CRANFIELD UNIVERSITY

CHEN LI

SWARM DRONES - EFFICIENT MACHINE LEARNING AND INFORMATICS

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Aerospace

PhD
Academic Year: 2019–2022

Supervisors: Prof. Weisi Guo, Prof. Antonios Tsourdos
December 2022

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Aerospace

PhD

Academic Year: 2019–2022

CHEN LI

Swarm Drones - Efficient Machine Learning and Informatics

Supervisors: Prof. Weisi Guo, Prof. Antonios Tsourdos
December 2022

This thesis is submitted in partial fulfilment of the requirements for the degree of PhD.

Abstract

In 2020, worldwide consumer drone unit shipment was 5 million, which is expected to be 9.6 million in 2030. This generates a global drone market with 26.3 billion USD in revenue. The popularity of drones in civilian and professional environments has changed the way humans live and work. However, they have also brought new challenges and threats to the environment and society (e.g., property and personal damage caused by inappropriate drone operations or hostile drones) which requires more robust regulatory mechanisms and advanced technologies of drones. Supervision of tiny shape, high-mobility drones by humans is inefficient and inaccurate, whereas Artificial Intelligence (AI) methods, especially deep learning (DL) show potential in drone detection, classification and tracking. However, as data-driven models, the performance of DL models is decided by the quality of data and model structure. At the same time, the structural complexity of black-box DL models affects their explainability and energy efficiency. These factors affect the willingness of people to trust DL models. Therefore, this thesis aims to analyze the impact of drone informatics on DL behaviour, and achieve more efficient training of high-trustworthiness DL models by efficient drone informatics. This will require research on DL explainability, trustworthiness and efficiency. The aforementioned researches are all interrelated and highly relevant to the data.

In this thesis, firstly, explainable AI (XAI) and DL trust factors are reviewed. A theoretical DL trustworthiness metric Quality of Trust (QoT) and a lifelong AI trustworthiness supervision protocol are proposed. Secondly, a novel partially explainable Gaussian-process-based neural network structure is proposed. Compared with conventional machine learning methods, it is more transparent and without any sacrifice in ac-

curacy. Thirdly, a GAN-TDA method is proposed to analyze the learning efficiency of convolutional layers on drone images and guide the collection of new data. Collecting new data with direction could boost the DL model performance more efficiently in time and cost. Fourthly, a transistor operations (TOs) model is proposed to analyze the DL energy consumption scaling law to different model architectures and settings. Finally, a physical visual neural stealth drone canopy is designed with the hard-to-learn design features analyzed by GAN-TDA and painted with adversarial evasion features to escape DL drone detection and classification. The canopy design method is further extended to swarm drone scenarios.

This thesis shows: 1) both model explainability and performance are related to DL trustworthiness, and need a trade-off according to the QoT of different tasks; 2) combining human-understandable efficient drone informatics and the understanding of DL energy scaling laws can find high-efficiency datasets and network structures, resulting in efficient DL models with high trustworthiness; 3) The above knowledge can be used to formulate attacks on drone-related DL models to reduce their trustworthiness.

Keywords

Artificial Intelligence; Deep Learning; Swarm Drones; Drone Informatics; DL Trustworthiness; DL Explainability; DL Learning Efficiency; DL Energy Efficiency; Neural Stealth; Evasion Features

Contents

List of Figures

List of Tables

List of Abbreviations

3D	Three Dimensions
5G	Fifth Generation Mobile Networks
6G	Sixth Generation Mobile Networks
AI	Artificial Intelligence
AF	Activation Function
ALU	Arithmetic Logic Units
BCE	Binary Cross-entropy
BO	Basic Operation
BP	BackPropagation
BS	Base Station
CBR	Case-Based Reasoning
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSI	Channel State Information
CV	Computer Vision
DCGAN	Deep Convolutional Generative Adversarial Network
DeepLIFT	Deep Learning Important FeaTures
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Tree
DQN	Deep-Q Network

(D)RAM	(Dynamic) Random-Access Memory
DRL	Deep Reinforcement Learning
eMBB	Enhanced Mobile Broadband
EU	European Union
FD	Feature Distribution
FLOPs	Floating-point Operations
FPGA	Field-Programmable Gate Array
FP	Floating-point Number
FPU	Floating-Point Unit
GAN	Generative Adversarial Network
GDP	Gross Domestic Product
GDPR	General Data Protection Regulation
GELU	Gaussian Error Linear Unit
GP	Gaussian Process
GPU	Graphical Processing Unit
IC	Integrated Circuit
ID	Identity Document
IoAT	Internet-of-Autonomous-Things
KPI	Key Performance Indicator
KST	Kolmogorov–Arnold Superposition Theorem
LNL	Linear and Non-Linear
LIME	Local Interpretable Model-Agnostic Explanations
MAC	Medium Access Control Layer / Multiply–accumulate Operation
MDS	Multidimensional Scaling
ML	Machine Learning
mMTC	Massive Machine Type Communications
MSE	Mean-square Error
NASA	National Aeronautics and Space Administration

NLP	Natural Language Processing
NN	Neural Network
No.	Number
PCA	Principal Component Analysis
PHY	Physical Layer
PMC	Performance Monitoring Counters
PR	Polynomial Regression
QoE	Quality-of-Experience
QoT	Quality-of-Trust
QoS	Quality-of-Service
RBF	Radial-basis Function
RGB	Red, Green, and Blue
RMSE	Root Mean Square Error
RQ	Rational Quadratic
RRM	Radio Resource Management
SIMD	Single Instruction Multiple Data
SO	Socially Optimal
SP-LIME	Sub-modular Pick Local Interpretable Model-Agnostic Explanations
SS	Static Slicing
SVM	Support Vector Machine
TDA	Topological Data Analysis
TO	Transistor Operation
TPU	Tensor Processing Unit
t-SNE	t-distributed Stochastic Neighbor Embedding
UAV	Unmanned Aerial Vehicle
uRLLC	Ultra-reliable Low Latency Communication
VAE	Variational Auto-encoder
XAI	Explainable Artificial Intelligence

XOR	exclusive OR

Acknowledgements

First of all, I would like to thank my supervisors, Professor Weisi Guo and Professor Antonios Tsourdos, for their help in my academic and life during my doctoral period. They always patiently guide my academic direction, and kindly share precious experiences with me. would also like to thank all my colleagues from Cranfield University AI Group and my PhD review team for their advice and accompany. Appreciate China Scholarship Council for funding my research. Special thanks to Chengyao Sun for his mathematical support in Chapter 4 and 5, and Xun Huang for 3D print works in Chapter 7.

I am also grateful to my lovely family, especially my parents. I would not have had the opportunity to embark on and complete this journey without their visionary advice and full support in every aspect of life. Special thanks to Grandpa Shi, Aunt Shi, and Uncle Shi from Shenzhen for their suggestions on my academic direction.

Lastly, I would mention all my friends who accompanied me during this period of time, especially Mengbang Zou, Yuchi Zhang, Cen Cui, Zirui Wang, Binkai Xia, and Shuai Zeng. Many thanks for all the entertainment and emotional support. Thanks to Dr. Bowen DU, Dr. Yujue Zhou, and Jiajun Zhang for their support and knowledge sharing.

Chapter 1

Introduction

In 2020, worldwide consumer drone unit shipment was 5 million, which is expected to be 9.6 million in 2030. This generates a global drone market with 26.3 billion USD in revenue [4, 5, 6]. With the deployment of drones in academia and industry, drones play more and more significant roles in data collection, transportation, information relay and environment monitoring. While high flexibility, small size and remote controllability grant drones the ability to process complex tasks, they also bring significantly raising of difficulties in the supervision of drones. The unauthenticated drone has been defined as low-slow-small (LSS) aerial threats in counter-unmanned aerial systems (C-UAS) [7, 8]. Currently, up-to-date C-UAS systems apply DL methods, especially convolutional neural networks (CNN) to assist vision-based drone-related tasks. CNNs have proven to be accurate and robust in detecting drones even when they are distant, low-flying, and co-exist in clutter [9, 8].

However, applying DL models in practical drone tasks still has some problems. Firstly, as black-box models, DL models are with low transparency, which blocks the understanding of model behaviour and reasoning. This low explainability and uncertainty in model behaviour reduce the human willingness to trust a DL model in critical tasks [10]. Secondly, the cost of maintaining DL models is also considerable [11]. As data-driven models, the performance of DL models significantly depends on the quality of training data.

