12}$$

By substituting p_7 and p_8 with p_1 and p_2 directly, the global bounds for p_{11} and p_{12} are:

$$\begin{aligned} -p_1 \leq p_{11} \leq 2 \cdot p_2 + 5, \\ 0.25 \cdot p_1 - 2.25 \cdot p_2 - 5.75 \leq p_{12} \leq 0.75 \cdot p_1 + 0.25 \cdot p_2 + 0.75. \end{aligned} \tag{4.13}$$

Thereby, as described so far, the back-substitution yields

$$l_{11} = -1, u_{11} = 7, l_{12} = -8, u_{12} = 2.5,$$

which are the final output bounds of our example network.

Robustness Analysis:

The inputs $p_1 = 1$ and $p_2 = 0$ lead to output $p_{11} = 1$ and $p_{12} = -1$ in our example. Thus, to verify the robustness of our example network, we aim to find the maximum ϵ that guarantees $p_{11} \geq p_{12}$ always holds true for any perturbed inputs within the l_∞-norm ball with a radius ϵ. In our example, the results of l_{11}, u_{11}, l_{12}, and u_{12} conclude that $p_{11} - p_{12} \in [-3.5, 3]$ which results in $p_{11} \geq p_{12}$ failing to hold. Thus, we apply binary search to reduce the value of ϵ and recalculate the output bounds for the network based on the new ϵ. When the maximum iteration is reached, we stop the binary search and choose the maximum ϵ that results in the lower bound of $p_{11} - p_{12} \geq 0$ as the final certified distortion.

4.4 Experiments

4.4.1 Experiments setting

Dataset

We evaluate 3DVerifier on ModelNet40 [161] dataset, which contains 9,843 training and 2,468 testings 3D CAD models. Each CAD model is used to sample a point cloud that comprises 2,048 points in three dimensions [110]. There are 40 categories of CAD models. We run verification experiments on point clouds with 1024, 512, and 64 points, and randomly selected 100 samples from all the 40 categories of the test set as the dataset to perform the experiments. All experiments are carried out on the same randomly selected dataset.

Models

We utilise PointNet [110] models as the 3D point cloud classifiers. Since the baseline verification method, 3DCertify [87], cannot handle the full PointNet with JANet, to make a comprehensive comparison, we first perform experiments on the PointNet without JANet. We then examine the performance of our 3DVerifier on full PointNet models. All models use the ReLU activation function.

Baseline

(1) We choose the existing 3D robustness certifier, 3DCertify [87], as the main baseline for the PointNet models without JANet, which can be viewed as a general CNN. Additionally, we also show the average distortions obtained by adversarial attacks on 3D point cloud models that extended from the CW attack [13], [162] and PGD attack [75]. (2) As for the complete PointNet proposed by [110], we provide the average and minimum distortions obtained by the CW attack for robustness estimation. The PGD attack takes a long time to seek the adversarial examples, and the attack success rate is below 10%, thus, it is not included as the comparative method.

Implementation

The 3DVerifier is implemented via NumPy with Numba in Python. All experiments are run on a 2.10GHz Intel Xeon Platinum 8160 CPU with 512 GB memory.

4.4.2 Results for PointNet models without JANet

In Table 4.1, we present the clean test accuracy (Acc.) and average certified bounds (ave) for PointNet models without JANet. We also record the time to run one iteration of the binary search. We demonstrate that our 3DVerifier can improve upon 3DCertify in terms of

Table 4.1: Average certified bounds (ave) and run-time on PointNet without JANet. For the certified bounds, the higher the better. The bounds obtained by attacks are referred to as upper bounds. '*' means that computing the bounds by 3DCertify is computationally unattainable, which automatically terminates after verifying several samples.

no.	Pool.	Acc.	N	l_p	Certified Bounds						Our vs.		Attack	
					Our			3DCertify			3DCertify (%)		CW	PGD
					min	ave	time	min	ave	time	ave	time	ave	ave
64	Ave	70.1	5	l_∞	0.0007	0.0122	0.5	0.0006	0.0080	1.6	+52.5	-67.9	0.03	0.02
				l_2	0.0014	0.0433	0.5	-	-	-	-	-	0.42	0.53
				l_1	0.0024	0.0553	0.4	-	-	-	-	-	2.50	3.21
		79.3	9	l_∞	16e-5	0.0044	8.9	12e-5	0.0034	92.1	+29.4	-90.4	0.02	0.02
				l_2	0.0014	0.0146	6.8	-	-	-	-	-	0.39	0.46
				l_1	0.0028	0.0188	6.7	-	-	-	-	-	2.97	2.55
	Max	73.6	5	l_∞	1e-4	0.0137	0.4	8e-5	0.0079	0.54	+73.4	-26.5	0.03	0.02
				l_2	0.0008	0.0481	1.5	-	-	-	-	-	0.16	0.28
				l_1	0.0021	0.0649	0.9	-	-	-	-	-	3.11	3.21
		81.2	9	l_∞	0.0007	0.0051	8.3	0.0004	0.0036	51.4	+41.7	-83.8	0.02	0.03
				l_2	0.0045	0.0209	10.5	-	-	-	-	-	0.24	0.16
				l_1	0.0102	0.0288	9.5	-	-	-			2.81	3.09
512	Ave	73.0	5	l_∞	0.0007	0.0104	15.3	0.0006	0.0063	100.4	+15.5	-84.8	0.07	0.02
				l_2	0.0079	0.0427	13.2	-	-	-	-	-	0.41	0.47
				l_1	0.0109	0.0648	12.3	-	-	-	-	-	4.54	4.41
	Max	78.3	5	l_∞	0.0007	0.0108	7.7	0.0006	0.0070	18.5	+54.3	-58.7	0.06	0.02
				l_2	0.0048	0.0313	5.2	-	-	-	-	-	0.26	0.19
				l_1	0.0116	0.0407	3.8	-	-	-	-	-	2.90	3.10
		82.1	9	l_∞	0.006	0.0048	60.2	*	*	*	-	-	0.03	0.02
				l_2	0.0180	0.0159	58.8	-	-	-	-	-	0.26	0.36
				l_1	0.0632	0.0196	54.3	-	-	-	-	-	4.49	5.08
1024	Ave	72.6	5	l_∞	0.0010	0.0145	43.1	*	*	*	-	-	0.05	0.04
				l_2	0.0125	0.0471	21.9	-	-	-	-	-	0.46	0.39
				l_1	0.0354	0.0544	10.1	-	-	-	-	-	2.38	3.02
	Max	77.8	5	l_∞	0.0006	0.0135	14.4	0.0002	0.0085	21.0	+58.8	-31.6	0.06	0.04
				l_2	0.0205	0.0393	19.0	-	-	-	-	-	0.36	0.27
				l_1	0.0493	0.0500	18.1	-	-	-	-	-	3.47	2.45

Table 4.2: Average certified bounds (ave) and times on PointNet models with T-Net

no.points	Pooling	Acc.	N	l_p	Our			CW	
					ave	min	time	ave	min
64	Average	70.71	8	l_∞	0.341	0.011	0.12	0.727	0.021
				l_2	0.862	0.077	0.19	1.700	0.162
				l_1	1.479	0.136	0.18	2.049	0.317
		80.22	12	l_∞	0.316	0.016	2.98	0.832	0.053
				l_2	0.588	0.032	2.86	2.164	0.276
				l_1	1.115	0.113	2.73	2.949	0.510
		84.11	15	l_∞	0.248	0.012	23.98	0.792	0.077
				l_2	0.466	0.026	23.25	3.310	0.313
				l_1	0.857	0.088	22.79	4.857	0.875
	Max	74.63	8	l_∞	0.554	0.029	0.43	0.791	0.093
				l_2	1.118	0.039	0.41	2.133	0.137
				l_1	1.650	0.041	0.397	3.078	0.322
		81.91	12	l_∞	0.541	0.007	4.26	0.846	0.078
				l_2	0.897	0.010	4.11	2.554	0.102
				l_1	1.326	0.010	3.18	3.772	0.167
		86.77	15	l_∞	0.122	0.027	24.42	0.218	0.044
				l_2	0.494	0.030	24.13	1.285	0.076
				l_1	0.615	0.035	23.29	1.721	0.103
512	Average	73.53	12	l_∞	0.231	0.005	71.26	0.395	0.035
				l_2	1.196	0.027	68.73	3.106	0.637
				l_1	1.622	0.107	66.91	4.852	0.833
	Max	84.61	12	l_∞	0.109	0.000	55.78	0.693	0.033
				l_2	0.272	0.000	53.64	0.882	0.085
				l_1	0.345	0.043	50.22	1.175	0.448
1024	Average	71.47	12	l_∞	0.406	0.017	129.41	0.721	0.041
				l_2	1.142	0.181	127.88	1.459	0.342
				l_1	1.574	0.233	126.24	1.926	0.596
	Max	80.22	12	l_∞	0.374	0.001	128.22	0.758	0.005
				l_2	0.959	0.006	127.51	1.591	0.009
				l_1	1.253	0.009	120.75	2.163	0.018

run-time and tightness of bounds. To make the comparison more extensive, we train a 5-layer (N=5) and a 9-layer (N=9) model. The specific configurations of the model structure are presented in Appendix A. We also show the performance of models with different types of pooling layers: global average pooling and global maximum pooling. For experiments, we set the initial ϵ as 0.05 and the maximum iteration of binary search as 10. The experiments are taken on three point-cloud datasets with 64, 512, and 1024 points. We can see from the results of average bounds that our 3DVerifier can obtain tighter lower bounds than 3DCertify and engages a significant gap in the distortion bounds found by CW and PGD attacks. Table 4.1 shows that our method is much faster than the 3DCertify. Notably, 3DVerifier enables an orders-of-magnitude improvement in efficiency for large networks. Additionally, our method can compute certified bounds of distortion constrained by l_1, l_2, and l_∞-norms, which is more scalable than 3DCertify in terms of norm distance.

4.4.3 Results for PointNet models with JANet

As the 3DCertify does not include the bounds computation algorithm for multiplication operation, it can not be applied to verify the complete PointNet with JANet architecture. Thus, as the first work to verify the point cloud models with JANet, we show the average and minimum distortion bounds of the C&W attack-based method to make comparisons. We examine N-layer models (N=8,12,15) with two types of global pooling: average pooling and maximum pooling on datasets with 64, 512, and 1024 points. The obtained certified average bounds of full PointNet models are shown in Table 4.2. According to previous verification works on images (e.g. [7]) and results in Table 4.1, the gap between certified bounds and attack-based average distortion is reasonable, where the average minimum distortion is ten times greater than the bounds. Thus, it reveals that our method efficiently certifies the point cloud models with JANet in a comparable quality with models without JANet.

4.4.4 Increasing number of test samples

In table 4.3 we present the results for verifying 1000 samples on 64 points. The results show that the obtained average certified bounds and run time for one epoch are similar to those under 100 samples. As the average certified bound is more related to the threat model, 100 samples are enough to evaluate the average certified bound, as they are randomly selected from different categories.

4.4.5 Experiments on extra datasets

Our certification method has shown to be versatile and applicable to a wide range of point cloud datasets. In addition to the ModelNet40 dataset, we have also tested our method on other 3D point cloud datasets. Our results have shown that our certification method

Table 4.3: Average certified bounds (ave) and run-time on PointNet without JANet on 1000 samples. For the certified bounds, the higher, the better. The bounds obtained by attacks are referred to as upper bounds.

Pooling	Acc.	N	l_p	Certified Bounds						Our vs.	
				Our			3DCertify			3DCertify (%)	
				min	ave	time	min	ave	time	ave	time
Ave	70.1	5	l_∞	0.00001	0.0122	0.77	0.000006	0.0077	1.34	+58.44	-42.5
			l_2	0.00007	0.0437	0.67	-	-	-	-	-
			l_1	0.00025	0.0562	0.81	-	-	-	-	-
	79.25	9	l_∞	0.00002	0.0045	10.87	0.00001	0.0033	80.5	+36.36	-86.50
			l_2	0.0001	0.0146	9.85	-	-	-	-	-
			l_1	0.00022	1.0187	9.43	-	-	-	-	-
Max	73.55	5	l_∞	0.00001	0.0142	0.44	0.000007	0.0078	0.552	+82.05	-20.3
			l_2	0.00009	0.0494	1.03	-	-	-	-	-
			l_1	0.00021	0.0666	0.84	-	-	-	-	-
	81.22	9	l_∞	0.00001	0.0050	12.73	0.00001	0.0036	50.22	+38.89	-74.65
			l_2	0.00009	0.0210	11.77	-	-	-	-	-
			l_1	0.0002	0.0289	10.43	-	-	-	-	-

generalizes well to these datasets, indicating its effectiveness and potential for broader application.

The ModelNet10 dataset is particularly noteworthy, which is also collected from 3D CAD models, and is a very clean dataset, where the 10 most popular object categories are manually selected and are then manually aligned with the orientation of the CAD models for this 10-class subset. This dataset provides an excellent benchmark for evaluating the robustness and accuracy of certification methods such as ours. Furthermore, we also present the experiments in full PointNet using our method on the SydneyUrban dataset, which contains various common road objects scanned with a Velodyne HDL-64E LIDAR. There are 631 scans of objects in total that contains categories of signs, vehicles, pedestrians, signs, and trees, which can be visualised in Figure 4.3. This dataset presents a unique set of challenges due to the complexity and variability of the objects, as well as the noise and occlusions present in the scans. Despite these challenges, our certification method has demonstrated its ability to accurately certify point cloud models of objects in this dataset.

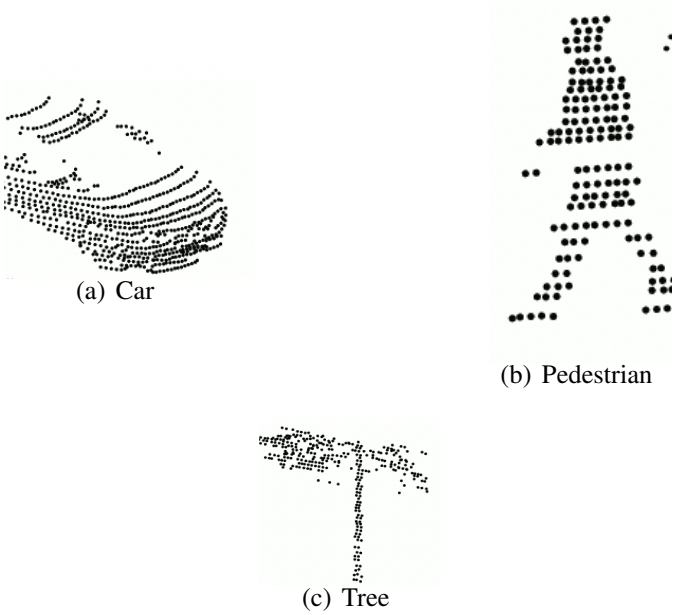
Overall, our experiments on different 3D point cloud datasets demonstrate the effectiveness and robustness of our certification method and its potential for use in a wide range of applications. We use ModelNet10 on 64 points to compare our work and baseline work on 64 points. The results shown in Table 4.4 confirm that our work still engages tighter bound and less run-time. We perform our method on ModelNet10 and SydneyUrban dataset on 64 points with an 8-layer full PointNet model, and the results are demonstrated in Table 4.5.

Table 4.4: Average certified bounds (ave) and run-time on PointNet without JANet. For the certified bounds, the higher, the better. The bounds obtained by attacks are referred to as upper bounds for ModelNet10 with 64 points.