However, the features of drones (e.g. high-flexibility, tiny shape) barrier the data collection, lead to a high-scarcity of drone-related dataset and a high data collection cost [12]. Low-quality drone datasets could significantly slow down the training of DL and increase the cost. Thirdly, the network size of DL models is an important factor to guarantee performance. However, a large network size will increase the training cost in energy and time, and reduce the transparency of DL models. So, a trade-off should be made between the DL model economy (which considers training/running cost) and model performance (e.g., accuracy, robustness, explainability).

There are two ways to optimize the economy and trustworthiness of the drone-related DL models. Firstly, find efficient drone informatics in DL, and collect efficient training data. This can both increase DL training efficiency (save time and energy cost), and also eliminate the cost of collecting data that is redundant. Secondly, analyze the efficiency of DL models, and discover energy-efficient and highly explainable network structures. However, research related to the above points is still in its infancy or has not been integrated into drone scenarios. In this thesis, we combine the research of DL trustworthiness, efficiency, and explainability to understand efficient drone informatics in DL. We show, efficient drone informatics could be used to boost DL performance and trustworthiness in a more efficient way or generate attacks on DL models. We demonstrate, the research outcome of this thesis would affect DL and swarm drones.

1.1 Definition of Problem

As demonstrated in Fig. 1.1, DL models are trained on data collected from multi-sensors (e.g. cameras) to detect, classify and track drones for supervision purposes. However, there are main problems that need to be addressed:

- **Problem 1**: Since DL models are black-box models and are always used in intelligent systems, the trustworthiness of DL is very important. Other than only system performance and accuracy, the willingness of people to trust DL models under dif-

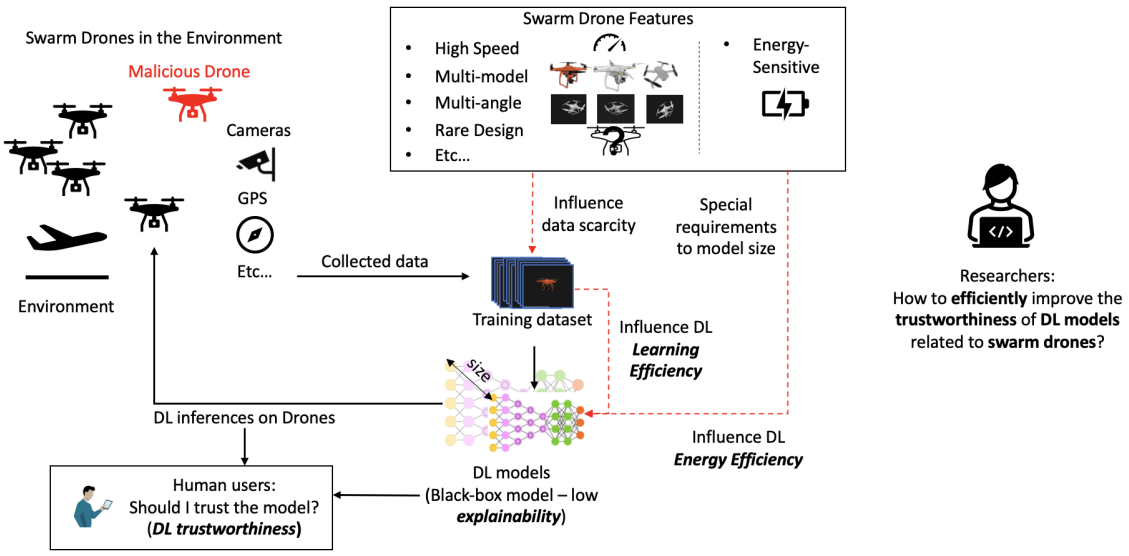


Figure 1.1: Problem Definition

ferent types of tasks is important (e.g. humans are critical to the trustworthiness of auto-pilot systems as the system behaviour could threaten life, while humans are more tolerant to non-risk systems like Alpha-Go). The high-performance metric of DL models contributes to trustworthiness, while the explainability of DL and the satisfaction of humans with their explanations are also important. Compared with specific performance metrics (e.g. F1-score) on a particular dataset, trustworthiness is a more comprehensive metric for the performance evaluation of DL models. This means, a uniform statistical metric to quantify DL trustworthiness with consideration of different trust factors is needed. Here, the structural explainability of the model itself and explanations by data are both significant. Since conventional DL models are with low explainability in the structure itself, effort should be made to design novel explainable network structures, which contribute to model trustworthiness.

- **Problem 2**: The energy consumption and inference latency of DL models are related to the computational complexity decided by network size. Network size can influence the performance of DL models, but also barrier the deployment of DL

models in energy-sensitive devices (e.g. drones). There should be a trade-off between DL network size, settings and performance according to different task requirements. It is required to understand the basic energy consumption unit of deep learning models first. Then, analyze how energy consumption is related to different model architectures and settings. Finally, analyze the energy efficiency of different DL models and find the most energy-efficient one.

- **Problem 3**: Drone dataset (e.g., images) is always with high scarcity, due to drone features (high flexibility, tiny shape and numbers of models) and collection methods (e.g. special requirements on camera). At the same time, some drone data are hard-to-collect (e.g. with specific design features, with high confidentiality, and rare drone models). A scarce drone data set increases the difficulty of DL models to learn correct data distribution, and generally reflects in a low model generalization ability. Collecting new drone images could reduce the scarcity of drone sets. However, a random collection of new data could contain redundant data, which reduces the training efficiency in both time and cost. A better way is to find what kind of data is critical in the current training phase, and collect efficient new datasets in a targeted manner. This requires analyzing efficient drone informatics from the learning efficiency of DL models on drone images, which involves research in DL learning efficiency and XAI.

1.2 Aims, Objectives, and Connections

- **Overall Aim**: The overall aim is to analyze the efficient drone informatics in DL to increase/decrease the trustworthiness of related DL models in a more efficient way.
- **Objectives** to achieve the overall aim, detailed as follows and demonstrated in Fig. 1.2:
  - **Objective 1: Analyze the contribution of XAI in DL trustworthiness; de-**

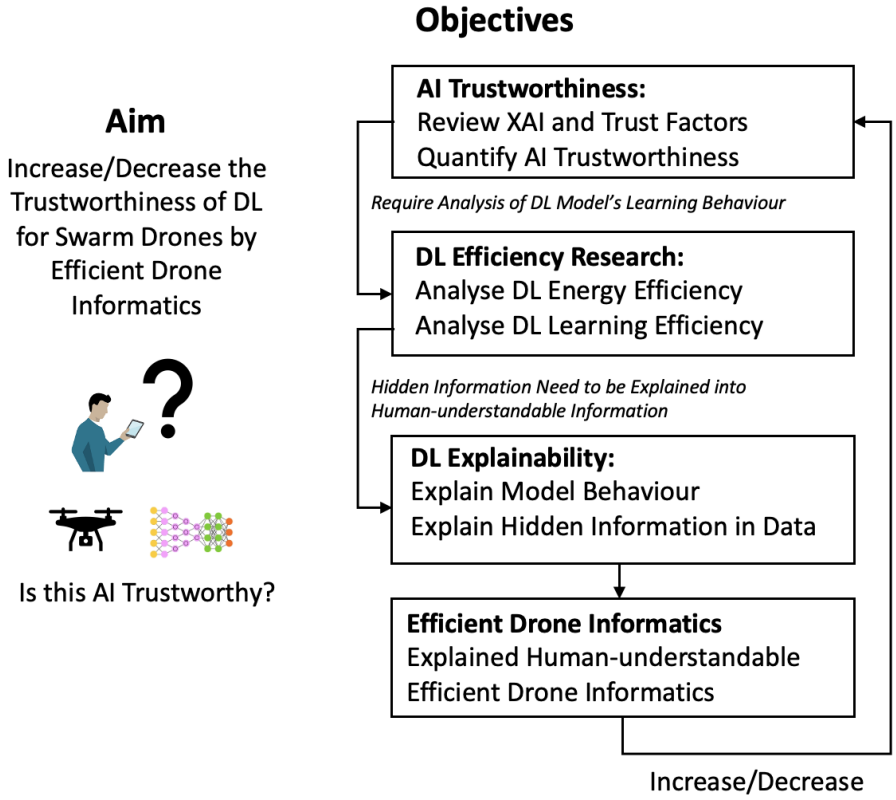


Figure 1.2: Thesis Aim, and Connections Among Each Objective

sign a new network structure to increase DL model transparency

This research will conduct a literature review of XAI methods, and analyze their contributions to DL trustworthiness. A theoretical DL trustworthiness metric Quality-of-Trust (QoT) will be proposed, which could be used to analyze the changing of trustworthiness with trust factors. A theoretical protocol for lifetime supervision of DL trustworthiness will be proposed. In order to increase the transparency and explainability of DL, a partially explainable neural network will be researched and proposed. With the use of flexible and learnable activation functions, the relationship between input features could be explained within the explainable neurons.

– **Objective 2: Analyze DL learning efficiency and energy efficiency**

The efficiency of DL models is divided into learning efficiency and energy efficiency, which focus on the learning speed of DL models and the energy

consumption for producing inferences respectively. Here, the learning efficiency indicates how fast the model could approach the distribution of the target data, reflect in the learning length (e.g., number of epochs). While energy efficiency focus on finding the relationship between the structural features of model and model energy consumption, representing why each model inference cost certain energy. By combining the learning efficiency and energy efficiency mentioned above, it is possible to gain experience in the training energy cost for different DL models. The learning efficiency of DL models will be revealed by the evolution of models' generalization ability during training. The low generalisation ability of DL models on certain latent features means these latent feature are hard-to-learn for DL models. This could tell the weak point of the DL model, and guide the improving directions. Then, the ways to quantify and estimate the energy consumption of DL models will be reviewed. A theoretical metric for DL energy consumption level will be proposed. This metric could be used to analyze the energy consumption level of a DL model based on its architecture and settings, which consider both linear and non-linear operations in DL models. And, with this metric, developers could directly estimate the energy consumption of a DL model based on its architecture, without practical deployments in hardware.

– **Objective 3: Explain the detected DL efficiency information at human-understandable level**

  The research will explain the learning behaviour of DL models, and explain the detected hidden information (e.g. hard-to-learned hidden features) about DL learning efficiency in a human-understandable way. These hidden features in drone data will further be connected to real-world drone design features, which indicate whether a drone is hard to learn for DL models. This research will produce evidence for generating efficient drone informatics.

– **Objective 4: Generate efficient drone informatics, and do verification**

(use the efficient drone informatics to improve or attack drone-related DL models in model trustworthiness)

In this research, efficient drone informatics will be generated, and evaluated in two ways: 1) improve the trustworthiness of DL drone classifier more efficiently; 2) guide the generating of attacks on DL drone detector and classifier to reduce their trustworthiness.

- **Connections**: Achieving high trustworthiness DL for swarm drones with high efficiency needs multi-discipline research involving all objectives mentioned above. AI trustworthiness research provides a more comprehensive metric which considers human emotion for evaluating drone-related DL systems (objective 1 - DL trustworthiness). Increasing the trustworthiness of drone-related DL needs improving both the explainability (objective 1 - Design transparent NN) and performance of the system, which kicks out the research of finding the aspects that need to be reinforced in DL models (objective 2 - DL learning efficiency). Here, DL efficiency research provides knowledge to understand the learning behaviour of these drone-related DL models, and finds the hard-to-learn drone features where the models are most likely to make wrong decisions (objective 2 - DL learning efficiency). As these drone features are always latent features, they need to be further explained by XAI methods into human-understandable information (objective 3). Then, the human-understandable information will be summarized into efficient drone informatics, which lead to the collection of high-efficiency training datasets to efficiently train drone-related models (objective 4). On the other hand, the DL energy efficiency research can also help the finding of high-efficiency network structure (objective 2 - DL energy efficiency).

1.3 Methodology Overview

- **Methodology Overview - Objective 1**

Quantification of DL model trustworthiness: A literature review of currently used XAI methods is proposed. DL trustworthiness is divided into physical trust and emotional trust. Physical trust implies trust only from the performance of DL models such as accuracy, robustness, and explainability. Whereas emotional trust means the willingness of different user groups to trust a DL model based on the performance metric of the model (e.g., satisfaction with the explanation). Then, physical and emotional trust are calculated theoretically, and further combined into a metric of the overall trustworthiness of the model. Quality-of-trust is proposed based on the requirement of trustworthiness from each DL task.

Explainable neural network structure: A review of neural network (NN) topology and activation functions is proposed, followed by a discussion on their relationship to NN explainability. The Gaussian process is used in this research as a trainable flexible activation function to replace conventional fixed-shape activation functions (e.g. ReLU). The learnable parameter is designed as the coordinates of control points in each neuron. Then, a high-transparency new neuron network topology is proposed based on Kolmogorov–Arnold Superposition Theorem (KST). It limits the number of layers, and sets the number of neurons according to the scaling law between the number of neurons and the overall explainability of NN.

- **Methodology Overview - Objective 2**

  DL energy efficiency: A review of current DL energy consumption estimation methods is proposed. How different types of numerical operation (linear and non-linear) are operated on hardware Arithmetic Logic Units (ALU) and the resulting energy consumption is discussed. DL model structures are opened to calculate how many basic operations (BOs, e.g., multiplication) are involved. Transistor operations (TOs) are counted based on BOs, as a metric for hardware workload. A regression model will be trained to fit the relationship between BOs and practical energy consumption on the hardware platform, which will be used to estimate the energy

consumption of other DL models.

DL learning efficiency: A review of how different error types lead to DL inference failure and a discussion on data distribution in DL are proposed. A method that uses generative neural networks (GAN) to represent the common learning behaviour of convolutional layers in different DL models on drone images is proposed. This explores the evolution of the convolutional layers' generalisation ability to latent features in drone images during model training. Low generalisation ability indicates DL model has low learning efficiency on current latent features.

- **Methodology Overview - Objective 3**

  Topological Data Analysis (TDA) is used to visualize and compare the data distribution in drone images and that learned by GAN. TDA can discover topological structures in drone images that are hard for GAN to generate synthetic images on (low generalization ability). Then, hard-to-learn topological structures (in drone images) will be summarized, and be explained by data tags (drone model name).

- **Methodology Overview - Objective 4**

  The common design features that make certain drones difficult to learn by DL will be analyzed by drone experts from hard-to-learn drones, which forms efficient drone informatics. To verify the efficient drone informatics, two research is proposed:

  Collect efficient drone image dataset for DL training: Given a drone image dataset (raw) for DL training, hard-to-learn drone models are analyzed by methods proposed for Objectives 2 and 3. A DL model for drone classification will be trained on this raw dataset. Two additional drone image sets will be collected: 1) evenly collected from all drone models or 2) from only hard-to-learn drones. Then, the drone classifier will be trained on the raw dataset and one of the additional image sets. If the DL model trained with additional images from only hard-to-learn drone

models outperforms the other one (in learning speed), the efficient drone dataset is effective.

Design visual neural stealth drone canopy: Based on the hard-to-learn drone models extracted previously, drone experts can analyze their common design features (analyze hard-to-learn design features). A new drone canopy will be designed on hard-to-learned features, produced by 3D printing, and assembled on a physical drone. Also, an adversarial patch with evasion features will be trained to fool DL models, and printed on the canopy. The effectiveness of the visual neural stealth drone will be verified on a DL generic object detector and a DL drone classifier. This method will further be extended into swarm drone usage.

1.4 List of Related Published/Submitted Works

- **Paper 1**: C. Li, W. Guo, S. C. Sun, S. Al-Rubaye and A. Tsourdos, "Trustworthy Deep Learning in 6G-Enabled Mass Autonomy: From Concept to Quality-of-Trust Key Performance Indicators." In *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 112-121, Dec. 2020, doi: 10.1109/MVT.2020.3017181. [Published, First author]

- **Paper 2**: S. C. Sun., C. Li, Z. Wei, A. Tsourdos, and W. Guo. "Scalable Partial Explainability in Neural Networks via Flexible Activation Functions (Student Abstract)." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 18, pp. 15899-15900. 2021. [Published, co-First author]

- **Paper 3**: C. Li, A. Tsourdos, and W. Guo. "A Transistor Operations Model for Deep Learning Energy Consumption Scaling." In *arXiv preprint arXiv:2205.15062*, 2022. [Accepted, First author]

- **Paper 4**: C. Li, S. C. Sun., Z. Wei, A. Tsourdos, and W. Guo, 2021. "Scarce Data Driven Deep Learning of Drones via Generalized Data Distribution Space." *arXiv*

Problem	Objectives	Papers	Chapter in Thesis
Problem 1	Objective 1	Paper 1,2	Chapter 2,3
Problem 2	Objective 2	Paper 3	Chapter 4
Problem 3	Objective 2,3	Paper 4	Chapter 5
Problem 1,2,3	Object 4	Paper 5	Chapter 6