Pooling	Acc.	l_p	Certified Bounds						Our vs.	
			Our			3DCertify			3DCertify (%)	
			min	ave	time	min	ave	time	ave	time
Ave	75.1	l_∞	0.00176	0.0139	0.93	0.00113	0.0084	1.51	+55.75	-38.41
		l_2	0.0132	0.0611	1.01	-	-	-	-	-
		l_1	0.0233	0.0788	1.22	-	-	-	-	-
Max	78.2	l_∞	0.00083	0.0215	0.66	0.00077	0.0177	0.71	+21.5	-7.04
		l_2	0.0044	0.0680	0.95	-	-	-	-	-
		l_1	0.0061	0.0903	1.22	-	-	-	-	-

Table 4.5: Average certified bounds (ave) and times on PointNet models with T-Net for ModelNet10 and Sydney Urban dataset with 64 points

Dataset	Pooling	Acc.	N	l_p	Our			CW	
					ave	min	time	ave	min
ModelNet10	Average	72.1	8	l_∞	0.451	0.01	0.14	0.931	0.022
				l_2	1.29	0.07	0.27	1.992	0.141
				l_1	2.05	0.13	0.18	2.591	0.422
	Max	78.2	8	l_∞	0.535	0.01	0.31	0.621	0.067
				l_2	1.534	0.04	0.34	2.773	0.152
				l_1	1.821	0.07	0.31	3.266	0.411
Sydney	Average	62.1	8	l_∞	0.668	0.010	0.18	1.225	0.018
				l_2	1.66	0.020	0.16	2.830	0.132
				l_1	1.993	0.020	0.10	3.744	0.210
	Max	67.4	8	l_∞	0.497	0.0002	0.11	0.843	0.072
				l_2	0.685	0.0003	0.08	2.771	0.152
				l_1	0.968	0.0004	0.10	1.439	0.334



(a) Car

(b) Pedestrian

(c) Tree

Figure 4.3: Visualization of Sydney Urban dataset

4.4.6 Importance of JANet

To demonstrate the significance of JANet in point cloud classification tasks, we conducted an ablation study by training three different models and recording the number of trainable parameters for each one. The results of the training accuracy are presented in Table 4.6, and the specific layer configurations for these models are outlined in Appendix A.

Since the PointNet without JANet can be viewed as a general convolutional neural network (CNN) model, while JANet is a more intricate architecture that includes a T-Net and multiplication layer, we examined two models for the PointNet without JANet to better understand the effect of JANet. The first model was a 7-layer model that omitted JANet from the 12-layer full PointNet, and the second model was a 12-layer model that added convolution and dense layers in T-Net to the 7-layer model.

The results indicated that the PointNet with JANet improved the training accuracy significantly compared to the PointNet without JANet when using the same number of layers. As a result, JANet played a crucial role in enhancing the performance of point cloud models.

Table 4.6: Training accuracy of different structures of point cloud models.

Models	N	no. trainable parameters	accuracy
PointNet without JANet	7	443656	72.94%
PointNet without JANet	12	822472	74.07%
Full PointNet with JANet	12	645425	81.31%

4.5 Discussions

3DVerifier is efficient for large-scale 3D point cloud models

There are two key features that enable 3DVerifier's efficiency for large 3D networks.

- *Improved global max pooling relaxation:* the relaxation algorithm for the global max pooling layer of CNN-Cert [7] framework cannot be adapted directly to 3D point cloud models, which is computationally unattainable. Thus, we proposed a linear relaxation for the global max pooling layer based on [126]. For example, to find the maximum value $p_r = max_{m\in\mathcal{M}}(p_m)$, if exists p_j such that its lower bound $l_j \geq u_k$ for all $k \in \mathcal{M}\backslash j$, the lower bound for the max pooling layer is $l^r = l_j$ and upper bound is $u^r = u_j$. Otherwise, we set the output of the layer $\phi^r \geq p_j$, where $j = \underset{m\in\mathcal{M}}{argmax}(l_m)$, and similarly $\phi^r \leq p_k$, where $k = \underset{m\in\mathcal{M}}{argmax}(u_m)$. The comparison results in Table 4.1 indicate that implementing the improved relaxation for the max pooling layer enables 3DVerifier to compute the certified bounds much faster than the existing method, 3DCertify.

- *Combing forward and backward propagation:* The verification algorithm proposed in [123] evaluated the effectiveness of combining forward and backward propagation to compute the bounds for Transformer with self-attention layers. Thus, in our tool 3DVerifier, we adapted the combining forward and backward propagation to compute the bounds for the multiplication layer. As Table 4.2 shows, the time spent for full PointNet models with JANet is nearly the same as the time for models without JANet.

CNN-Cert can be viewed as a special case of 3DVerifier

CNN-Cert [7] is a general efficient framework for verifying the neural networks for image classification, employing 2D convolution layers. Although our verification method shares a similar design mechanism as CNN-Cert, our framework is superior to CNN-Cert. One key difference is the dimension of input data, PointNet 3D models adopt 1D convolution layers, which 3DVerifier can efficiently handle. Additionally, besides the general operations such as pooling, batch normalization, and convolution with ReLU activation, we can also

handle models with JANet that contain multiplication layers. Thus, 3DVerifier can tackle more complex and larger neural networks than CNN-Cert. To adapt the framework for 3D point clouds, we introduced a novel relaxation method for max-pooling layers to obtain its certified bounds, which significantly improves the verification efficiency.

4.6 Conclusion

In this article, we introduced a comprehensive and efficient framework for verifying the robustness of 3D point cloud models. Our approach utilises linear relaxation for the multiplication layer and integrates forward and backward propagation, enabling us to swiftly compute certified bounds for different model architectures, including convolution, global pooling, batch normalisation, and multiplication. Our experiments on diverse models and point clouds, featuring varying numbers of points, demonstrate that 3DVerifier outperforms other methods regarding both computational efficiency and tightness of the certified bounds. One potential application of our proposed framework is in autonomous driving, where the robustness of 3D point cloud models is crucial for ensuring the safety of passengers and pedestrians. We can provide guarantees on the correctness of these models, making them more reliable and trustworthy.

Chapter 5

Certified Policy Smoothing for Cooperative Multi-Agent Reinforcement Learning

"Artificial intelligence is the new electricity. Just as electricity transformed almost everything 100 years ago, today I have a hard time thinking of an industry that I don't think AI will transform in the next several years."
(Andrew Ng.)

5.1 Introduction

The cooperative multi-agent reinforcement learning (c-MARL) has devoted a great deal of attention to a wide range of applications in the real world, such as autonomous cars [120], traffic lights controlling [142], and packet delivery [171]. The target of RL for single agents is to learn the action that can optimise the long-time reward, but when the environment is interacted with by a team of agents, the system needs to jointly optimise the action of each agent to maximise the accumulated team reward.

In recent years, there has been increasing interest in developing adversarial attacks on deep reinforcement learning (DRL) systems. Most of the existing attack solutions have focused on attacking single-agent RL systems. Huang, Papernot, Goodfellow, *et al.* [50] proposed an attack method called FGSM-RL, which uses the Fast Gradient Sign Method to generate adversarial examples for deep reinforcement learning agents. The method is effective in deceiving the agents into taking incorrect actions. Then, Lin, Hong, Liao, *et al.* [84] introduced an attack method called Adversarial Attack on Deep Reinforcement Learning using Policy Gradient Descent (PGD-RL), which uses policy gradient descent to optimise the adversarial perturbations. This approach is effective in generating more powerful and targeted attacks against DRL agents. To generate the attacks that are hard to detect, Kos and Song [68] proposed a reinforcement learning-based attack method called Adversarial Reinforcement Learning (ARL), which learns to generate adversarial examples that can deceive the DRL agents. Weng, Dvijotham, Uesato, *et al.* [152] proposed a method to attack bandit algorithms, which is based on adversarial multi-armed bandit algorithms. The RL has also been shown to be vulnerable to the perturbation on the environments [40].

There have been some recent efforts to extend adversarial attacks to cooperative multi-agent RL systems (c-MARLs). Two notable works in this area are [83] and [108]. Lin, Dzeparoska, Zhang, *et al.* [83] proposed a policy network that is trained to find a wrong action that the victim agent is expected to take and set as the targeted adversarial example. This approach is designed to deceive the victim agent into taking the wrong action, which can have negative consequences on the overall performance of the c-MARL system. Pham, Nguyen, Chen, *et al.* [108] then proposed to craft a stronger adversary by using a model-based approach.

Some adversarial defence works for RL are proposed [27], [30], [122], [132] and then towards these defences, stronger attacks are proposed [116], [117]. To end this repeated game, Lütjens, Everett, and How [89] first proposed a certified defence on the observations of DRLs. Zhang, Chen, Xiao, *et al.* [177] then provided empirically provable certificates to ensure that the action does not change at each state. However, this method cannot provide robustness certification for the reward if the action is changed under attacks. To tackle this problem, Kumar, Levine, and Feizi [73] proposed to directly certify the total reward via randomised smoothing-based defence, but this method cannot achieve robustness certification at the action level. Wu, Li, Huang, *et al.* [159] then proposed a policy smoothing method based on the randomised smoothing of the action-value function, which is chosen as

the baseline in this chapter for certifying the robustness of global reward under a single-agent scenario. However, all existing methods can only work on single-agent systems. To the best of our knowledge, this paper is the first work to certify the robustness of cooperative multi-agent RL systems.

Compared with the RL system with a single agent, certifying the c-MARL is more challenging work. which are addressed as below:

- *Action space grows exponentially in c-MARLs.* Certifying cooperative multi-agent reinforcement learning (c-MARL) is a more demanding task compared to RL systems with a single agent. This is because the number of feasible actions increases exponentially with the number of agents, and at each time step, all agents must be certified simultaneously, leading to a rise in uncertainty in determining the final decision bound.

- *Different agents have different contributions to the team reward.* In c-MARLs, modifying the action of a single agent may not impact the overall team reward, as it is a collective effort. As a result, existing certification works for single-agent systems are not sufficient in evaluating the robustness of the c-MARL system. To address this issue, new criteria must be developed to assess the robustness of the multi-agent system, which needs to consider the interdependence of the agents and the impact of their actions on the team's performance.

To cope with such challenges, we proposed two novel frameworks to certify the robustness of each state and the whole trajectory. We first propose a smoothed policy that each agent will choose the most provable action given the observation $\mathcal{N}(o, \sigma^2 I)$ and then we derive the certified radius for each agent per step, within which the chosen action of the agent will not be changed. We identify the multiple testing problem when evaluating the robustness for all agents per step, and propose to assert each agent's importance per step via the importance factor adapting from [36]. Multiplying each agent's importance factor with its p-value, we can use the Benjamini-Hochberg (BH) to control the false discovery rate (FDR) for each step. As for certifying the whole trajectory, we propose to apply the tree-based search algorithm to find the certified radius as well as the lower bound of the team reward with in this radius.

In this chapter, we focused on certifying the robustness of value-based c-MARL models under l_2-norm bounded attack. Our work can be easily extended to evaluate l_p robustness by using different sampling distributions like generalised Gaussian distribution as indicated in [45].

5.2 Methodology

In this section, we first outline an intuitive approach to certifying RL based on current classifier certification. Then, we sketch the challenges preventing the direct use of the intuitive approach and present how to address these challenges.

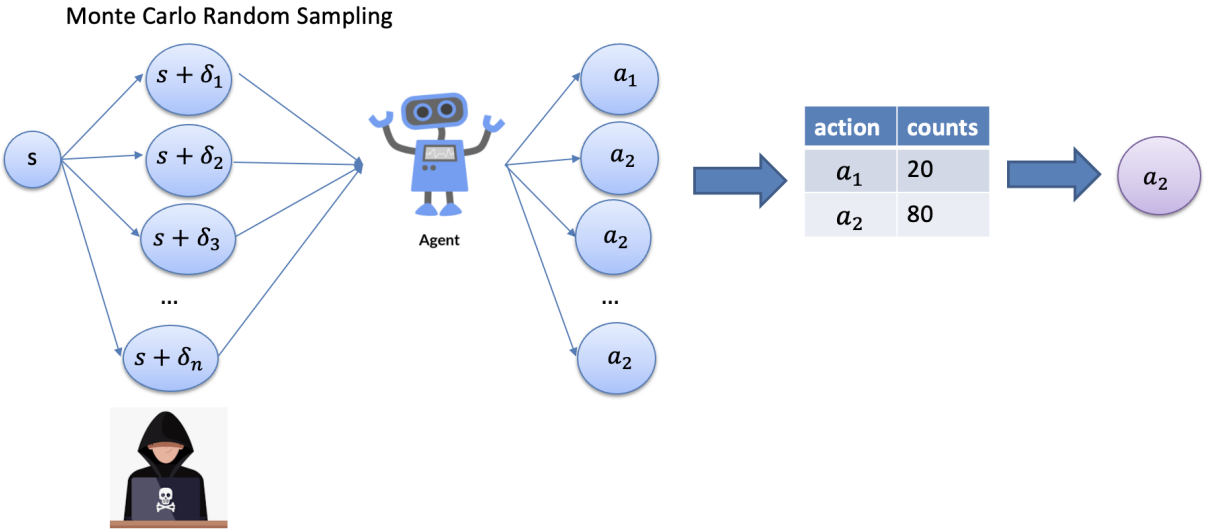


Figure 5.1: Abstract workflow for the intuitive approach for single-agent reinforcement learning. In the figure, δ is the noise sampled from Monte Carlo Random Sampling, where $(\delta_1, ..., \delta_n)$ i.i.d. $\mathcal{N}(0, \sigma^2 I)$

5.2.1 Problem Formulation

We aim to design a robust policy for multi-agent reinforcement learning algorithms. Following the standard set of existing adversarial attacks on c-MARLs, e.g. [83], where the adversarial perturbation is added to each step's observation of each agent, our proposed policy is expected to be provably robust against the perturbation bounded by the l_2-norm around the observation of each agent.

Definition 1. *(Smoothed policy) Given a trained multi-agent reinforcement learning network Q^π with policy π, suppose that there are N agents, at the time step t, let $\forall s_t \in S$, given that the noise vector $\Delta_t = (\delta_t^1, ..., \delta_t^N)$ is $i.i.d$ $\mathcal{N}(0, \sigma^2 I)$, the joint smoothed policy can be represented as*

$$\tilde{\pi}(s_t) = \arg\max_{\mathbf{a}_t \in \mathcal{A}} \widetilde{Q}^\pi (s_t + \Delta_t, \mathbf{a}_t). \tag{5.1}$$

To certify the robustness of the smoothed policy, we define the certification robustness for a per-step action as

$$\tilde{\pi}_t(s_t) = \tilde{\pi}_t(s_t + \epsilon_t) \quad s.t. \forall \epsilon_t, ||\epsilon_t||_2 \leq D, \tag{5.2}$$

where $\epsilon_t \in \mathbb{R}^N$ represents the maximum perturbation applied to the observations of each agent at the t-th time step. In other words, for each agent, in the presence of the l_2-norm bounded perturbation in each state, the smoothed policy is expected to return the same action that is most likely to be selected in the unperturbed state s_t.

Algorithm 6 Intuitive Policy Smoothing for Certifying Per-state Action

Input: Trained Q^π with N agents
Parameter: sampling times M; Gaussian distribution parameter σ; confidence parameter α
Function: BioPVALUE: two-sided hypothesis test; MultiConBnd: function to calculate the probability bounds with confidence $1-\alpha$.

function SMOOTHING(M, Q, α, σ)
  for $m \leftarrow 1, M$ **do** ▷ Get smoothed policy $\tilde{\pi}$
    generate $\Delta_m = (\delta_m^1, ..., \delta_m^N)$ i.i.d $\mathcal{N}(0, \sigma^2 I)$
    $s' \leftarrow s + \Delta_m$
    $\mathbf{a} \leftarrow \pi(s')$
    Add $\mathbf{a} \rightarrow Actlist$
  return $Actlist$
$Actlist \leftarrow$ SMOOTHING$(M, Q^\pi, \alpha, \sigma)$
$\mathbf{a}^m, \mathbf{a}^r, ct_1, ct_2 \leftarrow$ Top two action sets with their counts
if $BioPVALUE(ct_1, ct_1 + ct_2, 0.5) \leq \alpha$ **then**
  $Cert \leftarrow True$ ▷ Get certified radius for $\tilde{\pi}$
  $\underline{p_{\mathbf{a}^m}}, \overline{p_{\mathbf{a}^r}} \leftarrow MultiConBnd(Counts(Actlist), \alpha)$
  $D \leftarrow \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_{\mathbf{a}^m}}\right) - \Phi^{-1}\left(\overline{p_{\mathbf{a}^r}}\right)\right)$
else
  $Cert \leftarrow False$, $D \leftarrow 0$
return $\mathbf{a}, d, Cert$

5.2.2 Intuitive Approach

Intuitively, the randomised smoothing can be adapted to certify the robustness of the per-state action in RLs by replacing the classifier with policy $\pi(s_t)$. For the certification of each step, Monte Carlo randomised sampling is used to estimate the smoothed policy $\tilde{\pi}$. As shown in Algorithm 6, we record the action vector $\mathbf{a}$, which is a combination of actions taken by all agents at each sampling step. The most likely selected action set is chosen as the action was taken by $\tilde{\pi}$. A larger number of samples can be used to estimate the lower bound on the probability ($\underline{p_{\mathbf{a}^m}}$) of the most frequently selected action set, $\mathbf{a}^m$, and the upper bound on the probability ($\overline{p_{\mathbf{a}^r}}$) of the second most frequently selected ("runner-up") action, $\mathbf{a}^r$. The function MULTICONBND in Algorithm 6 is based on a Chi-Square approximation [42], which takes the number of observations for each category as input and returns the $(1-\alpha)$ confidence levels.

Lemma 3. *Let X and Y be random variables from Gaussian distributions, $X \sim \mathcal{N}(x, \sigma^2 I)$ and $Y \sim \mathcal{N}(x+\delta, \sigma^2 I)$. Suppose h ($\mathbb{R}^d \rightarrow 0, 1$) is a random or deterministic function. Then we can have:*
a. If $S = \left\{z \in \mathbb{R}^d : \delta^T z \leq \beta\right\}$ for some β and $\mathbb{P}(h(X) = 1) \geq \mathbb{P}(X \in S)$, then $\mathbb{P}(h(Y) =$

$1) \geq \mathbb{P}(Y \in S)$.
b. if $S = \left\{z \in \mathbb{R}^d : \delta^T z \geq \beta\right\}$ *for some* $t > 0$ *and* $\mathbb{P}(h(X) = 1) \leq \mathbb{P}(X \in S)$*, then* $\mathbb{P}(h(Y) = 1) \leq \mathbb{P}(Y \in S)$.

Proposition 1. *If the certification in Algorithm 6 returns the action set* $\mathbf{a}^m : (a^{m,1}, a^{m,2}, ..., a^{m,N})$ *in the time step* t *with a certified radius*

$$D = \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_{\mathbf{a}^m}}\right) - \Phi^{-1}\left(\overline{p_{\mathbf{a}^r}}\right)\right)$$

then with probability at least $(1-\alpha)$*, the smoothed policy* $\tilde{\pi}(s_t + \epsilon_t)$ *chooses the action* $\mathbf{a}^m$*,* $\forall ||\epsilon_t||_2 \leq D$.

Proof. To prove that the smoothed policy $\tilde{\pi}(s_t + \epsilon_t)$ returns the action $\mathbf{a}^m$, the following constraint should be certified:

$$\mathbb{P}\left(\tilde{\pi}(s_t + \epsilon_t) := \mathbf{a}^m\right) > \max_{\mathbf{a}^m \neq \mathbf{a}^r} \mathbb{P}\left(\tilde{\pi}(s_t + \epsilon_t) := \mathbf{a}^r\right)$$

Therefore, we need to guarantee $\mathbb{P}\left(\tilde{\pi}(s_t + \epsilon_t) := \mathbf{a}^m\right) > \mathbb{P}\left(\tilde{\pi}(s_t + \epsilon_t) := \mathbf{a}^r\right)$ to make sure the action set $\mathbf{a}^m \neq \mathbf{a}^r$. Suppose we have two random variables $\mathbf{X}$ and $\mathbf{Y}$:

$$\begin{aligned}\mathbf{X} &:= s_t + \Delta = \mathcal{N}\left(s_t, \sigma^2 I_N\right)\\ \mathbf{Y} &:= s_t + \varepsilon_t + \Delta = \mathcal{N}\left(s_t + \epsilon, \sigma^2 I_N\right)\end{aligned}$$

As we have obtained the lower bound on the probability ($\underline{p_{\mathbf{a}^m}}$) of the most frequently selected action set, $\mathbf{a}^m$, and the upper bound on probability ($\overline{p_{\mathbf{a}^r}}$) of the second most frequently selected ("runner-up") action, $\mathbf{a}^r$, with confidence (1-α) calculated by the MULTICONBND function based on a chi-squared approximation [42], which takes the number of observations for each category as input and output the confidence levels.

In this domain, we get

$$\begin{aligned}\mathbb{P}\left(\tilde{\pi}(\mathbf{X}) := \mathbf{a}^m\right) &\geq \underline{p_{\mathbf{a}^m}}\\ \mathbb{P}\left(\tilde{\pi}(\mathbf{X}) := \mathbf{a}^r\right) &\leq \overline{p_{\mathbf{a}^r}},\end{aligned}$$

As we are going to show:

$$\mathbb{P}\left(\tilde{\pi}(\mathbf{Y}) := \mathbf{a}^m\right) > \mathbb{P}\left(\tilde{\pi}(\mathbf{Y}) := \mathbf{a}^r\right) \tag{5.3}$$

the half-spaces can be defined as:

$$\begin{aligned}A &:= \left\{y : \epsilon_t^T(y - s_t) \leq \sigma\|\epsilon_t\|\Phi^{-1}\left(\underline{p_{\mathbf{a}^m}}\right)\right\}\\ B &:= \left\{y : \epsilon_t^T(y - s_t) \geq \sigma\|\epsilon_t\|\Phi^{-1}\left(1 - \overline{p_{\mathbf{a}^r}}\right)\right\}\end{aligned} \tag{5.4}$$

Noticing that $\mathbb{P}(\mathbf{X} \in A) = p_{\mathbf{a}^m}$, we can conclude $\mathbb{P}\left(\pi(\mathbf{X}) := \mathbf{a}^m\right) \geq \mathbb{P}(\mathbf{X} \in A)$. Hence, in Lemma 3, let $h(y) := \mathbf{1}\left[\pi(y) := \mathbf{a}^m\right]$, we can have

$$\mathbb{P}(\pi(\mathbf{X}) = \mathbf{a}^r) \geq \mathbb{P}(\mathbf{Y} \in A) \tag{5.5}$$

Similarly, given $\mathbb{P}(\mathbf{X} \in B) = \overline{p_{\mathbf{a}^r}}$, and $\mathbb{P}(\pi(\mathbf{X}) = \mathbf{a}^r) \leq \mathbb{P}(\mathbf{X} \in B)$, we can define $h(y) := \mathbf{1}[\pi(y) := \mathbf{a}^r]$and conclude that

$$\mathbb{P}(\pi(\mathbf{Y}) := \mathbf{a}^r) \leq \mathbb{P}(\mathbf{Y} \in B) \tag{5.6}$$

To ensure Equation 5.3, based on the Equation 5.5 and 5.6, it suffices to show that

$$\mathbb{P}(\mathbf{Y} \in A) > \mathbb{P}(\mathbf{Y} \in B),$$

which completes the chain of inequalities:

$$\mathbb{P}(\pi(\mathbf{Y}) := \mathbf{a}^m) \geq \mathbb{P}(\mathbf{Y} \in A) > \mathbb{P}(\mathbf{Y} \in B) \geq \mathbb{P}(\pi(\mathbf{Y}) := \mathbf{a}^r) \tag{5.7}$$

Finally, we can calculate the following equations:

$$\begin{aligned} \mathbb{P}(\mathbf{Y} \in A) &= \Phi\left(\Phi^{-1}\left(\underline{p_{\mathbf{a}^m}}\right) - \frac{\|\epsilon_t\|}{\sigma}\right) \\ \mathbb{P}(\mathbf{Y} \in B) &= \Phi\left(\Phi^{-1}\left(\overline{p_{\mathbf{a}^r}}\right) + \frac{\|\epsilon_t\|}{\sigma}\right) \end{aligned} \tag{5.8}$$

Based on above analysis, $\mathbb{P}(\mathbf{Y} \in A) > \mathbb{P}(\mathbf{Y} \in B)$ is true if and only if :

$$\|\epsilon_t\| < \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_{\mathbf{a}^m}}\right) - \Phi^{-1}\left(\overline{p_{\mathbf{a}^r}}\right)\right). \tag{5.9}$$

□

The intuition behind the method shown in Algorithm 6 is similar to the certification procedure for classification through randomised smoothing [24]. The BIOPVALUE is applied to calculate the p-value of the two-sided hypothesis test to choose the action $\mathbf{a}^m$. However, rather than abstaining from the action when the p-value does not meet the confidence level, we set the certified radius of this step as $D = 0$ to indicate that the certification failed, since the RL relies on decisions of multiple steps. When $n = 1$, the algorithm can be used to certify RLs with a single agent as Wu, Li, Huang, *et al.* [159], but instead of using the smoothed action value function Q^π, we utilise the frequency of occurrence of each action to determine which action to be selected. Since c-MARLs are trained under the premise that each agent would always select the best action, they do not reliably anticipate the team reward when some agents behave badly.

In the c-MARLs, there are some additional challenges that preclude us from using this intuitive certification criterion.

Challenge 1. The perturbation D added to the observation of each agent can be different.

For c-MARLs, each agent develops its own policy to choose its action. If the certified bound is calculated using Algorithm 6, all agents will engage with the same perturbation

bound, making the results less accurate for each agent. As one agent can be more robust than the other, the same perturbation added to the agents will lead to different performances, which provides the need to certify the robustness of each agent. Thus, we will first consider certifying the robustness for every agent and then estimating the robustness in each state for all agents. To reduce the computation cost, we can sample from the joint policy $\pi(s')$ instead of each agent's policy separately. To this end, we can change $\mathbf{a}^m, \mathbf{a}^r$ in Algorithm 6 (Line 9) to the two most likely actions $(a^{m,n}, a^{r,n})$ for each agent and then calculate the corresponding lower bound $\underline{p_{a^{m,n}}}$ on probability $\mathbb{P}(\tilde{\pi}^n(z^n) := a^{m,n})$ and upper bound $\overline{p_{a^{r,n}}}$ for choosing the "runner up" action, $a^{r,n}$. The certified bound for each agent per state can be computed as:

Corollary 1. *(Certification for the actions of each agent in each state) In state s, given the joint smoothed policy $\tilde{\pi}(s) = \{\tilde{\pi}^1(z^1), ..., \tilde{\pi}^N(z^N)\}$, we can obtain the certified bound in state s for each agent to guarantee $\tilde{\pi}^n(z^n + \epsilon) := a^{m,n}, \forall ||\epsilon||_2 \leq d_n$:*

$$d_n = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_{a^{m,n}}}) - \Phi^{-1}(\overline{p_{a^{r,n}}})) \tag{5.10}$$

Proof. Similar as the proof for **Proposition 1**, we modified the certification goal as:

$$\mathbb{P}\left(\tilde{\pi}(z^n + \epsilon) := a^{m,n}\right) > \max_{a^{m,n} \neq a^{r,n}} \mathbb{P}\left(\tilde{\pi}(z^n + \epsilon) := a^{r,n}\right)$$

Thus, we need to show

$$\mathbb{P}\left(\tilde{\pi}(z^n + \epsilon) := a^{m,n}\right) > \mathbb{P}\left(\tilde{\pi}(z^n + \epsilon) := a^{r,n}\right)$$

Now the two random variables $\mathbf{X}$ and $\mathbf{Y}$ are defined as :

$$\mathbf{X} := z^n + \delta^n = \mathcal{N}\left(z^n, \sigma^2 I\right)$$
$$\mathbf{Y} := z^n + \varepsilon + \delta^n = \mathcal{N}\left(z^n + \epsilon, \sigma^2 I\right)$$

Following the above proof, we can calculate the lower bound $\underline{p_{a^{m,n}}}$ on probability $\mathbb{P}(\tilde{\pi}^n(z^n) := a^{m,n})$ and upper bound $\overline{p_{a^{r,n}}}$ for the probability of choosing the "runner up" action, $a^{r,n}$, we get

$$\mathbb{P}\left(\tilde{\pi}(\mathbf{X}) := a^{m,n}\right) \geq \underline{p_{a^{m,n}}}$$
$$\mathbb{P}\left(\tilde{\pi}(\mathbf{X}) := a^{m,r}\right) \leq \overline{p_{a^{m,n}}},$$

We are going to show:

$$\mathbb{P}\left(\tilde{\pi}(\mathbf{Y}) := a^{m,n}\right) > \mathbb{P}\left(\tilde{\pi}(\mathbf{Y}) := a^{m,r}\right) \tag{5.11}$$

The half-spaces for each agent can be defined as:

$$\begin{aligned} A &:= \left\{y : \epsilon^T(y - z^n) \leq \sigma\|\epsilon\|\Phi^{-1}\left(\underline{p_{a^{m,n}}}\right)\right\} \\ B &:= \left\{y : \epsilon^T(y - z^n) \geq \sigma\|\epsilon\|\Phi^{-1}\left(1 - \overline{p_{a^{m,r}}}\right)\right\} \end{aligned} \tag{5.12}$$

Hence, following the proof of proposition 1, in Equation 5.5 and Equation 5.6, we can replace the $\mathbf{a}^m$ and $\mathbf{a}^r$ with $a^{m,n}$ and $a^{m,r}$ to show

$$\mathbb{P}\left(\pi(\mathbf{Y}) := a^{m,n}\right) \geq \mathbb{P}(\mathbf{Y} \in A) > \mathbb{P}(\mathbf{Y} \in B) \geq \mathbb{P}\left(\pi(\mathbf{Y}) := a^{m,r}\right) \tag{5.13}$$

Then, we can conclude that, $\mathbb{P}(\mathbf{Y} \in A) > \mathbb{P}(\mathbf{Y} \in B)$ is true if and only if :

$$\|\epsilon\| < \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_{a^{m,n}}}\right) - \Phi^{-1}\left(\overline{p_{a^{m,n}}}\right)\right). \tag{5.14}$$

□

Finally, the most likely chosen action for each agent can be combined as the final action set $\{a^{m,1}, a^{m,2}, ..., a^{m,N}\}$ and the certified bound at each step can be defined as:

Definition 2. *Given the certified bound obtained for each agent in state s, $\{d_1, d_2, ..., d_N\}$, the certified bound in this state for all agents is determined by the least robust agent: $D = min\{d_1, d_2, ..., d_N\}$.*

Challenge 2. If we choose the bound of the least robust agent as the bound for all agents per state, the confidence level decays.

As Proposition 1 indicates, on each call of certification, the certified robustness bound obtained only holds with confidence level $(1 - \alpha)$. As we sample noise from the Gaussian distribution *independently*, the hypothesis tests are *independent*. Based on Definition 2, to calculate the certified bound for each state, we have the following constraint for the probability of making an error:

$$\begin{aligned}&\mathbb{P}(\bigvee\nolimits_{n\in N}, n\text{-th agent's cert failed }) \leq \\ &\min\left(\sum\nolimits_n \mathbb{P}(n\text{-th agent's cert failed }), 1\right) = \min(N\alpha, 1).\end{aligned}$$

Therefore, for multiple tests, without any control over the error, the probability of making an error will increase with the number of tests. Suppose that there are T steps in the entire trajectory, we will have $N * T$ tests in total, which can be a great challenge. To address this problem, for certifying per-state actions, the confidence level can be reduced to α/N. Additionally, we can first perform agent selection to control the selective error by considering the importance of each agent, since sometimes an agent changing its action will not diminish the team reward. Moreover, to evaluate the global certification bound, we propose a tree-search-based method to find the lower bound of the team reward. In Section 5.3 and 5.4, we will detail our proposal to certify the robustness of per-state actions and global reward.

5.3 Robustness Certification for Per-State Action with Correction

5.3.1 Multiple Hypothesis Testing

Corollary 2. *(Certified bound per state) In state s, given N agents with action* $\mathbf{a}$*, the joint policy is $\pi(s) = \{\pi^1(z^1), ..., \pi^N(z^N)\}$. Suppose that the observation of each agent is perturbed by random noise δ^n, where $\delta^n \sim \mathcal{N}(0, \sigma^2 I)$. $\forall n \in N$, if $\mathbb{P}(\pi^n(z^n + \delta^n) := a^{m,n}) \geq 0.5$, we can compute the certified bound by* ***Definition 2****.*

Proof. For every agent n $\in N$, with confidence $1 - \alpha$, the

$$p_{a^{m,n}} = \mathbb{P}(\pi^n(z^n + \delta^n) := a^{m,n}) \geq 0.5$$

is satisfied. Then we invoke Corollary 1 to compute the certified bound d_n and guarantee that

$$\tilde{\pi}^n(z^n + \epsilon) := a^{m,n}, \forall ||\epsilon||_2 \leq d_n.$$

therefore, with $D = min\{d_1, d_2, ..., d_N\}$, we can show that

$$\forall n \in N, \tilde{\pi}^n(z^n + \epsilon) := a^{m,n}, \forall ||\epsilon||_2 \leq D \leq d_n.$$

D is determined as the certified bound per state that holds for and agents satisfy $p_{a^{m,n}} \geq 0.5$. □

In order to obtain the certified bound for all agents per state, we can employ Corollary 2, and, as suggested, for each agent, we need to ensure that condition

$$\mathbb{P}(\tilde{\pi}^n(z^n) := a^{m,n}) \geq 0.5$$

is satisfied. Hence, after sampling, with the count (ct_1^n) for the most frequent action taken by agent n, we can implement the one-sided binomial test to obtain its p-value pv_n. These p-values can be processed to indicate which tests should be accepted under $(1 - \alpha)$ confidence.