Table 1.1: Mapping the Research Problems, Objectives and Papers

preprint arXiv:2108.08244, 2021. [Submitted, co-First author]

- **Paper 5**: C. Li, X. Huang, A. Tsourdos, W. Guo, "Designing AI Stealth Drones using Adversarial Topology and Imagery.", 2022. [Submitted, First author]

1.5 Paper Organizations

This thesis structure is illustrated in Fig. 1.3, and the mapping relationship of research problems, objectives, submitted works and chapter of the thesis is shown in Table 1.1. In **Chapter 2**, a literature review of XAI methods and AI explainability is presented; followed by the definition of physical trust and emotional trust, and how they contribute to Quality-of-Trust (QoT); DL trustworthiness testing protocol is proposed later. In **Chapter 3**, a partial explainable NN structure with Gaussian process as activation functions (AF) is proposed, followed by a case study to demonstrate how to explain the NN behaviour through AFs. In **Chapter 4**, a method using Transistor Operations (TOs) as a metric of DL energy consumption is proposed, followed by a method validation, which estimates the energy consumption of different DL models on different hardware platforms. In **Chapter 5**, a GAN-TDA framework is proposed to analyze which drone images in a drone image set are crucial to improving the performance of DL drone classifiers. In **Chapter 6**, based on the common design features of hard-to-learn drone models found in Chapter 7, an AI stealth drone canopy is designed to escape DL detection. Evasion patterns are trained and printed on the canopy to further fool the DL object detector and drone classifier (for individual and swarm drone scenarios). **Chapter 7** summarizes the key findings and explains their impacts on academics and industries. Finally, the conclusion and future works are clarified in **Chapter 8**.

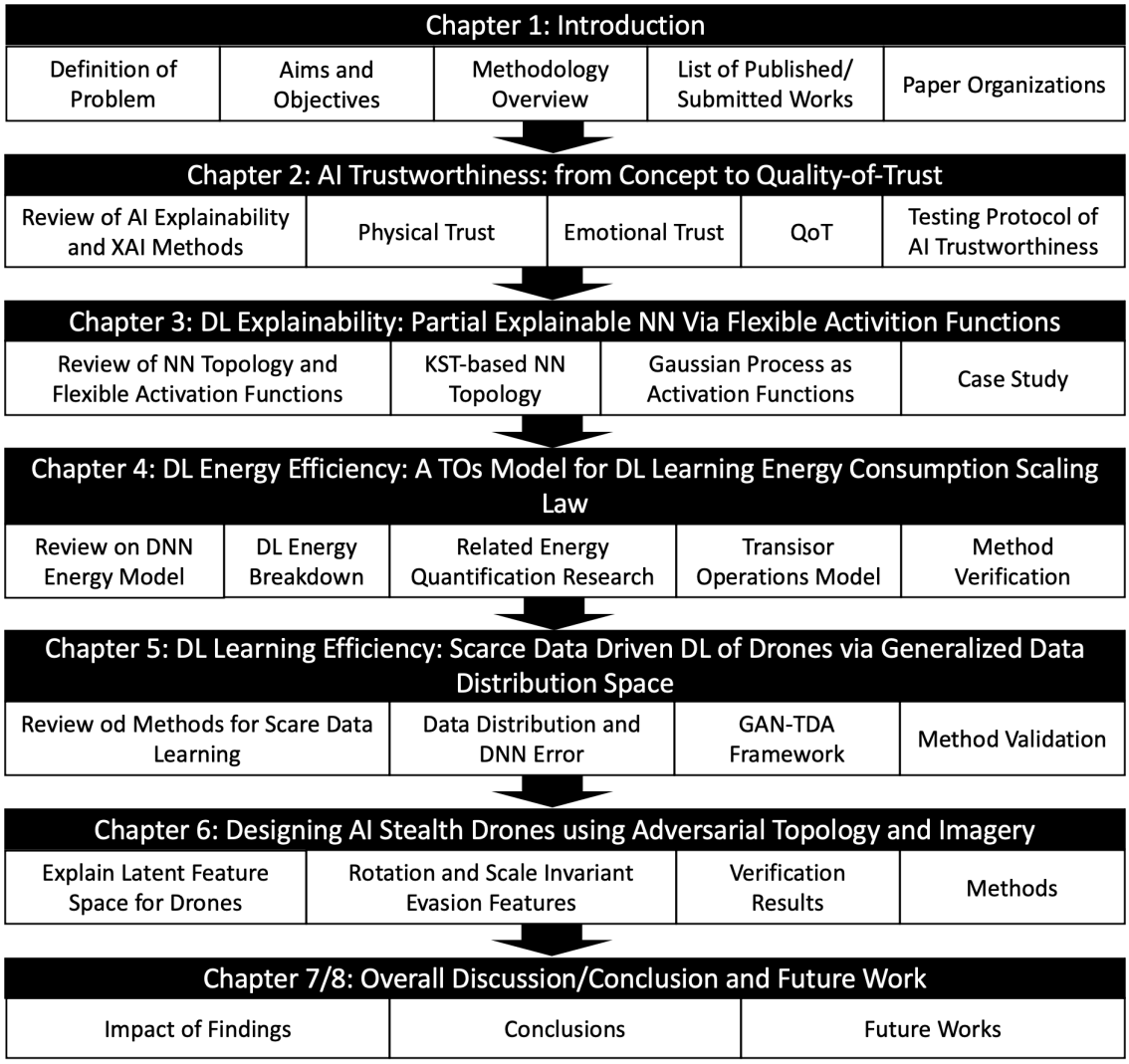


Figure 1.3: Paper Organizations

Chapter 2

AI Trustworthiness: from Concept to Quality-of-Trust

Mass autonomy promises to revolutionise a wide range of engineering, service, and mobility industries. Coordinating complex communication between hyper-dense autonomous agents requires new artificial intelligence (AI) enabled orchestration of wireless communication services in beyond fifth generation (5G) and sixth generation (6G) mobile networks. In particular, safety and mission critical tasks will legally require both transparent AI decision processes, and quantifiable Quality-of-Trust (QoT) metrics for a range of human end-users (consumer, engineer, legal). The thesis outline the concept of trustworthy autonomy for 6G, including the essential elements such as how Explainable AI (XAI) can generate the qualitative and quantitative modalities of trust. The thesis also provide XAI test protocols for integration with radio resource management and associated key performance indicators (KPIs) for trust. The research directions proposed will enable researchers to start testing existing AI optimisation algorithms and develop new ones with the view that trust and transparency should be built in from the design through to the testing phase.

This chapter provides a fundamental understanding of AI trustworthiness considering both model-oriented physical and human-oriented emotional trust. The proposed QoT

metric theoretically quantifies the trustworthiness of DL models under different task requirements, and guides the improvements of (swarm) drone-related DL models in trustworthiness. This chapter indicates the improving of DL trustworthiness needs explainable model units (Chapter 3), appropriate and efficient network structure (Chapter 4), and finding the weak ability of the model (Chapter 5).

This chapter has been published as a research paper by the IEEE Vehicular Technology Magazine [13].

2.1 Introduction

As 5G networks roll out across the world, researchers are sewing the seeds for the ideas and technologies that will shape future 6G mobile networks. 6G networks are likely to be increasingly integrated with hyper-dense mass autonomy, where intelligent agents (from mechanical robots to data analytic engines) are complementing and supplementing human labour across a diverse range of local industrial, commercial, agricultural, and mobility services. Internet-of-Autonomous-Things (IoAT) will require highly tactile and robust wireless communication channels. This will increase the network complexity in safety critical operations, and therefore Quality-of-Trust (QoT) requirements are necessary from legal, safety, and ethical reasons. Although AI is anticipated to be an enabler - to optimize high dimensional network resource management from the bottom to top layer [14], many deep learning algorithms' low transparency in processing logic leads to difficulties in exploring model transparency and uncertainty. In turn, this risks undermining the human trust in AI-empowered services, and slow down their ubiquitous adoption in real systems. Here, the chapter attempt to describe both a technology agnostic approach for 6G - adding a trust brokerage, but also give wireless specific examples to aid understanding.