Definition 3. *(Hypothesis Test) The hypothesis test with null hypothesis for each agent is*

$$H_0 : \mathbb{P}(\tilde{\pi}^n(z^n) := a^{m,n}) < 0.5,$$

and the alternative is

$$H_1 : \mathbb{P}(\tilde{\pi}^n(z^n) := a^{m,n}) \geq 0.5.$$

In the hypothesis test, if the null hypothesis H_0 is true, we can determine the p-value, which is the probability of finding a statistic that is equally extreme as the observed one or more extremes. Given the statistical test in **Definition 3**, if the p-value is below the

Algorithm 7 Certified Robustness Bound of the Perturbation for Actions of Each State with Correction (CRSC)

Input: Trained Q^{π}; N agents;

Parameter: sampling size M; Gaussian distribution parameter σ; confidence parameter α

1: $Actlists \leftarrow \text{SMOOTHING}(M, Q^{\pi}, \alpha, \sigma)$
2: $a^{m,n}, a^{r,n}, ct_1^n, ct_2^n \leftarrow Counts(Actlists[n])$ for $n \in N$
3: $IF \leftarrow IF\_function(Q^{\pi}, Actlists)$
▷ Obtain importance factor for agent
4: $pv_n \leftarrow BioPVALUE(ct_1^n, M, 0.5)$ for $n \in N$
5: $c_n \leftarrow BHproc((pv_n * IF[n]), \alpha)$ for $n \in N$
6: If $\neg c_n : d_n \leftarrow 0$ ▷ Remove failed agent
7: $\mathcal{I}_{\boldsymbol{cert}} := \{n \mid d_n \neq 0\}$ ▷ Obtain certified agent set
8: Compute d_n for each agent in $\mathcal{I}_{\boldsymbol{cert}}$
9: $D = min(d_n | n \in \mathcal{I}_{\boldsymbol{cert}})$
10: **return** D, $\mathcal{I}_{\boldsymbol{cert}}$

confidence level, we can reject the null hypothesis, which means that the bound is certified; otherwise, we accept it.

In multiple hypothesis tests, the probability of the occurrence of false positives (FP) will increase, where the FP denotes that we reject the null hypothesis when it is true, which is also called *type I error*. Suppose that the confidence level is α, and the probability of FP is expected to be less than α. To control *type I error* for multiple tests with H tests, the family-wise error rate (FWER) is introduced, which changes α for each test to α/H. However, it is still conservative, which can increase the true negative rate (i.e., *type II error*).

To solve this problem, Benjamini and Hochberg [5] proposed the false discovery rate (FDR) to find the expected false positive portion. The FDR method applies a corrected p-value for each test case, achieving a better result: testing for as many positive results as possible while keeping the false discovery rate within an acceptable range. The Benjamini-Hochberg (BH) procedure first sorts the p-values of tests in ascending order and then finds the largest k such that $p_k \leq k\alpha/H$, rejecting null if the p-value is below p_k. Fithian, Sun, and Taylor [35] then proposed selective hypothesis tests by applying inference to the selected model to control the selective *type I error*, which controls the global error as

$$\frac{\mathbb{E}[\#FalseRejections]}{\mathbb{E}[\#H_0Selected]} \leq \alpha.$$

Inspired by the selective hypothesis tests, we propose to multiply every agent's importance factor with its p-value to control the selective FDR via executing the BH procedure on the corrected p-values.

5.3.2 Measuring the Importance of Agents

To obtain each agent's important factor, we can measure each agent's contribution to the team reward at each state. We adapt the advantage function proposed in COMA [36], which is used to decentralise agents by estimating the individual reward during training. As the importance factor defined in **Definition** 4, it is applied to examine the behaviour of the current action of the agent.

Definition 4. *For each agent, n, the importance factor IF^n of each agent is computed by comparing the Q value of the current action a^n with the counterfactual reward baseline, which is obtained by altering the action of agent n, $a^{n'}$, and keeping the other agents' actions $\mathbf{a}^{-n}$ unchanged:*

$$IF^n(s, \mathbf{a}) = Q(s, \mathbf{a}) - \sum_{a^{n'} \in \mathcal{A}} \mathbb{P}(\tilde{\pi}^n(s) := a^{n'}) \cdot Q\left(s, \left(\mathbf{a}^{-n}, a^{n'}\right)\right).$$

Algorithm 7 shows the process for certifying the robustness of the actions of each state while controlling the error. To correct the p-value in the multiple tests, we adapt the p-value for each test by multiplying it with the agent's importance factor (Line 4). Then we can perform the BH procedure (Line 5) to determine which tests should be rejected. Lastly, we obtain the set of certified agents $\mathcal{I}_{\boldsymbol{cert}}$ with certified bounds.

Theorem 3. *For each agent in $\mathcal{I}_{\boldsymbol{cert}} := \{n \mid d_n \neq 0\}$, the action can be certified as*

$$\tilde{\pi}^n(z^n + \epsilon^n) = \tilde{\pi}^n(z^n),$$

where $\|\epsilon^n\|_2 \leq D := min(d_n), \forall n \in \mathcal{I}_{\boldsymbol{cert}}$.

Proof. Considering each agent independently, given that agent n updates its policy $\tilde{\pi}^n(z^n)$ in each state, under the condition

$$\mathbb{P}(\tilde{\pi}^n(z^n) := a^{m,n}) > 0.5,$$

we can obtain the lower bound probability of selecting the $a^{m,n}$ and the upper-bound probability for the "runner-up" action, $a^{r,n}$, for each agent and then compute the certified bound d_n. The minimum certified bound holds for any agent that satisfies the condition, denoted by the set $\mathcal{I}_{\boldsymbol{cert}}$. □

5.4 Robustness Guarantee on Global Reward

To certify the bound of global reward under the certified perturbation bound for each step, the CRSC is no longer applicable, as it cannot find the lower bound of global reward. Therefore,

Algorithm 8 Tree-Search-based certified robustness bound and global reward (T-CRGR)

Input: Trained Q^{π}; N agents; confidence parameter α
Parameter: sampling times M; Gaussian distribution parameter σ

function GETNODE(s)
  $Actlists \leftarrow$ SMOOTHING$(M, Q^{\pi}, \alpha, \sigma)$
  $IF \leftarrow IF\_function(Q, Actlists)$
  $A\_dic, d\_list \leftarrow \emptyset$
  for $n \in \mathcal{I}_{agent}$ **do**
    $a^{m,n}, a^{r,n}, ct_1^n, ct_2^n \leftarrow Counts(Actlists)$
    $pv_n \leftarrow BioPVALUE(ct_1^n, M, 0.5)$
    if $pv_n * IF[n] > \alpha$ **then**
      $A\_dic[n] \leftarrow A\_dic[n] \cup \{a^{m,n}, a^{r,n}\}$
      $\underline{p_1} \leftarrow BioConBnd(ct_1^n + ct_2^n, M, 1-\alpha)$
    else
      $A\_dic[n] \leftarrow A\_dic[n] \cup \{a^{m,n}\}$
      $\underline{p_1} \leftarrow BioConBnd(ct_1^n, M, 1-\alpha)$
    $d\_list \leftarrow d\_list \cup (\sigma\Phi^{-1}(\underline{p_1}))$
  $d \leftarrow min(d\_list)$
  return A_dic, d
function SEARCH($d, s, \mathbf{a}, R, done$):
  if $R \geq R_{min}$ **then** ▷ Prune the tree
    return 0
  if $done$ **then**
    $R_{min} \leftarrow min(R, R_{min})$
    return 0
  $A\_dic, d_{new} \leftarrow$ GETNODE(s)
  $d \leftarrow min(d_{new}, d), Action\_list \leftarrow A\_dic$
  for $\mathbf{a}$ in $Action\_list$ **do**
    $s_{new}, done \leftarrow env.step(\mathbf{a}, s)$
    SEARCH$(d, s_{new}, \mathbf{a}, R + Q(s, \mathbf{a}), done)$

we propose a tree-search-based method to find the global lower bound of the team reward under the certified bound of perturbation.

The insight of implementing the search tree is that, if we cannot certify the bound of perturbation at some time steps for some agent, we can take the second most frequent action, which will result in a new trajectory. Then we can explore the new trajectory by developing it as an expanded branch of the search tree, which may result in a lower global reward. Thus, the minimum reward can be determined as the certified lower bound of the global reward after exploring all trajectories. The main function for the tree-search-based method is presented in Algorithm 8. As it shows, at first, we figure out all possible actions to formulate the action list to be explored using the function GETNODE. Then we perform the SEARCH function to expand the tree based on each action node. Once all new trajectories have been explored, we obtain the certified bound of perturbation and the minimum reward among all leaf nodes. We also apply to prune to control the size of the search tree, which requires the reward in the environments to be non-negative. When the cumulative reward of the current node has already reached the lower bound, it can be pruned, as the subsequent tree will not lead to a lower bound.

5.5 Experiments

We first present the certified lower bound of the global reward under the certified perturbation, then we show the certified robustness for actions per state. Moreover, since our method is general and can be applied in single-agent systems, we will show the comparison experiments with the state-of-the-art baseline on certifying the global reward for RL with a single agent.

5.5.1 Experimental Setup

Baseline

In single-agent environments, we compare our method with the state-of-the-art RL certification algorithm, CROP-LORE [159]. CROP-LORE is based on local policy smoothing that has a similar goal to our work – obtaining a lower bound of the global reward under the certified bound for the actions of each state. Since CROP-LORE also employs the tree-search-based algorithm, we follow the same setting for a fair comparison. For certifying the c-MARLs, since there is no existing solution, we apply the PGD attack [**PGD**] to demonstrate the validity of the certified bounds.

RL Algorithms

We apply our method to certify the DQN trained by SA-MDP (PGD) and SA-MDP (CVX) [177] in the single-agent setting since they have been empirically shown to achieve the

highest certified robustness among all the baselines examined. For c-MARLs, we use VDN [133] and QMIX [113], which are well-established value-based algorithms.

Experiments setup

For all experiments, we sample noise 10,000 times for smoothing and set the discount factor γ to 1.0. In the single-agent environment, we follow the same setting as the baseline, where the time step is 200 and the confidence level is $\alpha = 0.05$. For c-MARLs, $\alpha = 0.01$.

5.5.2 Environments settings

For the single agent environment, we use the "Freeway" in OpenAI Gym [8], which is the most stable game reported in the baseline. To demonstrate the performance of our method on c-MARLs, we choose two environments "Checkers" with *two* agents, "Switches" with *four* agents and "Traffic Junction" with *four* and *ten* agents from ma-gym [70].

In the single-agent environment, we follow the settings in the baseline. The Freeway is an Atari-2600 environment that can be implemented by OpenAI Gym [8] on the top of the Arcade Learning Environments. The input states are high-dimensional images with shape $210 \times 160 \times 3$ and the actions are discrete actions. The experiments use the NoFrameSkip-v4 version, where the randomness of the environments is fully controlled by setting the environments' random seed at the beginning.

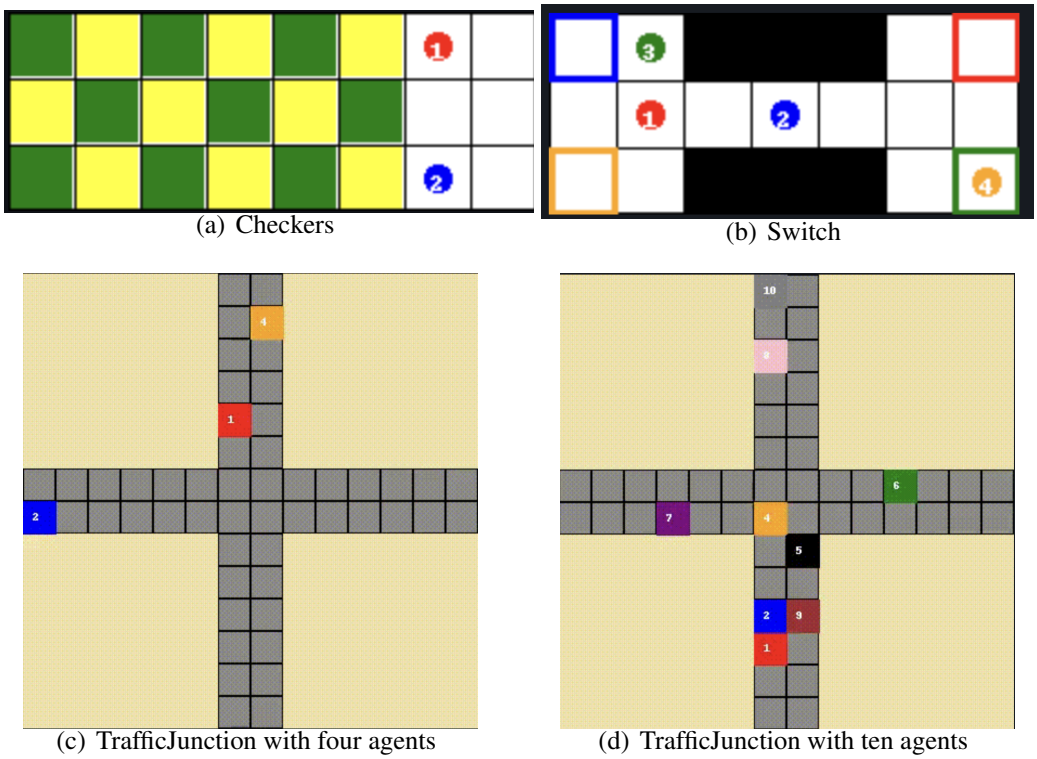
As for the c-MARL environments, we choose two maps shown in Figure 5.2 from the ma-gym [70]: Checkers with two agents and switch with four agents, which are used in [133]. The observations are byte values of size $3 \times 5 \times 5$ RGB images. The action spaces are: 0 (Down), 1 (Left), 2 (Up), 3 (Right), and 4 (Noop). The version used is Checkers-v0 and Switch4-v0, where each agent observes only its local position.

Checkers

The map of checkers contains apples and lemons. The red agent will give a higher score for the team: +10 for the apple (green) and -10 for the lemon (yellow). The blue agent will give +1 for the apple (green) and -1 for the lemon. In this game, the red agent is expected to consume all apples and the blue agent needs to leave them to its mate and eat the obstructing lemons.

Switch

This game is a grid-world environment with four agents. Each agent expects to move to their home which is marked in the box with the same outlined colour. The challenge is how to pass through the narrow corridor, since at each time, only one agent can pass through.