The legal requirements for an all data-driven autonomy requires decisions to be explainable to human beings to enable transparency and pave the way for legal responsibility. After all, communication channels are increasingly responsible for *safety-critical*

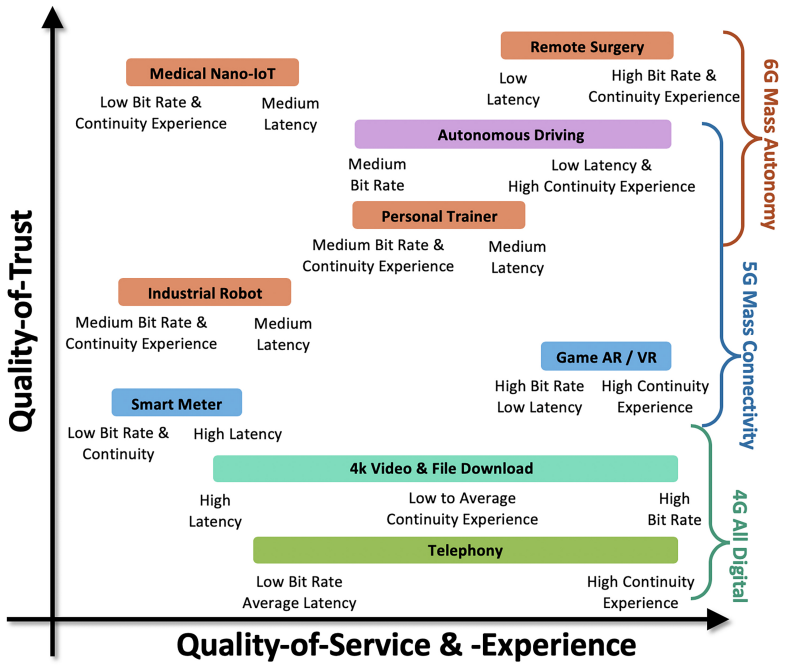


Figure 2.1: Relationship Between the Proposed Quality-of-Trust (QoT) with Traditional Motions of QoS and QoE for Diverse Wireless Services.

tasks such as autonomous driving, remote surgery, and manufacturing. Legally, the General Data Protection Regulation (GDPR) in EU propose ”right to explanation” that request machine learning models provide reasoning through dyadic statements. As such, AI orchestration of resources (communication, computing, storage) in 5G beyond and 6G will need to offer QoT in additional to the current Quality-of-Service (QoS) and Quality-of-Experience (QoE) targets. As the chapter expand the use of autonomous systems, trust and the associated KPIs to measure it will become increasingly important.

2.1.1 Trust of XAI in 6G Autonomy

Orchestration of diverse service requirements in 5G and beyond has led to the proposed adaptation of deep learning optimisation approaches to overcome growing complexity. For example, in the PHY layer, deep learning’s high-dimensional ability to achieve effective non-linear channel equalisation can enable new levels of QoS in highly complex scatter rich channels without channel state information (CSI) [15]. In the MAC layer, deep Q-network (DQN) is used to optimise a variety of high-dimensional dynamic RRM

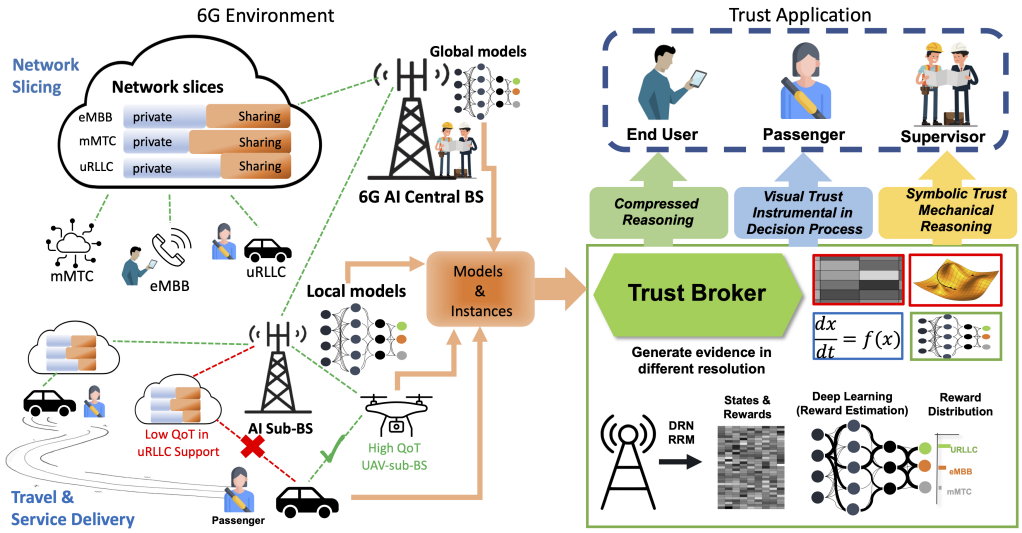


Figure 2.2: 6G Network Slicing and Trust Broker for Different Applications and End User Stakeholders. The Trust Broker Translates AI Algorithms into Explainable Outputs. uRLLC: Ultra-reliable Low Latency Communication. mMTC: Massive Machine Type Communications. eMBB: Enhanced Mobile Broadband. DRN: Deep Reinforcement Learning. RRM: Radio Resource Management.

challenges including unmanned aerial vehicle (UAV) relay joint navigation and communication [16]. Deep Learning (DL) can in many complex cases improve performance compared to classic approaches (DSP, SVM, Bayesian inference), especially in the absence of explicitly accurate models. For example, swarm drones are expected to be used as the remote backbones in 5G-beyond/6G to assist information relay, where a number of AI systems will involve. DNN models can be used to discover the locations that have temporary communication resource requirements. DRL models within AI-BS could indicate the topology of swarm drones to best cover the whole area for a higher QoS. During the trip of individual drones to the destination, on-board DRL models also contribute to accurate navigation and autopilot. However, the lack of transparency in the reasoning of DL models involved in each task yields a lack of human trust. For example, the communication engineers may doubt whether the DL system correctly allocates the topology of swarm drones, if the model reasoning is with low transparency. If the behaviour of the DL-based Autopilot system is with high uncertainty and without any explanation, drone engineers will distrust the system once crash happens. End users will not trust and be sat-

isfied with the system if QoS is still low and no explanations for the RRM decision made by DL models. As such, whilst the design logic of DL and Deep Reinforcement Learning (DRL) is clear, the data features' propagation and the logical reasoning processes are not.

The increased demand for mass autonomous prompt the requirement of trust metrics, such as the Quality of Trust (QoT) proposed in this chapter. As shown in Fig. 2.1, there is increased emphasis from new services on high trust - ranging from remote surgery (high trust, high QoS & QoE) to industrial robotics (medium trust, but low QoS demand). These will sit alongside current telephony and multimedia services that require very little trust, but a large variation in QoS/QoE. To achieve trust of AI wireless resource orchestration, the chapter will propose the need for a trust broker entity in future wireless networks - see Fig. 2.2. This entity can produce a variety of visual, textual, symbolic explainable outputs, offering reasoning to deep learning actions embedded in the base stations. The reasoning outputs speak to human stakeholders in a variety of applications, ranging from engineering experts to end-users. As such, a range of KPIs and test scenarios should be developed. Considerations should include human psychology and philosophy aspects and a high-quality XAI model should have the ability to clarify itself in human-understandable ways (different modes) based on their purpose.