(a) Checkers (b) Switch

(c) TrafficJunction with four agents (d) TrafficJunction with ten agents

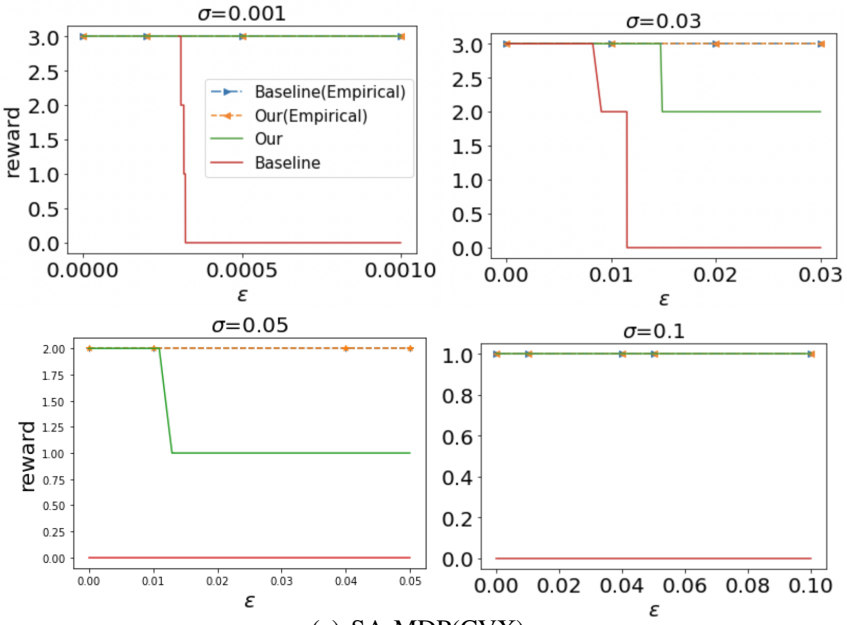
Figure 5.2: c-MARLs environments

Whenever the agent reaches their home block, a +5 score will be awarded, until all agents arrive at their home or reach the maximum of 100 steps.

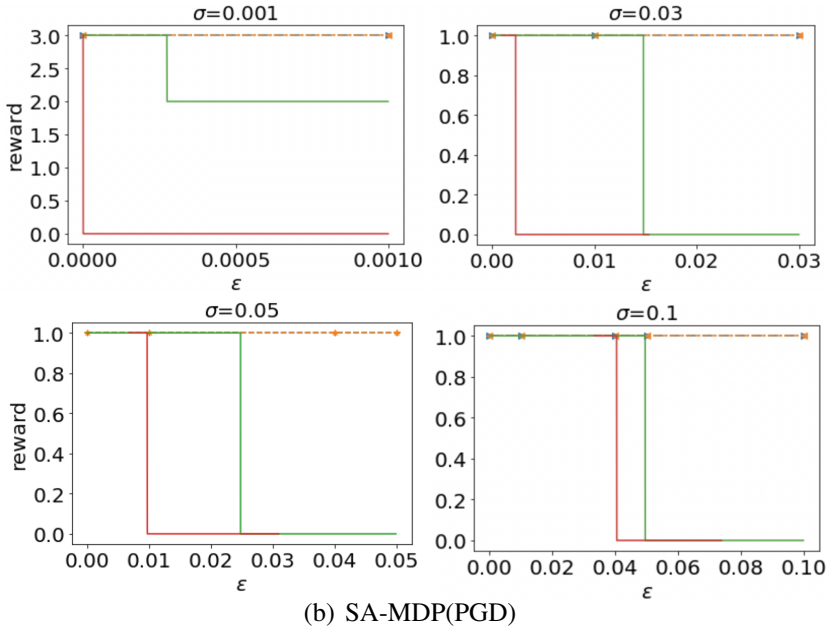
Traffic Junction

On a grid of 14 by 14, there is a 4-way intersection. New cars (agents) join the grid with a probability from each of the four directions at each time step. However, there can never be more than $Nmax$ cars on the road at once. At each time, the car occupies a single block and is allotted one of three potential paths at random. Each agent has two options at a time step: gas, which moves it forward one cell on its path, or stops, which keeps it in place. Once a car arrives at its destination at the grid's edge, it will be eliminated.

The overlap of two cars' locations denotes the collision, which incurs a reward of -10. To prevent a traffic jam, each car receives the reward of $\tau * r_{time} = -0.01\tau$ at every time step, where τ is the number of time steps passed.



(a) SA-MDP(CVX)



(b) SA-MDP(PGD)

Figure 5.3: Comparing the robustness certification of the total reward for SA-MDP in Freeway with Wu, Li, Huang, *et al.* [159]. Solid lines are the certified lower bounds of reward, and dashed lines indicate the empirical results under the PGD attack.

Table 5.1: Lower bound of global reward under the minimum certified bound of perturbation ϵ, where the line with '*' denotes that we run the trajectory to the end without pruning to obtain the certified reward.

Game	No. agent	Models	σ	ϵ_{cert}	Reward	
					Our	PGD
Checkers	2	VDN	0.03	0.0117	79.84	79.84
			0.06	0.0221	79.84	79.84
			0.1	0.0309	79.84	79.84
		QMIX	0.03	0.0144	19.96	19.96
			0.06	0.0369	19.96	19.96
			0.1	0.0384	19.96	19.96
Switch	4	VDN	0.03	0.0147	19.4	19.4
			0.06	0.284	14.4	19.4
			0.1	0.036	14	14.4
		QMIX *	0.03	0.0173	-20	-20
			0.06	0.0233	-20	-20
			0.1	0.038	-20	-20
TrafficJunction	4	VDN	0.03	0.0171	-103.1	-102.83
			0.06	0.0310	-103.25	-103.25
			0.1	0.0536	-105.36	-105.36
		QMIX	0.03	0.0150	-146.24	-134.5
			0.06	0.0315	-109.94	-109.19
			0.1	0.0501	-189.4	-172.7
TrafficJunction	10	VDN	0.03	0.0164	-202.0	-202.0
			0.06	0.0347	-202.0	-202.0
			0.1	0.0618	-202.0	-202.0
		QMIX	0.03	0.0165	-175.23	-137.74
			0.06	0.0398	-175.23	-139.6
			0.1	0.0556	-175.45	-79

5.5.3 Evaluate the Robustness of the Global Reward on Single Agent

The baseline develops the smoothed policy based on the action-value function bounded by Lipschitz continuous, while our method is based on the probability of selecting the most frequent action. To make a fair comparison, we employ the same search tree structure as the baseline, which organises all possible trajectories and grows them by increasing the certified bound to choose an alternative action. we adapted our method to the similar structure of the tree used in the baseline.

As Algorithm 9 shows, our adapted tree AT-CRGR aims to explore all possible trajectories by progressively growing it as CROP-LoRE that is used in the baseline. First, the root node, which is at depth 0 of the tree, represents the initial state $s0$. As each node of the tree denotes the state, for each node, we can calculate a sequence of non-decreasing bound of perturbation $d^k(s)$, where k is the number of all possible actions at state s. If current d_{lim} is between the $d^i(s)$ and $d^{i+1}(s)$, we can grow (i+1) branches for state s. We repeat the same procedure for the newly expanded branch to expand the tree, until achieving the terminate state or the number of node depths to H. We keep track of the cumulative reward for each trajectory as we grow the tree and update the lower bound of the reward at the end of the trajectory.

Following the setting in the baseline, we apply the perturbation magnitude growth to find the certification under a larger perturbation. The priority queue is employed to choose the next critical d_{lim} effectively. When exploring the trajectory, we will store the possible action with their certified bound $d^k(s)$ into the priority queue. As the perturbation grows, these actions whose certified bound below the perturbation are going to be explored. Thus, after all, trajectories for d_{lim} are explored, we will grow the d_{lim} to the next perturbation magnitude by popping out the element in the priority queue.

As shown in Figure 5.3, our method obtains a tighter bound than the baseline. Since we measure the probability of selecting actions instead of the action value function to calculate the bound and choose an action, we do not include the actions that have never been chosen in the possible action list, leading to a more reasonable action selection mechanism, resulting in a tighter calculated bound. Moreover, the Lipschitz continuity is used to compute the upper bound of the smoothed value function in the baseline, which is less tight than our bound based on high-probability guarantees.

Comparison on time complexity

The time cost is related to the number of agents and complexity of the network, ranging from 30 mins to 2 days. For CROP-LoRE, the complexity is

$$O(H|S_{explored} \times (log|S_{explored} + |A|T)),$$

where $|S_{explored}|$ is the number of states that have been explored in the search procedure. H represents the horizon length, A denotes the size of the action set and T is the time spent

sampling the noise.

Compared with the complexity of CROP-LoRE, our method can reduce the size of action space $|A|$ to limit it by the number of actions that have been explored, which decreases the total complexity to

$$O(H|S_{explored} \times (log|S_{explored} + |A_{possible}|T)).$$

Algorithm 9 Adapted Search-Tree based Certified robustness bound and global reward (AT-CRGR)

Input: Trained Q^{π}; N agents; confidence parameter α
Parameter: sampling times M; Gaussian distribution parameter σ

function GETNODE(s, d_{lim}, R)
  $Que \leftarrow \emptyset$ ▷ Initialise the empty Que containing the tuple (s, a,d, R)
  $A\_dic \leftarrow \emptyset$
  $Actlists \leftarrow$ SMOOTHING$(M, Q^{\pi}, \alpha, \sigma)$
  for $n \in N$ **do**
    $poss\_action \leftarrow Sort(Counts(Actlist[n]))$ ▷ Get possibles actions for agent n
    $a^* \leftarrow argmax(Counts(Actlist[n]))$
    $A\_dic[n] \leftarrow A\_dic[n] \cup \{a^*\}$
    $ct_1^n, ct_2^n \leftarrow$ Top two in $Counts(Actlist[n])$
    for a in $poss\_action$ **do**
      if $a == a^*$ **then**
        $pv \leftarrow BioPVALUE(ct_1^n, ct_1^n + ct_2^n, \alpha)$
        if $pv > \alpha$ **then**
          d=0
        else
          $\underline{p} \leftarrow BioConBnd(ct_1^n, M, 1-\alpha)$
          d = $\sigma\Phi^{-1}\left(\underline{p}\right)$
      else
        $ct_a^n \leftarrow Counts(Actlist[n])[a]$
        $pv \leftarrow BioPVALUE(ct_1^n, ct_1^n + ct_a^n, \alpha)$
        if $pv > \alpha$ **then**
          d=0
        else
          $\underline{p_{a^*}}, \overline{p_a} \leftarrow MultiConBnd(Counts(Actlist[n]), \alpha)$
          $d = \frac{\sigma}{2}\Phi^{-1}\left(\underline{p_{a^*}}\right) - \Phi^{-1}\left(\overline{p_a}\right)$
      if $d \leq d_{lim}$and a not in $A\_dic[n]$ **then**
        $A\_dic[n] \leftarrow A\_dic[n] \cup \{a\}$
      else

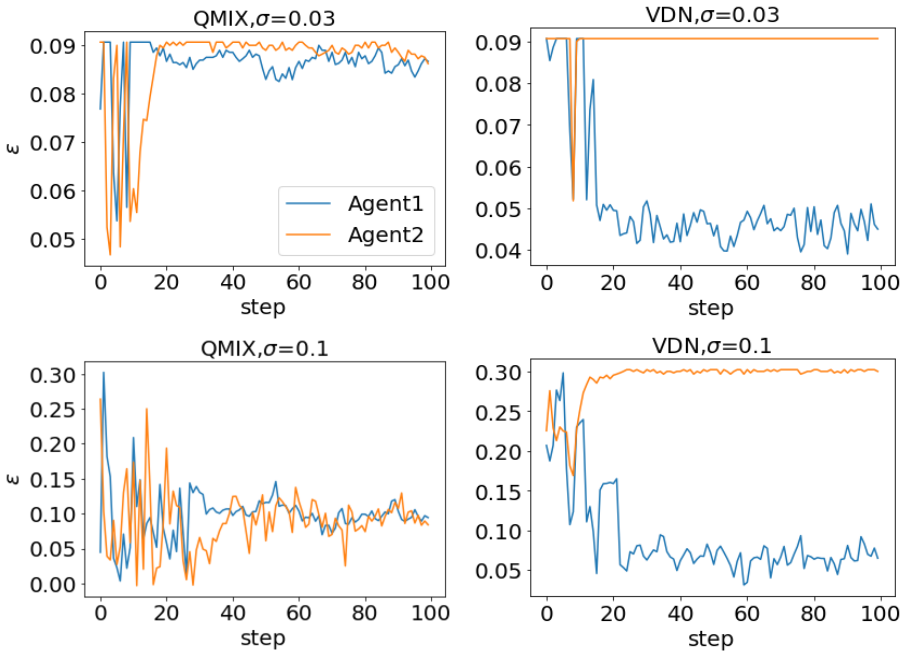
$Que.push(s, a, d, R)$
Return A_dic
function SEARCH($s, d_{lim}, R, done$):
    if $R \geq R_{tot}$ **then** ▷ Prune the tree
        return 0
    if $done$ **then**
        $R_{tot} \leftarrow min(R, R_{tot})$
        return 0
    $A\_dic \leftarrow$ GETNODE(s, d_{lim}, R)
    for $\mathbf{a}$ in A_dic **do**
        $s_{new}, done \leftarrow env.step(s, \mathbf{a})$
        SEARCH($s_{new}, d_{lim}, R + Q(s, \mathbf{a}), done$)
SEARCH($s_0, d_{lim} = 0, R = 0$) ▷ Construct initial trajectory
while True **do**
    if $Que = \emptyset$ **then**
        break
    $(s, \mathbf{a}, d, R) \leftarrow Que.pop()$ ▷ Pop out the first element in the Que
    $(\_, \_, d', \_) \leftarrow Que.top()$ ▷ Check the next element
    $Map[d] \leftarrow R_{tot}$
    SEARCH($env.step(s, \mathbf{a}), d', R + Q(s, \mathbf{a})$)

5.5.4 Evaluate the Robustness of the Global Reward on C-MARLs

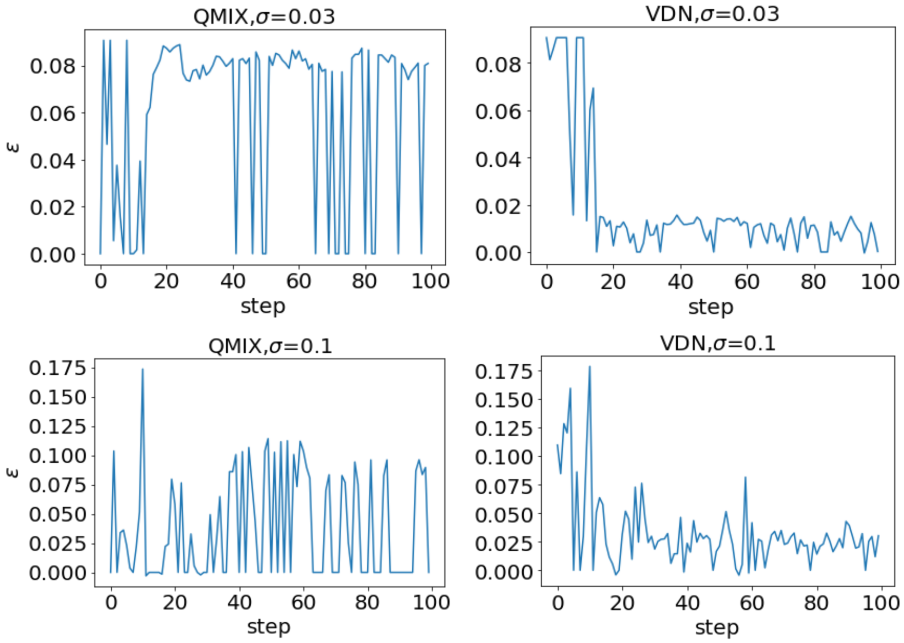
In Table 5.1, we show the results of the lower bound of global reward under the minimum certified bound of perturbation ϵ_{cert}. To perform pruning, the per-step reward in each environment is set to be non-negative. However, as the global reward obtained for QMIX on Switch are below zero, for this case, we run each trajectory to the end without pruning to calculate the global reward. We can see that VDN obtains a higher reward compared to QMIX but is less robust (has lower ϵ_{cert}). This is because, during the training process, VDN simply adds rewards obtained by the two agents to achieve a centralisation, leading one agent to choose a simpler strategy once another agent has learned a useful strategy. On the other hand, QMIX employs a more complex network to centralise the agents instead of only adding their rewards, which helps the network to capture more complex interrelationships between different agents and encourage each one to learn. This leads to VDN achieving higher rewards faster than QMIX but being more vulnerable to perturbations.

5.5.5 Evaluate the Robustness for Each State

In Figure 5.4, 5.5,5.6 and 5.7, we present the certified perturbation bounded by l_2 norm for each agent and for all agents at each state separately. We see that in Checkers with two

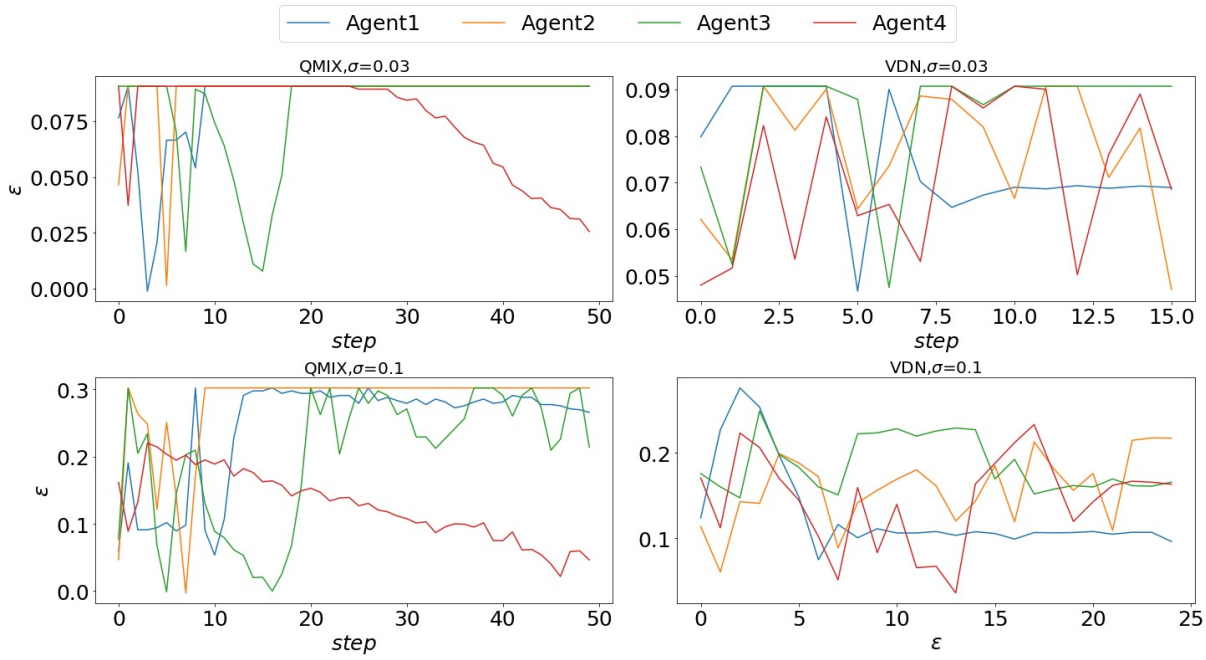


(a) Certified bound of perturbation for per-state action of each agent

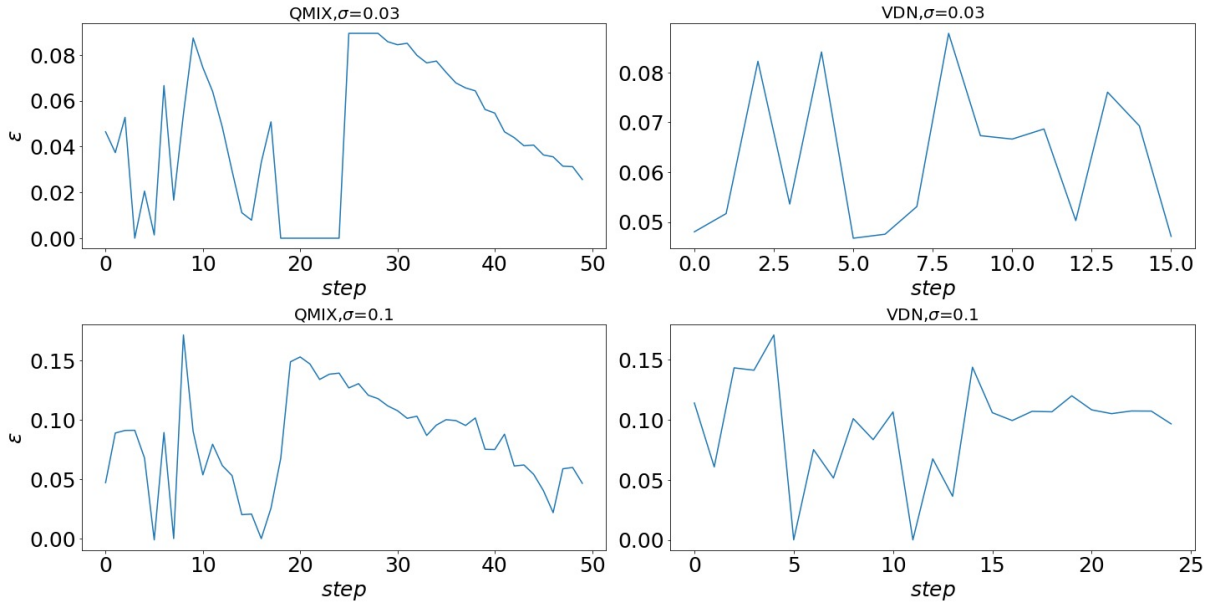


(b) Certified bound of perturbation for actions of each state

Figure 5.4: Certified robustness bound of perturbation in *Checkers*.

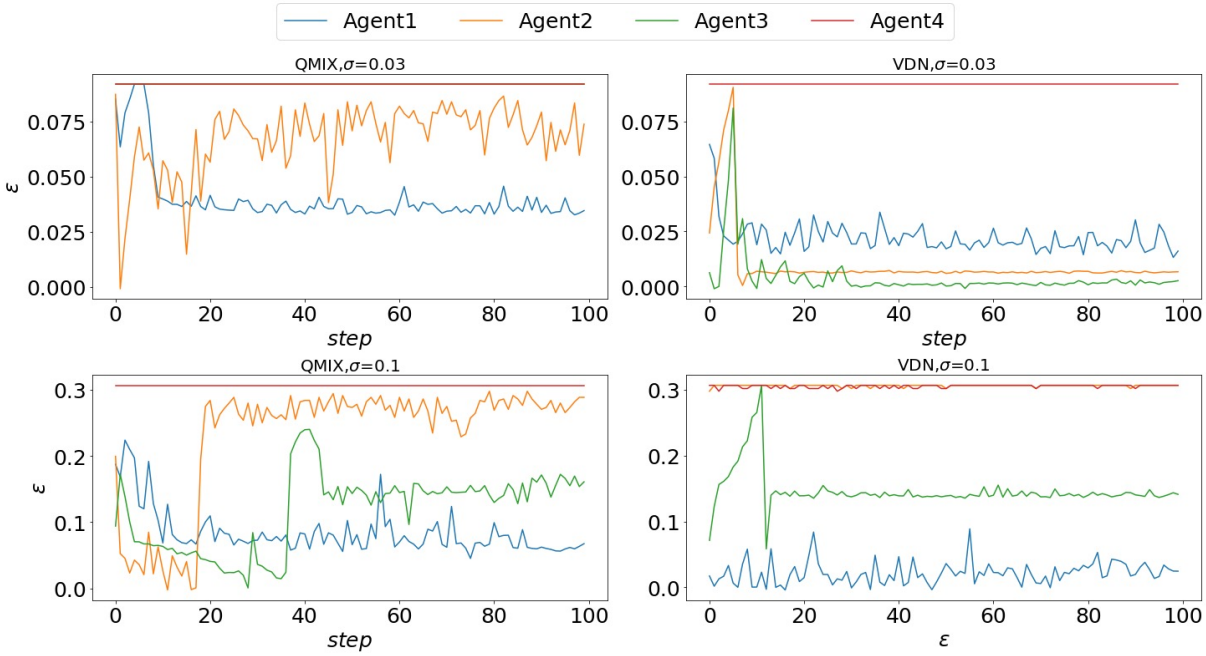


(a) Per-state certified bound of perturbation for per-state action of each agent

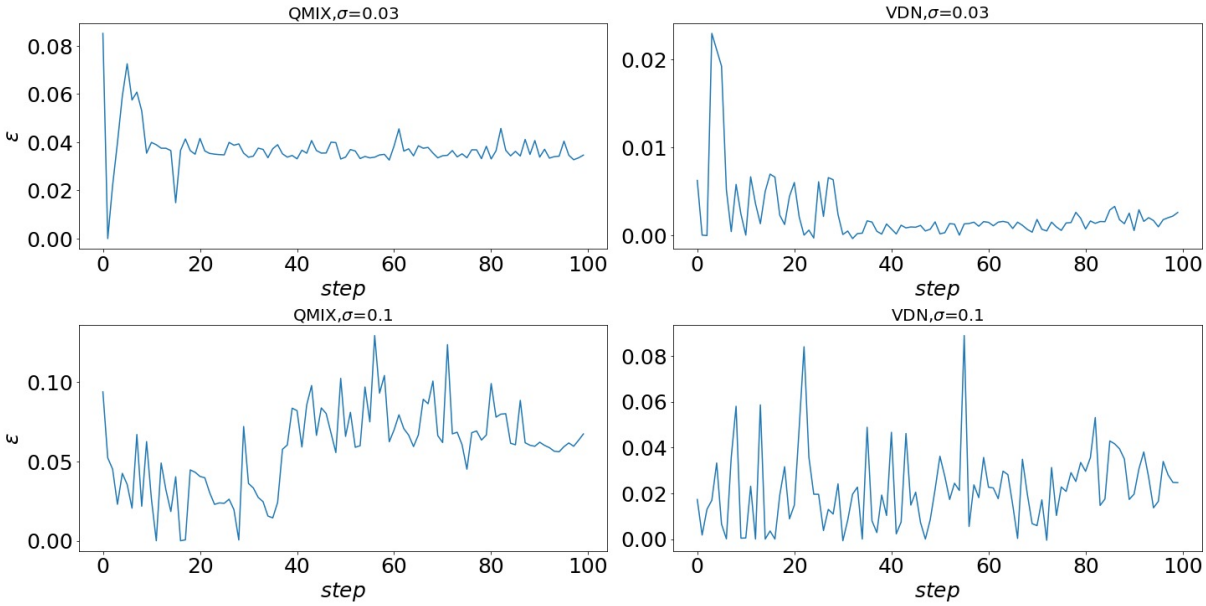


(b) Certified bound of perturbation for actions of each state

Figure 5.5: Per-state certified robustness bound of perturbation in *Switch*.

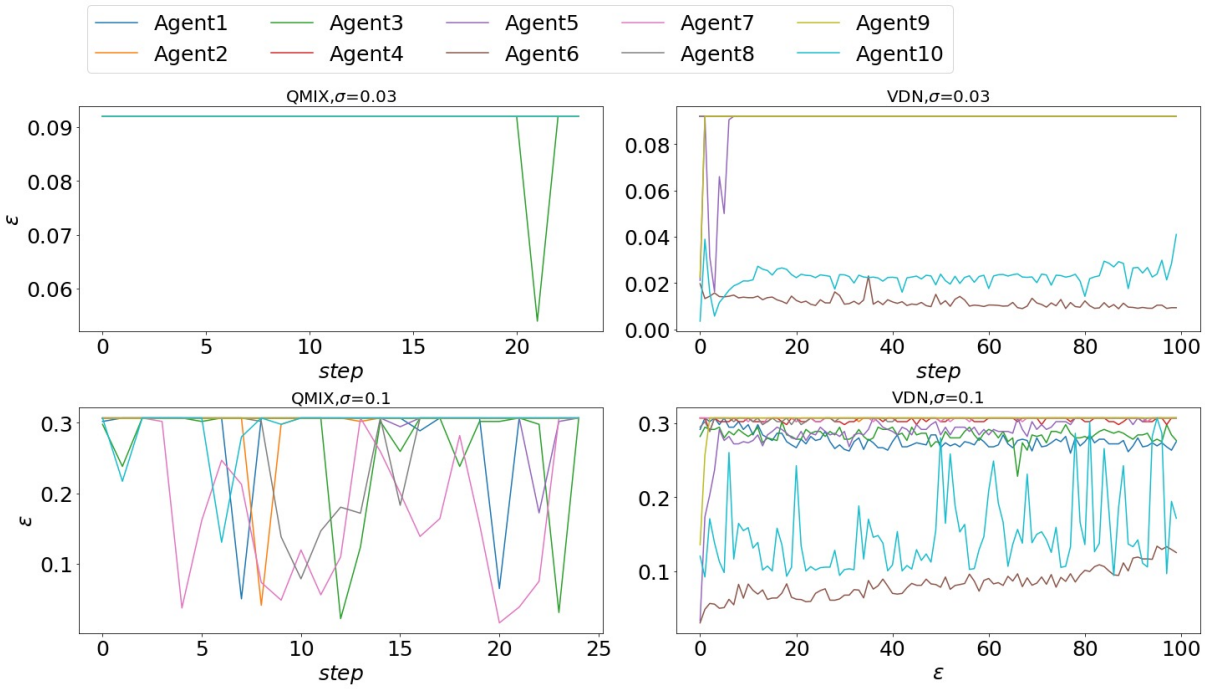


(a) Per-state certified bound of perturbation for per-state action of each agent

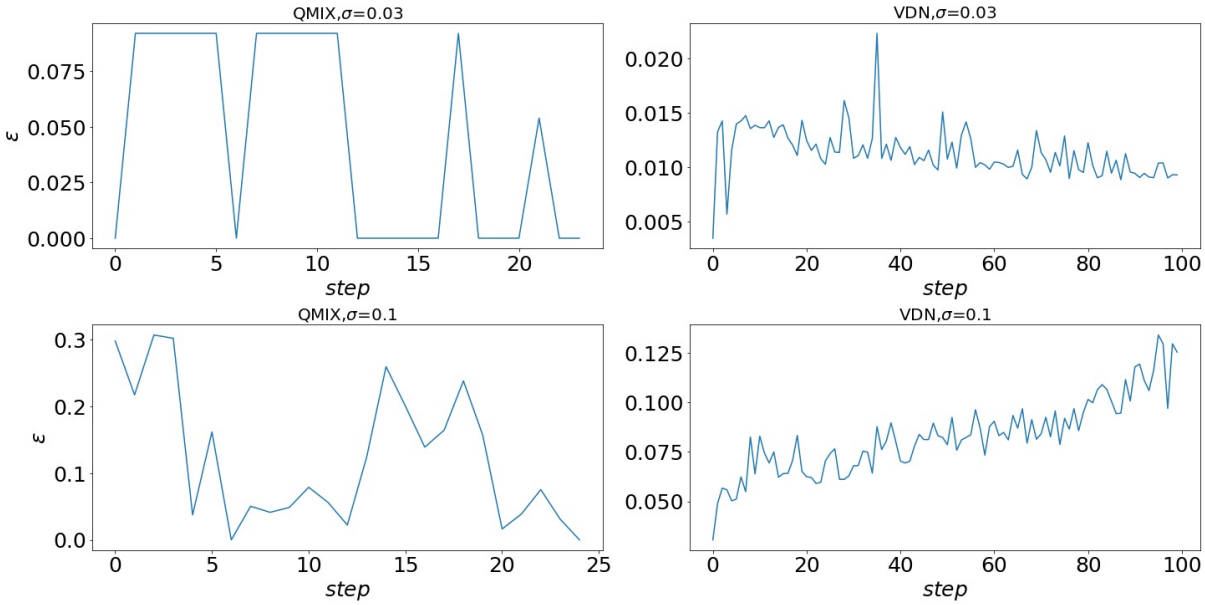


(b) Certified bound of perturbation for actions of each state

Figure 5.6: Per-state Certified bound of perturbation in *TrafficJunction* with four agents.



(a) Per-state certified bound of perturbation for per-state action of each agent



(b) Certified bound of perturbation for actions of each state

Figure 5.7: Per-state certified bound of perturbation in *TrafficJunction* with ten agents.

agents, the certified bound for each agent (trained by QMIX) is close to each other when the smoothing variance σ is 0.03. When we increase the variance to 0.1, Agent2 engages a slightly higher bound than Agent1, which means that Agent2 is more robust. For the agents trained by VDN, Agent2 always has a much higher robustness bound than Agent1. It may be because when training QMIX, all agents are expected to learn useful strategies, while VDN only needs some agents to learn well, and others may use lazier strategies, which results in a big divergence in the robustness between agents of VDN. In Switch with four agents, we observe that, by applying our p-value corrected method, the locally certified bound at each step will not always take the minimum bound among all agents and ignore the bound of agents with low impact.