2.1.2 XAI and QoT in Future 6G Network Slicing

Mass autonomy in 6G will demand localized sub-network slicing for diverse and dynamic service demands (incl. trust in safety critical multi-modal actions). Therefore, current Software-Defined Networks (SDN) in 5G will need to adapt its network slicing (NS) to meet rapid multi-modal service requirement transitions in hyper-dense autonomous system environments [17]. AI-empowered 6G is envisaged to grant BSs with *edge intelligence* by embedding high speed, precise and robust AI algorithms to ensure safety critical multi-modal mass autonomy in localized sub-network settings, as shown in Fig. 2.2. Current 5G network slicing has virtually split the network into different independent slices according to service types (e.g. eMBB). Future AI-empowered slicing in 6G will be more

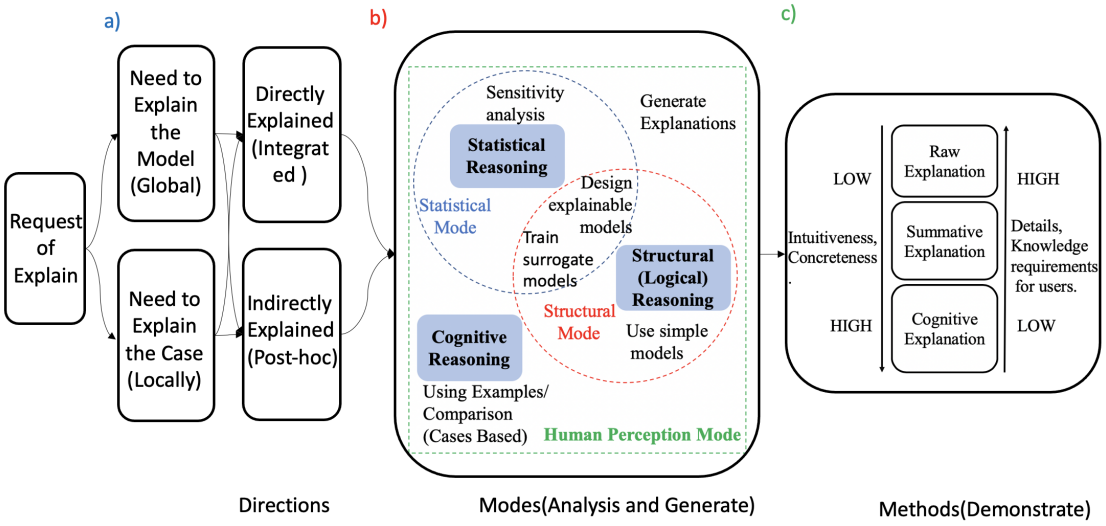


Figure 2.3: Mapping between Explainable AI to Trust: Demonstration of a) Directions, b) Modes, c) Methods and Their Relationships

fine-grained at the sub-network level and allocate by different new human-centric requirements (e.g. QoT for safety, ethics). Simultaneously, XAI can explain behaviors of mass control systems in both individual instances and overall policy, combined with system performance as important evidence in continuous trust supervision of 6G services.

2.1.3 Current Work, Novelty, & Organisation

In current research, uncertainty propagation in neural networks with different structure is analysed in [18]. Trust in 6G physical security is analysed and defined in [17], but lack analysis of mobile resource management trust. In the work of [19], initial and continuous trust in AI are defined but lack test protocol studies, and a recent EU 'EASA Road Map' defines trustworthiness and risk profiles for aerial autonomous systems, but lack consideration of 6G and trust quantification.

The chapter focus on reviewing the relevant concepts of trust for 6G RRM automation. The chapter focus focus on mapping the technical aspects of XAI to the psychological aspects of trust in the context of wireless networks. The chapter develops potential trust test protocols and key performance indicators (KPIs) that map AI architecture, performance,

to trustworthiness. Firstly, the chapter introduce deep learning model explainability, an important concept in the trust of AI. Secondly, the chapter articulate the methodological approaches in explainability, spanning different modalities and depth. Thirdly, the chapter design trustworthy KPIs and the test protocols for trust in 6G enabled autonomy, factoring in both quantitative physical trust and qualitative emotional trust.

2.2 Explainability of AI

Globally implementation of AI in a variety of industries has raised the attention of legal issues regarding both its reliability (e.g., variability and robustness of performance to diverse circumstances) and the provenance of reasoning (e.g., which data features caused which decisions). The chapter define the concepts of research direction, mode of analysis, methods, and KPIs for both explainability and trust in Fig. 2.3. Explainability is extracted from different characteristics of ML models with multi-level expression of reasoning in statistics, semantics, mathematics, and vision.

2.2.1 XAI Research Directions: Integrated and Post-hoc for Local and Global Interpretability

The chapter envisage that a request for explanation maybe demanded either post-event or continuously during reasoning (operations) - see Fig. 2.3-a). In either case, the request should specify the direction or granularity in which the reasoning needs to be made (e.g., at the local or global reasoning level, and at the post-hoc or integrated level.)

Integrated interpretability directly extracts explanations from the ML model structures and processing logic [20]. The structural complexity of ML models limit current integrated interpretability to only models with low complexity. Some low-dimensional classifiers (e.g. decision-tree or Naïve Bayes based channel state prediction models) could directly be explained by its processing logic. But models with complex structures, like deep neural network (DNN) or support vector machine (SVM) based RRM optimizing

models, are hard to be explained due to complex multi-layer connections or hyperplanes.

Post-hoc interpretability twin-system approach proposes to explain the AI model by using high transparency (white-box) systems (like decision trees, linear models) to mimic/mirror the AI model after training. The twin-system can roughly explain black-box models (e.g. DNN, DQN) [20] at the risk of poor approximation or over-fitting to a particular training set.

Local and global explanation indicates the granularity scope. Local explanation examines an individual prediction while global explanation explains the operational logic of the entire ML model behaviour [21]. As in Fig. 2.2, complex service slice handover decisions between BSs can be explained by local explanations (e.g., the mobility and data demand of one service), but can also be explained by the overall handover policy that governs the process.

2.2.2 Different Modes of Generating Explainability

The modes indicate methods to extract and generate the compositions of explainability from learning models, guides the turning of learning model into explainable learning model (Integrated), or the analysed explanation based on the output of models (post-doc) as shown in Fig. 2.3-b).

Structural Mode

Use simple models: low complexity in structure can generate logical explanations at the perceptual level to be accepted by users, the backtracking of the decision-making process from decision tree can directly generate explanation. Symbolic classification can also relate to physical laws or well-established optimisation results (e.g. water-filling or channel inversion).

Design explainable models: the machine learning models are reconstructed by interactive sections that process sub-tasks respectively from the overall prediction task, and each section and interaction could be inherent architecturally explainable. Sub-modular

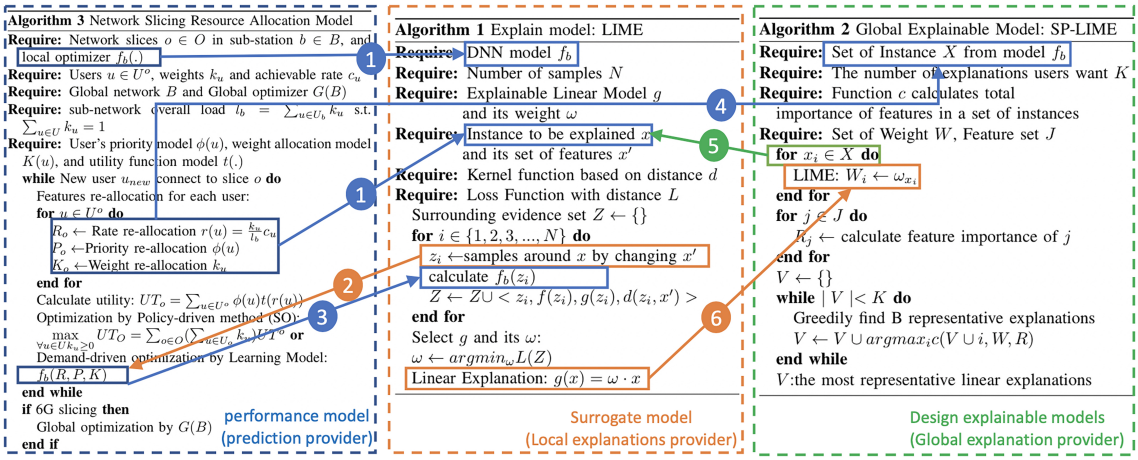


Figure 2.4: Demonstration of Surrogate XAI Model: Mapping 6G Optimisation (3) to XAI Local (1) and Global Algorithms (2).

Pick Local Interpretable Model-Agnostic Explanations (SP-LIME) [22] generates reliable non-redundant explanations globally by using a set of representative-enough instances from LIME (a surrogate model introduced later) see - Algorithm 2 in Fig. 2.4; it iterates to greedily find the explainable set by the softmax of instance coverage then let users choose interested cases themselves and track the raw data.

Statistical Mode

Sensitivity analysis: the gradient of input features respect to a label can give out which part contains the most influential information while doing decision. For instance, Layer-Wise Relevance Propagation (LRP)[23] allocates each input with a relevance score to intuitively analyse the contribution of each layer in DNN by the proportion for each neuron output between each layer pair.

Train surrogate models: similar to twin-system, surrogate models [24] ideally use a simple-structured model to fit the output of a complex model. LIME[22] in Fig. 2.4-Algorithm 1, is a model-agnostic algorithm, explains the complex prediction by approximation of the local case via an interpretable model (e.g. linear regression). Authors in [3] proposed a method to build a soft decision tree created by DNN, to makes hierarchical decisions with the ability to provided better generalizations and robustness to unlabelled

data.