5.6 Conclusion

We propose the first robustness certification solution for c-MARLs. By combining the FDR-controlling strategy with the importance factor of each agent, we certify the actions for each state while mitigating the multiple testing problem. In addition, a tree-search-based algorithm is applied to obtain a lower bound of the global reward. Our method is also applicable to single-agent RL systems, where it can obtain tighter bounds than the state-of-the-art certification methods. Furthermore, experiments were conducted on various multi-agent reinforcement environments, ranging from 2 to 10 agents, to demonstrate the performance of the proposed verification algorithms. These experiments provide evidence that the proposed method can effectively handle the challenges presented in verifying the robustness of multi-agent reinforcement learning and can achieve superior performance.

Chapter 6

Conclusions and Future Work

"AI is a rare case where I think we need to be proactive in regulation instead of reactive because I think by the time we are reactive in AI regulation, it's too late."
(Elon Musk)

In conclusion, this thesis makes a significant contribution to addressing the critical issue of the vulnerability of deep neural networks (DNNs) to adversarial attacks, which has raised serious concerns about their suitability in safety-critical applications. Through our research, we propose novel and effective approaches to assess the robustness of DNNs in three critical aspects: adversarial attacks, adversarial training, and robustness verification.

In Chapter 3, we introduce DeepSAVA, a sparse adversarial attack approach for videos, and a novel adversarial training framework based on the perturbations generated by DeepSAVA. Our proposed approach demonstrates superior performance in terms of attack success rate and adversarial transferability. In Chapter 4, we propose an efficient verification framework, 3DVerifier, to address the challenges of robustness verification in 3D point cloud models. Our approach provides tighter certification bounds than existing state-of-the-art solutions. In Chapter 5, we propose a novel certification method for cooperative multi-agent reinforcement learning models (c-MARLs) to tackle the challenges of robustness verification in this domain. Overall, our research contributes to the development of robust and trustworthy DNNs, which are essential for the successful deployment of DNNs in safety-critical applications. Our proposed approaches provide an effective means of assessing and enhancing the robustness of DNNs against adversarial attacks, which is a critical requirement for ensuring the safety and reliability of DNNs in real-world applications.

6.1 Thesis Summary

This thesis aims to assess the robustness of different DNNs, such as video recognition models, 3D point clouds classifiers, and multi-agent reinforcement learning algorithms. Each chapter is summarised as follows:

- (Chapter 3) **Sparse Adversarial Video Attacks and Defences** [97] In this chapter, we focus on performing an effective sparse attack, DeepSAVA, on videos and also proposed an adversarial training method to improve the robustness of the model against various perturbations. As the sequential structure of videos, to achieve a very sparse, we propose to use the Bayesian Optimisation (BO) algorithm to determine the most critical frame and then attack it, which could improve the efficiency of the sparse attack. Additionally, considering in the real world, the perturbation is more complex, we combine the additive perturbation with spacial transformation perturbation to perturb the chosen frame, which makes DeepSAVA a stronger and more general attack framework. we also find that the most commonly used l_p-norm metric in previous work is not the most effective measurement metric for the spacial transformation perturbation, which can be replaced by the structural similarity index measure (SSIM). Overall, DeepSAVA outperforms the state-of-the-art baselines in terms of both fooling rate and transferability. Significantly improvements are achieved for the I3D model, by only attacking one frame of the video to obtain a high attack success rate. What's more,

we also perform an effective adversarial training method, which is more effective than the state-of-the-art adversarial training algorithm based on PGD.

- (Chapter 4) **Efficient Robustness Verification for 3D Point Cloud Models**[95] we study the robustness verification on 3D point cloud models. As no existing work can handle verifying the complete 3D point clouds models, PointNet, in this chapter, we propose a more effective and general verification framework, which is also the first work to handle the complete PointNet. The main challenge of verifying the 3D point cloud models is the multiplication layers in the JANet block, which contains cross-non-linearity operation. To deal tackle this problem, we proposed to use of a linear relaxation function to bound the multiplication layer. Also, the point cloud models engage a high-dimensional input, which leads to a high computational complexity problem. our solution is to combine the forward and backward propagation process to compute the bound and improved global max pooling relaxation to reduce the computational complexity. We take extensive experiments on various models and point-cloud datasets to demonstrate the computational efficiency and tightness of the certified bounds.

- (Chapter 5)**Certified Policy Smoothing for Cooperative Multi-Agent Reinforcement Learning** [96] Verification methods for reinforcement learning have primarily focused on single agent systems, leaving the verification of cooperative multi-agent reinforcement learning (c-MARL) largely unexplored. This chapter presents a novel framework for verifying the robustness of multi-agent learning in terms of each-state actions and global reward. The proposed framework addresses two significant challenges in verifying the robustness of multi-agent reinforcement learning. The first challenge arises from the fact that the team reward is determined by a group of agents, which may lead to different agents contributing differently to the team reward. The second challenge stems from the increasing uncertainty of certified robustness bounds as the number of agents increases. To address these challenges, the proposed framework first verifies each agent at each step and estimates the bound for all agents per step. We propose to consider the importance of each agent when estimating the per-state bound for all agents. An important factor is assigned to each agent, which can be applied to determine the bound for all agents. This solution helps overcome the first challenge by accounting for the contribution of each agent to the team reward. For the second challenge, we adapted the false discovery rate (FDR) to control the selective *type I error*. It should be noted that the proposed method is not limited to multi-agent systems and can also be applied to single-agent systems. In order to showcase the effectiveness of the proposed approach, experiments were conducted to compare it with state-of-the-art verification frameworks on single-agent reinforcement learning. The experimental results confirm that our method can provide tighter bounds and is more efficient than the baseline. Also, we take experiments on various multi-agent

reinforcement environments with 2,4, and 10 agents to show the performance of the proposed verification algorithms.

6.2 Unsolved Challenges and Future Work

Although we have solved many challenges in the previous study, there are still unsolved and unexplored challenges for future explorations. Below we introduce some unsolved challenges in our previous study and we also present more fields we can explore in the future.

- **Adversarial Training** In Chapter 3, we present the adversarial defence based on adversarial training to improve the robustness of video models. We reduce the fooling rate by 24.7% for the CNN+LSTM model at the most, and there is still room for improvement. As we can see from Table 3.11, although our combined perturbation adversarial training can obtain a much lower fooling rate than traditional PGD adversary training, a subtle alteration in clean model accuracy is introduced by our algorithm. When conducting the experiments, we also tried the Trades [176], which can achieve a trade-off between model accuracy and robustness, but it shows worse performance on video classification models. Thus, in the future, how to design an effective trade-off algorithm for video classification models can be explored.

- **Black box** In Chapter 3, we adopt the Bayesian Optimisation algorithm to select the most critical frame, but it still requires the output score to estimate the performance of the selected frame. Therefore, how to make it a black-box attack is still an open problem.

- **Computation Cost** In Chapter 4, we already proposed some algorithms to solve the high computation cost problem in verifying the 3D point cloud model, but because of the high dimensional input data and complex structure of the model, for some large neural networks, as we can see from Table 4.2, it still cost a long time to run one epoch. There is still room to explore that how to reduce the time cost when verifying large neural networks.

- **Other Perturbation** In Chapter 5, we mainly studied the perturbation on the observation of each agent. How the proposed verification framework can be applied to other perturbations, like reward perturbation, action perturbation, and state perturbation can also be considered in the future.

- **Verify Poisoning Attack on Multi-Agent Reinforcement Learning** To execute a poisoning attack, one would alter the inputs to include perturbed samples in the training dataset. In the case of c-MARLs, the attack could target observation, action, or reward. The objective of verification would be to ensure that the reward or action

remains unchanged under the perturbation added to the observation within a specific region.

- **Real-world Scenes Applications** As we suggested in this thesis, assessing the robustness of deep neural networks is urgent in safety-critical scenes like autonomous driving systems. Therefore, in the future, we will apply proposed approaches to some real-world safety-critical applications, like passenger detecting and monitoring self-driving cars. We start with applying it to some robotics systems or simulation software.

- **Out-of-Distribution (OOD) Problems** The out-of-distribution sample is the sample that is significantly different from the samples in the training dataset. Therefore, the model could give the wrong prediction for these ”unseen” data, which can cause serious consequences, especially in safety-critical scenes. Some works have developed the technology to detect the OOD samples or mitigate the impact of these OOD samples to improve the robustness of DNNs. This can be a future direction to be explored.

Appendix A

Configuration of PointNet models

Below are the specific layer configurations for 13-layer full PointNet with JANet and PointNets without JANet.

No	Type	no. features	normalization	activation function
1	Conv1D	32	BatchNormalization	ReLU
2	Conv1D	64	BatchNormalization	ReLU
3	Conv1D	512	BatchNormalization	ReLU
4	GlobalMaxpooling			
5	Dense	256	BatchNormalization	ReLU
6	Dense	512	BatchNormalization	ReLU
	Reshape			
	Multiplication			

Table A.1: Configuration for T-Net.

No	Type	no. features	normalization	activation function
	T-Net			
7	Conv1D	32	BatchNormalization	ReLU
8	Conv1D	64	BatchNormalization	ReLU
9	Conv1D	512	BatchNormalization	ReLU
10	GlobalMaxpooling			
11	Dense	512	BatchNormalization	ReLU
12	Dense	256	BatchNormalization	ReLU
	Dropout			

Table A.2: Configuration for full PointNet with JANet.

No	Type	no. features	normalization	activation function
1	Conv1D	32	BatchNormalization	ReLU
2	Conv1D	32	BatchNormalization	ReLU
3	Conv1D	64	BatchNormalization	ReLU
4	Conv1D	512	BatchNormalization	ReLU
5	GlobalMaxpooling			
6	Dense	512	BatchNormalization	ReLU
7	Dense	256	BatchNormalization	ReLU
	Dropout			

Table A.3: Configuration for 7-layer PointNet without JANet.

No	Type	no. features	normalization	activation function
1	Conv1D	32	BatchNormalization	ReLU
2	Conv1D	32	BatchNormalization	ReLU
3	Conv1D	64	BatchNormalization	ReLU
4	Conv1D	64	BatchNormalization	ReLU
5	Conv1D	512	BatchNormalization	ReLU
6	Conv1D	512	BatchNormalization	ReLU
7	GlobalMaxpooling			
8	Dense	512	BatchNormalization	ReLU
9	Dense	256	BatchNormalization	ReLU
10	Dense	256	BatchNormalization	ReLU
11	Dense	128	BatchNormalization	ReLU
12	Dense	128	BatchNormalization	ReLU
	Dropout			

Table A.4: Configuration for 12-layer PointNet without JANet.

References

[1] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 7163–7172.

[2] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*, PMLR, 2018, pp. 274–283.

[3] C. Badue, R. Guidolini, R. V. Carneiro, *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, p. 113 816, 2020.

[4] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*, Springer, 2017, pp. 262–275.

[5] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal statistical society: series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[6] N. Berthier, Y. Sun, W. Huang, Y. Zhang, W. Ruan, and X. Huang, "Tutorials on testing neural networks," *arXiv preprint arXiv:2108.01734*, 2021.

[7] A. Boopathy, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, *Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks*, 2018. arXiv: 1811.12395 [stat.ML].

[8] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Open-AI gym," *arXiv preprint arXiv:1606.01540*, 2016.

[9] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018.

[10] R. R. Bunel, I. Turkaslan, P. Torr, P. Kohli, and P. K. Mudigonda, "A unified view of piecewise linear neural network verification," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 4795–4804.

[11] Y. Cai, C. Zhang, W. Shen, X. Zhang, W. Ruan, and L. Huang, "Reprem: Representation pre-training with masked model for reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'23)*, 2023.

[12] Y. Cao, C. Xiao, D. Yang, *et al.*, "Adversarial objects against lidar-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019.

[13] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57. DOI: 10.1109/SP.2017.49.

[14] N. Carlini and D. Wagner, *Towards Evaluating the Robustness of Neural Networks*, 2016. arXiv: 1608.04644 [cs.CR].

[15] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 3–14.

[16] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[17] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.

[18] X. Chen, K. Jiang, Y. Zhu, X. Wang, and T. Yun, "Individual tree crown segmentation directly from uav-borne lidar data using the pointnet of deep learning," *Forests*, vol. 12, no. 2, p. 131, 2021.

[19] Z. Chen, L. Xie, S. Pang, Y. He, and Q. Tian, "Appending Adversarial Frames for Universal Video Attack," *arXiv e-prints*, 2019. arXiv: arXiv:1912.04538 [cs.CV].

[20] Chen, Siheng and Liu, Baoan and Feng, Chen and Vallespi-Gonzalez, Carlos and Wellington, Carl, "3D Point Cloud Processing and Learning for Autonomous Driving: Impacting Map Creation, Localization, and Perception," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 68–86, 2021. DOI: 10.1109/MSP.2020.2984780.

[21] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: https://www.aclweb.org/anthology/D14-1179.

[22] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[23] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 1934.

[24] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*, PMLR, 2019, pp. 1310–1320.

[25] Y. Deng, X. Zheng, T. Zhang, C. Chen, G. Lou, and M. Kim, "An analysis of adversarial attacks and defenses on autonomous driving models," in *2020 IEEE international conference on pervasive computing and communications (PerCom)*, IEEE, 2020, pp. 1–10.

[26] J. Donahue, L. A. Hendricks, M. Rohrbach, *et al.*, *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*, 2016. arXiv: 1411.4389 [cs.CV].

[27] P. L. Donti, M. Roderick, M. Fazlyab, and J. Z. Kolter, "Enforcing robust control guarantees within neural network policies," *arXiv preprint arXiv:2011.08105*, 2020.

[28] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks.," in *UAI*, vol. 1, 2018, p. 3.

[29] K. D. Dvijotham, R. Stanforth, S. Gowal, C. Qin, S. De, and P. Kohli, "Efficient neural network verification with exactness characterization," in *Uncertainty in artificial intelligence*, PMLR, 2020, pp. 497–507.

[30] B. Eysenbach and S. Levine, "Maximum entropy rl (provably) solves some robust rl problems," *arXiv preprint arXiv:2103.06257*, 2021.

[31] D.-P. Fan, W. Wang, M.-M. Cheng, and J. Shen, "Shifting more attention to video salient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8554–8564.

[32] Y. Fang, Z. Wang, W. Lin, and Z. Fang, "Video saliency incorporating spatiotemporal cues and uncertainty weighting," *IEEE transactions on image processing*, vol. 23, no. 9, pp. 3910–3921, 2014.

[33] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[34] S. A. Fezza, Y. Bakhti, W. Hamidouche, and O. Déforges, "Perceptual Evaluation of Adversarial Attacks for CNN-based Image Classification," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, 2019, pp. 1–6. DOI: 10.1109/QoMEX.2019.8743213.

[35] W. Fithian, D. Sun, and J. Taylor, "Optimal inference after model selection," *arXiv preprint arXiv:1410.2597*, 2014.

[36] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[37] D. Fohr, O. Mella, and I. Illina, "New Paradigm in Speech Recognition: Deep Neural Networks," in *proceedings of IEEE International Conference on Information Systems and Economic Intelligence*, Marrakech, Morocco, Apr. 2017. [Online]. Available: `https://hal.archives-ouvertes.fr/hal-01484447`.

[38] I. Fortran, W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, "Numerical Recipes," *Cambridge, UK, Cambridge University Press*, Jan. 1992.

[39] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE symposium on security and privacy (SP)*, IEEE, 2018, pp. 3–18.

[40] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *arXiv preprint arXiv:1905.10615*, 2019.

[41] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv e-prints*, 2014. arXiv: `arXiv:1412.6572 [stat.ML]`.

[42] L. A. Goodman, "On simultaneous confidence intervals for multinomial proportions," *Technometrics*, vol. 7, no. 2, pp. 247–254, 1965.

[43] D. Gragnaniello, F. Marra, L. Verdoliva, and G. Poggi, "Perceptual quality-preserving black-box attack against deep learning image classifiers," *Pattern Recognition Letters*, vol. 147, pp. 142–149, 2021.

[44] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018.

[45] J. Hayes, "Extensions and limitations of randomized smoothing for robustness guarantees," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 786–787.

[46] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defense: Ensembles of weak defenses are not strong," in *11th USENIX workshop on offensive technologies (WOOT 17)*, 2017.

[47] S. Hershey, S. Chaudhuri, D. P. Ellis, *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, IEEE, 2017, pp. 131–135.

[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[51] X. Huang, D. Kroening, W. Ruan, *et al.*, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100 270, 2020.

[52] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[53] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang, "Black-box adversarial attacks on video recognition models," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 864–872.

[54] G. Jin, X. Yi, W. Huang, S. Schewe, and X. Huang, "Enhancing adversarial training with second-order statistics of weights," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 273–15 283.

[55] G. Jin, X. Yi, D. Wu, R. Mu, and X. Huang, *Randomized adversarial training via taylor expansion*, 2023. arXiv: 2303.10653 [cs.LG].

[56] G. Jin, X. Yi, P. Yang, L. Zhang, S. Schewe, and X. Huang, *Weight expansion: A new perspective on dropout and generalization*, 2022. arXiv: 2201.09209 [cs.LG].

[57] G. Jin, X. Yi, L. Zhang, L. Zhang, S. Schewe, and X. Huang, *How does weight correlation affect the generalisation ability of deep neural networks*, 2020. arXiv: 2010.05983 [cs.LG].

[58] M. Jordan, N. Manoj, S. Goel, and A. G. Dimakis, "Quantifying Perceptual Distortion of Adversarial Examples," *arXiv e-prints*, 2019. arXiv: arXiv:1902.08265 [stat.ML].

[59] A. Joshi, *Next-frame video prediction with convolutional lstms*, https://keras.io/examples/vision/conv_lstm/, 2021.

[60] M. E. Kalfaoglu, S. Kalkan, and A. A. Alatan, "Late Temporal Modeling in 3D CNN Architectures with BERT for Action Recognition," in *arXiv e-prints*, 2020. arXiv: arXiv:2008.01232 [cs.CV].

[61] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.

[62] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223.

[63] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International conference on computer aided verification*, Springer, 2017, pp. 97–117.

[64] F. A. Al-Khayyal and J. E. Falk, "Jointly constrained biconvex programming," *Mathematics of Operations Research*, vol. 8, no. 2, pp. 273–286, 1983.

[65] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *In proceedings of CoRR*, vol. abs/1412.6980, 2015.

[66] Z. Kolter and A. Madry, *Adversarial robustness - theory and practice*, https://adversarial-ml-tutorial.org/introduction/, 2019.

[67] Y. Kong and Y. Fu, *Human Action Recognition and Prediction: A Survey*, 2018. arXiv: 1806.11230 [cs.CV].

[68] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv preprint arXiv:1705.06452*, 2017.

[69] R. Kotikalapudi and contributors, *Keras-vis*, https://github.com/raghakot/keras-vis, 2017.

[70] A. Koul, *Ma-gym: Collection of multi-agent environments based on openai gym.* https://github.com/koulanurag/ma-gym, 2019.

[71] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, 2016.

[72] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 2556–2563.

[73] A. Kumar, A. Levine, and S. Feizi, "Policy smoothing for provably robust reinforcement learning," *arXiv preprint arXiv:2106.11420*, 2021.

[74] R. Kumar, *Video predictions using transformer*, 2021. [Online]. Available: https://github.com/iamrakesh28/Video-Prediction.

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

[76] C. Laidlaw and S. Feizi, "Functional Adversarial Attacks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019, pp. 10 408–10 418. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/6e923226e43cd6fac7cfe1e13ad000ac-Paper.pdf.

[77] C. Laidlaw, S. Singla, and S. Feizi, "Perceptual adversarial robustness: Defense against unseen threat models," in *International Conference on Learning Representations*, 2020.

[78] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," *arXiv preprint arXiv:1603.03827*, 2016.

[79] K. Lee, Z. Chen, X. Yan, R. Urtasun, and E. Yumer, "Shapeadv: Generating shape-aware adversarial 3d point clouds," *arXiv preprint arXiv:2005.11626*, 2020.

[80] B. Li, S. Wang, S. Jana, and L. Carin, "Towards understanding fast adversarial training," *arXiv preprint arXiv:2006.03089*, 2020.

[81] S. Li, A. Neupane, S. Paul, *et al.*, "Stealthy Adversarial Perturbations Against Real-Time Video Classification Systems," *In Proceedings of 2019 Network and Distributed System Security Symposium*, 2019. DOI: 10.14722/ndss.2019.23202. [Online]. Available: http://dx.doi.org/10.14722/ndss.2019.23202.

[82] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 641–656.

[83] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," in *2020 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2020, pp. 62–68.

[84] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," *arXiv preprint arXiv:1703.06748*, 2017.

[85] D. Liu, R. Yu, and H. Su, "Extending adversarial attacks and defenses to deep 3d point cloud classifiers," in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 2279–2283.

[86] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into Transferable Adversarial Examples and Black-box Attacks," *In proceedings of ICLR*, 2017.

[87] T. Lorenz, A. Ruoss, M. Balunović, G. Singh, and M. Vechev, "Robustness certification for point cloud models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7608–7618.

[88] W. Lotter, G. Kreiman, and D. Cox, *Deep predictive coding networks for video prediction and unsupervised learning*, 2017. arXiv: 1605.08104 [cs.LG].

[89] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *Conference on Robot Learning*, PMLR, 2020, pp. 1328–1337.

[90] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[91] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[92] V. Mittal, D. Gangodkar, and B. Pant, "Exploring The Dimension of DNN Techniques For Text Categorization Using NLP," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 497–501. DOI: 10.1109/ICACCS48705.2020.9074228.

[93] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[95] R. Mu, W. Ruan, L. S. Marcolino, and Q. Ni, "3dverifier: Efficient robustness verification for 3d point cloud models," *Machine Learning*, pp. 1–28, 2022.

[96] R. Mu, W. Ruan, L. S. Marcolino, G. Jin, and Q. Ni, "Certified policy smoothing for cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:2212.11746*, 2022.

[97] R. Mu, W. Ruan, L. S. Marcolino, and Q. Ni, "Sparse adversarial video attacks with spatial transformations," *arXiv preprint arXiv:2111.05468*, 2021.

[98] I. Naeh, R. Pony, and S. Mannor, "Flickering Adverparsearial Attacks against Video Recognition Networks," *arXiv e-prints*, 2020. arXiv: arXiv:2002.05123 [cs.LG].

[99] J. Neyman and E. S. Pearson, "Ix. on the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, no. 694-706, pp. 289–337, 1933.

[100] A. Nguyen, J. Yosinski, and J. Clune, "Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[101] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

[102] A. Paigwar, O. Erkent, C. Wolf, and C. Laugier, "Attentional pointnet for 3d-object detection in point clouds," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[103] J. Pan, Y. Yin, J. Xiong, W. Luo, G. Gui, and H. Sari, "Deep learning-based unmanned surveillance systems for observing water levels," *In proceedings of IEEE Access*, vol. 6, pp. 73 561–73 571, 2018.

[104] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, *The Limitations of Deep Learning in Adversarial Settings*, 2015. arXiv: 1511.07528 [cs.CR].

[105] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*, IEEE, 2016, pp. 582–597.

[106] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," *arXiv preprint arXiv:1702.08360*, 2017.

[107] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated White box Testing of Deep Learning Systems," *In Proceedings of the 26th Symposium on Operating Systems Principles - SOSP '17*, 2017. DOI: 10.1145/3132747.3132785. [Online]. Available: http://dx.doi.org/10.1145/3132747.3132785.

[108] N. H. Pham, L. M. Nguyen, J. Chen, H. T. Lam, S. Das, and T.-W. Weng, "Evaluating robustness of cooperative marl: A model-based approach," *arXiv preprint arXiv:2202.03558*, 2022.

[109] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proceedings of Advances in neural information processing systems*, vol. 30, 2017, pp. 5105–5114.

[110] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, 2017. arXiv: 1612.00593 [cs.CV].

[111] C. Qin, B. O'Donoghue, R. Bunel, *et al.*, "Verification of non-linear specifications for neural networks," *arXiv preprint arXiv:1902.09592*, 2019.

[112] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[113] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4295–4304.

[114] W. Ruan, X. Yi, and X. Huang, "Adversarial robustness of deep learning: Theory, algorithms, and applications," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[115] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[116] A. Russo and A. Proutiere, "Optimal attacks on reinforcement learning policies," *arXiv preprint arXiv:1907.13548*, 2019.

[117] H. Salman, J. Li, I. Razenshteyn, *et al.*, "Provably robust deep learning via adversarially trained smoothed classifiers," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[118] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Curran Associates Inc., 2019.

[119] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks : the official journal of the International Neural Network Society*, vol. 61, pp. 85–117, 2015.

[120] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[121] D. Shen, G. Wu, and H.-I. Suk, "Deep Learning in Medical Image Analysis," *Annual Review of Biomedical Engineering*, vol. 19, no. 1, pp. 221–248, 2017, PMID: 28301734. DOI: 10.1146/annurev-bioeng-071516-044442. eprint: https://doi.org/10.1146/annurev-bioeng-071516-044442. [Online]. Available: https://doi.org/10.1146/annurev-bioeng-071516-044442.

[122] Q. Shen, Y. Li, H. Jiang, Z. Wang, and T. Zhao, "Deep reinforcement learning with robust and smooth policy," in *International Conference on Machine Learning*, PMLR, 2020, pp. 8707–8718.

[123] Z. Shi, H. Zhang, K.-W. Chang, M. Huang, and C.-J. Hsieh, "Robustness verification for transformers," *arXiv preprint arXiv:2002.06622*, 2020.

[124] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[125] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, 2014. arXiv: 1312.6034 [cs.CV].

[126] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[127] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," in *International Conference on Learning Representations*, 2018.

[128] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *ICLR (Poster)*, 2018.

[129] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," *arXiv e-prints*, 2012. arXiv: 1212.0402 [cs.CV].

[130] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[131] J. D. Stets, Y. Sun, W. Corning, and S. W. Greenwald, "Visualization and labeling of point clouds in virtual reality," in *SIGGRAPH Asia 2017 Posters*, 2017, pp. 1–2.

[132] Y. Sun, R. Zheng, P. Hassanzadeh, *et al.*, "Certifiably robust policy learning against adversarial communication in multi-agent systems," *arXiv preprint arXiv:2206.10158*, 2022.

[133] P. Sunehag, G. Lever, A. Gruslys, *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[134] C. Szegedy, W. Liu, Y. Jia, *et al.*, *Going Deeper with Convolutions*, 2014. arXiv: 1409.4842 [cs.CV].

[135] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[136] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199.

[137] T. Tanay and L. Griffin, "A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples," *arXiv e-prints*, 2016. arXiv: arXiv:1608.07690 [cs.LG].

[138] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.

[139] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1633–1645, 2020.

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

[141] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *Learning Spatiotemporal Features with 3D Convolutional Networks*, 2015. arXiv: `1412.0767 [cs.CV]`.

[142] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of learning, inference and control of multi-agent systems (at NIPS 2016)*, vol. 1, 2016.

[143] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2017, pp. 2442–2447.

[144] F. Wang, C. Zhang, P. Xu, and W. Ruan, "Deep learning and its adversarial robustness: A brief introduction," in *HANDBOOK ON COMPUTER LEARNING AND INTELLIGENCE: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation*, World Scientific, 2022, pp. 547–584.

[145] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Proceedings of Neural Information Processing Systems*, 2018, pp. 6369–6379.

[146] Z. Wang and A. C. Bovik, "A universal image quality index," *In proceedings of IEEE signal processing letters*, vol. 9, no. 3, pp. 81–84, 2002.

[147] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *In proceedings of IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[148] X. Wei, J. Zhu, S. Yuan, and H. Su, "Sparse adversarial perturbations for videos," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8973–8980.

[149] Z. Wei, J. Chen, X. Wei, *et al.*, "Heuristic Black-Box Adversarial Attacks on Video Recognition Models," in *proceedings of AAAI*, 2020, pp. 12 338–12 345.

[150] Y. Wen, J. Lin, K. Chen, C. P. Chen, and K. Jia, "Geometry-aware generation of adversarial point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[151] L. Weng, H. Zhang, H. Chen, *et al.*, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5276–5285.

[152] T.-W. Weng, K. D. Dvijotham, J. Uesato, *et al.*, "Toward evaluating robustness of deep reinforcement learning with continuous control," in *International Conference on Learning Representations*, 2019.

[153] D. Whitley, "A Genetic Algorithm Tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, 1994.

[154] M. Wicker and M. Kwiatkowska, "Robustness of 3d deep learning in an adversarial setting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 767–11 775.

[155] R. Wiyatno and A. Xu, "Maximal jacobian-based saliency map attack," *arXiv preprint arXiv:1808.07945*, 2018.

[156] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5286–5295.

[157] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations*, 2020.

[158] E. Wong, F. R. Schmidt, and J. Z. Kolter, "Wasserstein Adversarial Examples via Projected Sinkhorn Iterations," *arXiv e-prints*, 2020. arXiv: `arXiv:1902.07906 [cs.LG]`.

[159] F. Wu, L. Li, Z. Huang, Y. Vorobeychik, D. Zhao, and B. Li, "Crop: Certifying robust policies for reinforcement learning through functional smoothing," *arXiv preprint arXiv:2106.09292*, 2021.

[160] H. Wu and W. Ruan, "Adversarial driving: Attacking end-to-end autonomous driving systems," *arXiv preprint arXiv:2103.09151*, 2021.

[161] Z. Wu, S. Song, A. Khosla, *et al.*, "3D shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[162] C. Xiang, C. R. Qi, and B. Li, *Generating 3d adversarial point clouds*, 2019. arXiv: `1809.07016 [cs.CR]`.

[163] C. Xiao, P. Zhong, and C. Zheng, "Enhancing adversarial defense by k-winners-take-all," *arXiv preprint arXiv:1905.10510*, 2019.

[164] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *In proceedings of IJCAI*, 2018.

[165] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially Transformed Adversarial Examples," in *International Conference on Learning Representations*, 2018. [Online]. Available: `https://openreview.net/forum?id=HyydRMZC-`.

[166] P. Xu, W. Ruan, and X. Huang, "Quantifying safety risks of deep neural networks," *Complex & Intelligent Systems*, pp. 1–18, 2022.

[167] H. Yan, X. Wei, and B. Li, "Sparse Black-box Video Attack with Reinforcement Learning," *arXiv e-prints*, 2020. arXiv: `arXiv:2001.03754 [cs.CV]`.

[168] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3166–3173.

[169] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[170] J. Yang, Q. Zhang, R. Fang, B. Ni, J. Liu, and Q. Tian, "Adversarial attack and defense on point sets," *arXiv preprint arXiv:1902.10899*, 2019.

[171] D. Ye, M. Zhang, and Y. Yang, "A multi-agent framework for packet routing in wireless sensor networks," *Sensors*, vol. 15, no. 5, pp. 10 026–10 047, 2015.

[172] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[173] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[174] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, Springer, 2014, pp. 818–833.

[175] H. Zhang and J. Wang, "Towards adversarially robust object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 421–430.

[176] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International conference on machine learning*, PMLR, 2019, pp. 7472–7482.

[177] H. Zhang, H. Chen, C. Xiao, B. Li, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on observations," *arXiv preprint arXiv:2003.08938*, 2020.

[178] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Proceedings of Neural Information Processing Systems*, 2018, pp. 4944–4953.

[179] S. Q. Zhang, Q. Zhang, and J. Lin, "Efficient communication in multi-agent reinforcement learning via variance based control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[180] Y. Zhang, W. Ruan, F. Wang, and X. Huang, "Generalizing universal adversarial attacks beyond additive perturbations," in *2020 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2020, pp. 1412–1417.

[181] Y. Zhang, G. Liang, T. Salem, and N. Jacobs, "Defense-pointnet: Protecting pointnet against adversarial attacks," in *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 5654–5660.

[182] Y. Zhao, Y. Wu, C. Chen, and A. Lim, *On isometry robustness of deep 3d point cloud models under adversarial attacks*, 2020. arXiv: 2002.12222 [cs.LG].

[183] Z. Zhao, Z. Liu, and M. Larson, "Towards large yet imperceptible adversarial image perturbations with perceptual color distance," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1039–1048.

[184] H. Zhou, K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu, "Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1961–1970.

[185] Zhou Wang and A. C. Bovik, "Mean squared error: Love it or Leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009. DOI: 10.1109/MSP.2008.930649.