Human Perception Mode

Using Examples/ Comparison (Cases Based): Similar/opposite cases can guarantee users with confidence in the reason why AI models handle the recent problem in a specific way or not. Case-Based Reasoning (CBR) could select relevant similar cases from database by selecting items with minimum distance to give out the reasons for model predictions.

Generate Explanations: Linguistic explanations could be one of the most powerful and intuitive explanations which should extract the interaction and logic from features, generate semantic sentences and cooperate with visualisation methods to demonstrate intuitive explanations. Authors in [25] proposed a method to generate linguistic explanation by using the coupling of visual recognition and text definitions, which generate an understandable high-leveled explanation of the prediction.

2.2.3 Methods of Explainability in 6G Mass Autonomy

Previously, modes defined where and how to generate explanations from ML models. Methods guide the way to demonstrate the explanation to human users, based on their individual demands and different knowledge backgrounds. The chapter defines 1) experts: sufficient background knowledge, designer for learning logic; 2) trained-users: sufficient background knowledge in specific area, designer of applications; 3) end-users: lack specific background knowledge, user of designed applications. XAI resolutions are divided into 3 methods based on different dimensions of their demonstration: raw explanation for experts, summative explanation for trained-users and cognitive explanation for end-users, which explain the learning model from abstract to concrete as shown in c), Fig. 2.3, details are listed as follows.

Raw Explanation

Data is the most direct, meaningful and detailed explanation in ML models, the raw explanation highlights the features with high contribution to the decision-making, which contains rich unbiased and unmodified raw information (e.g. Trust Broker in Fig. 2.2 provides activation map, gray-scales and derivatives to supervisors), but lacks expressions in logic (why and how extracted features cooperated). For example, In 6G RRM, underlying data response in the MAC layer extracted and used by ML models could help experts understand the output resource allocation, and dig out problematic channels refers to the features.

Summative Explanation

Summative explanation using design explainable models and surrogate models (high transparency 'white box') mentioned above, generates smoothing and fitted explanations based on the processing of statistics from low transparency 'black box'. As Auto-drive tasks in Fig. 2.2, the decision flow of on-board autonomous model is not visible from Raw Explanation explanations, especially for CNN encoded high dimensional features, while summative explanation could generate a fitted symbolic expression (e.g. Meijer G function) of the DL model to clarify the relationships among inputs in formula expressions. But during the extraction process, meaningful sharp data (such as outliers) could be ignored that add difficulty in data trace to experts.

Cognitive Explanation

For end-users who without the ability and interest to comprehend summative explanations, a clearly semantic or visualised explanation will be needed. Simple models (e.g. decision tree), high transparency surrogate models could propose the simple and basic information needed. Cognitive explanation will conclude the evidence to clarify the prediction in human-understandable linguistic and visualized explanations, but highly concluded explanations ignore some value details and break the information integrity that

important for experts and trained-users.

Explainability introduced above could increase the transparency of autonomy and help the development of trust supervision in 6G environment. Take service delivery as an example in Algorithm 3 in Fig. 2.4. Here, the chapter demonstrate mobility and resource allocation, and how XAI can be used to understand AI reasoning:

- The chapter starts with user u under high mobility transfer into another sub-network (Fig. 2.2). As the mobility request is received, the local sub-BS b will allocate the user to uRLLC network slice. It will then analyze its priority $\phi(u)$ and achievable rate $c(u)$, and allocate the user with individual weight $k(u)$ and re-optimize the resource allocation.

- During the optimization, 5G RRM uses policy-driven optimization that calculates the utility of each slice and find solutions based on different baseline methods like Socially Optimal (SO) and Static Slicing (SS) [26]. This basically meet the requirement in different targets (e.g. SO: maximize overall utility; SS: unilaterally optimize node), but lack the balance of these factors due to modeling clashes. As such, deep learning (f_b) can overcome the clash by finding new high dimensional nonlinear models to replace these explicit policy-driven optimization. As network slicing greatly increased network efficiency, the chapter demonstrate how XAI-modes introduced explain RRM in network slicing in Fig. 2.4 with the numbered flow steps:

1. To achieve XAI: the sub-BS provide LIME (surrogate model in statistical mode) for f_b and optimisation instance x;

2. LIME then generate i surrounding perturbation instances z_i, and use the model f_b to make predictions based on the set of z_i;

3. f_b send back the predictions and LIME use the predictions to find local approximate linear explanation;

4. To globally explain f_b, SP-LIME request a set of instances X used in performance model f_b;

5. SP-LIME send each instance x into LIME for local explanations;

6. LIME select local explanations to SP-LIME, and SP-LIME greedily find K most representative explanation sets V to globally explain the model f_b.

As such, the chapter have demonstrated both local and global explanations, both of which are important for trust, e.g. local is for understanding specific feature importance in decisions, whereas global is for the overall optimisation balancing between competing demands. Now that the chapter introduces algorithmic understanding of DL reasoning, it is required to develop trust metrics to translate between explainability and human perception of trust. As different services have different QoT requirements, trust analytic and supervision detailed in the next chapter can better guide decisions on whether trust requirements are met in a continuous manner.

2.3 Trust in 6G Enabled Mass Autonomy

"Trust" is a highly abstract conception, indicating the reliability of technology and willingness of users to trust the performance. In order to quantify the trust in 6G, several items are defined as follows. *Trust*: $T(m)$ for an explainable DL model m, which is composed by a set of sub-models $M = \{m'_1, m'_2 ... m'_n\}$, calculated by a linear combination of *physical trust* $P(m)$ and *emotional trust* $E(m)$, adjusted by a coefficient α as shown in equation 2.1. According to the phenomenon proposed in [27] that users may not need explanations from systems with extremely high accuracy, or systems they can not participate in, Trust of highly autonomic models should depend more on physical trust while multi-model-human-interaction models should be allocated more weight on emotion trust than that of physical trust.

$$T(m) = \alpha P(m) + (1-\alpha)E(m) \tag{2.1}$$

2.3.1 Physical Trust of AI Model

$P(m)$ is quantified by a product of model's robustness $R(m)$, accuracy $A(m)$ and explainability, where explainability calculated by division of transparency $\tau(m)$ and complexity $C(m)$ as:

$$\begin{aligned} P(m) &= R(m)\tau(m)\frac{A(m)}{C(m)} \\ &= R(m)\tau(m)\frac{a_m^d(\prod_{m'_n \in M} a_{m'_n})}{\sum_{m'_n \in M} \Omega_{g_n}(\omega(m'_n))/n}, \end{aligned} \tag{2.2}$$

The parameter $A(m)$ is a combined-accuracy-indicator in $(0,1)$, which equals to the product of accuracy for each sub-model $a_{m'_n}$ with different functionalities (inner accuracy for fitting and explaining), and overall prediction accuracy a_m (prediction accuracy) powered by an *importance-adjust factor, d* to adjust the importance of prediction accuracy in overall system performance. For example, accuracy of system in Fig. 2.4 is calculated by both the accuracy of NS resource allocation model and explain model LIME/SP-LIME.

Legal commercial DL model should not use confidential personal data without permission by users. Transparency $\tau(m)$ is a rate of visible features to all input features and that of sensitive information encryption. The data used in model training may contain personal or confidential information, that affects data privacy in 6G communication, but encryption algorithm (e.g. Hash) could be imported to convert the original data into training set without losing information and will be studied in future research. In [27], Glass et al indicate that the participation of human users can also be seen as part of system transparency, which will be quantified as a part of emotional trust introduced later.

Complexity $C(m)$ is highly dependent on the inner algorithms of models with different structures and processing logic. In formula 2.2, G is a set of all learning models (DT, NN, DNN, DRL, etc.), g_n is the model type of m'_n, $g \in G$; function ω quantifies the structural complexity for sub-models (for DT, the depth; for DNN, the number of hidden layers); function Ω calculates the complexity indicator for sub-model m', based on its model type g_n; a balance of complexity should be considered in Ω when multi-models cooperating to

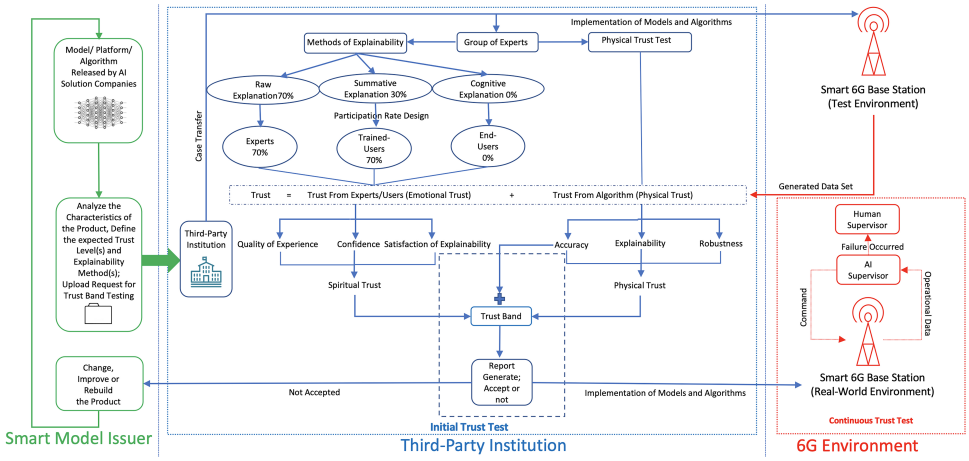


Figure 2.5: Trust Testing Flow Chart for New Released AI Models and Algorithms.

make explanations and decisions, the average complexity of sub-models is chosen in this chapter.

Robustness $R(m)$ is determined by the ability of the system to against noise and perturbation within inputs. Here, inputs will be manipulated by different methods (e.g. adding different types and levels of noise) and fed into the system for robustness verification. The system robustness could be analysed and quantified by the invariance of the system when facing the manipulated inputs. Different from the system accuracy, the robustness here is the ratio of the system that maintains the original inferences when facing manipulated inputs.

Assume the complexity $C(m)$ is stable, models with low accuracy in each sub-model will not have the ability to generate high accuracy in overall prediction. Models with high inner accuracy, but low in overall prediction will have lower physical trust. Whilst, if the complexity of the model could be reduced with the same performance, the physical trust will rise to indicate the advancement. Issuers should make improvements based on their prototype project by $argmax(P(m))$ and using the physical trust as an evolutionary indicator.

2.3.2 Emotional Trust from Human Experts and End Users

The emotional trust parameter $E(m)$ can not directly be sensed and analysed from the physical structure of model m, but can be sensed from emotion changes collected by brain-machine interactions in future 6G environment [21], or a questionnaire on user experiences. In order to quantify emotional trust, the testing institution needs to organise a test group with q participants $\{t_1, t_2..., t_q\} \in T$. Daily trust baseline indicates the willingness of trust for each individual, and could affect their choice in emotional trust test (emotional changes will make people highly or lowly willing to trust). Continuous testing of the individual baseline is necessary that those with unstable moods should not participate in the emotional trust test [21]. As shown in equation 2.3, accepted testing data $\gamma(t)$ will be fine-tuned by a factor $l(t)$ based on the willingness gap between the daily baseline and long-term baseline of individual participants.

$$E(m) = \frac{1}{q} \sum_{t \in T} l(t)\gamma(t) \tag{2.3}$$

The models with soft decision tree are important for visual recognition, which is a critical element in real world autonomous system safety and trust. As such, it is envisaged that visual data is important in 6G mass autonomy support. The chapter analyzes the physical trust of models from [3] (DNN, DT and soft-DT) to demonstrate the framework, with assumptions that the robustness and transparency of these models are the same in Table 2.1. The chapter takes function Ω for DT as a linear function $\Omega_{DT}(x) = 1/4x$ that the explainability of DT is linearly influenced by it depth and a exponential function $\Omega_{NN}(x) = 2^x$ for DNN according to the difficulty to open network structures; and *importance-adjust factor* $d = 2$ as demonstration. Please note that these are intended only as proof of principle, the main contribution is the framework itself rather than any specific algorithm or parameter settings.

According to physical trust above, roughly, it is considered the pure decision tree as

Table 2.1: Performance table for models in [3]

Models	Accuracy	DNN	DT	Physical Trust
DT	94.45%	None	Depth:8	0.22302
DNN	96.86%	Hidden Layers:3	None	0.11727
DNN-sDT	99.22%	Hidden Layers:3	Depth:4	0.19689

the best model that DNN is too difficult to be explained, although the using of DNN-sDT significantly prompt the accuracy of overall model, its physical trust result influenced by the complexity indicator with the intervention of DNN, but considering robustness, transparency and emotional trust of models in real-use, the conclusion could be different for specific tasks. As for precision machining, high accuracy is the most significant; for large-scale systems like heavy industry, equipment, labor and material dispatch, the overall explainability is important that human supervisor could fine-grained monitor the overall processes and states. Chemical plant and vehicle transport systems, both explainability and precision are needed, and scenarios in this area highly depend on physical trust.

2.4 Testing Protocol for 6G Mass Autonomy

With the explainability of learning models guarantee transparency, KPI quantifies the trust of learning models, the chapter proposed a trust testing protocol in Fig. 2.5 for smart products, both initial trust test (before it launched in real use) and continuous trust test (after implement in real-world environment) should be completed to guarantee security and legality. The rating of learning models should be completed by qualified third-party institutions using a uniformed criterion rather than the product issuer.

Trust Band in dashed box of Fig. 2.5 is designed to justify which level of trust does the model achieve, layered from high to low, as 'totally trusted'; 'totally trusted with risk'; 'highly trusted'; 'partly trusted'; 'low quality' and 'fail'. Totally trusted level contains models that could directly affect human safety (like auto-break in 6G autonomous vehi-

cles), should achieve high accuracy and none failures while running; in continuous trust testing of totally trusted models, once failure observed by AI supervisor, the model will be layer-downed into 'totally trusted with risk', and the failure case will be reported to human supervisors to decide whether the product needs rebuilding. But in some processing industries, high accuracy is more necessary then trust band (e.g. precision machining), for whom, the trust band could be 'highly trust', with high accuracy but allow low probability of failures.

The newly released product should be analysed by the issuer, and upload the testing request to the third-party institution, with a packet contains: the product, its demo/running data, expected method of explainability and trust band. A group of experts will be organised to define the participation of different layer users in emotional test, based on the explainability methods introduced above, for example, AI-based transport control will need raw explanations in 70% cases, and 30% summative explanations, the test group should be allocated 70% developers and experts, and 30% trained users. By analysing the monitored data from the 6G test environment, whether the smart product is accepted or not will depend on the trust report generated from the physical test result and the emotional trust test result. Once accepted, the model will be implemented into real-world environment and be handed over to continuous trust supervision mentioned earlier; if not accepted, or human supervisors define the model is risked after 'totally trusted' model labeled as 'totally trusted with risk', the model issuer should recall the product, modified it, and re-request for trust testing.

2.5 Conclusion and Further Research

In this chapter firstly attempt to quantify trust of AI in a future wireless communication and 6G context, and outline the KPIs and testing protocols to guide its development to work alongside legal frameworks and standards. This chapter does assume a technology agnostic approach for 6G, adding a trust brokerage alongside current and new wireless

technologies. The KPI and test protocol guarantee universality that the KPI and testing protocol could be used in all learning models and scenarios. This chapter outline a number of promising local and global XAI methods, ranging from post-hoc explainability to integrated design. The proposed KPIs factor in both AI model accuracy and complexity, as well as their explainability and human emotional trust.

For future research, the measurement of model complexity ω and Ω in function 2.2 based on different algorithm structures should be defined at finer scales and these functions need to catch up with the rapid development of AI. The importance of applying brain-machine interactions in emotional trust is significant, and the influence factor $l(t)$ in function 2.3 needs to be clearly defined based on the long-term emotional trust gap.

Chapter 3

DL Explainability: Partial Explainable Neural Networks via Flexible Activation Functions

Achieving transparency in black-box deep learning algorithms is still an open challenge. High dimensional features and decisions given by deep neural networks (NN) require new algorithms and methods to expose its mechanisms. Current state-of-the-art NN interpretation methods (e.g. Saliency maps, DeepLIFT, LIME, etc.) focus more on the direct relationship between NN outputs and inputs rather than the NN structure and operations itself. In current deep NN operations, there is uncertainty over the exact role played by neurons with fixed activation functions. This chapter achieves partially explainable learning model by symbolically explaining the role of activation functions (AF) under a scalable topology. This is carried out by modelling the AFs as adaptive Gaussian Processes (GP), which sit within a novel scalable NN topology, based on the Kolmogorov–Arnold Superposition Theorem (KST). In this scalable NN architecture, the AFs are generated by GP interpolation between control points and can thus be tuned during the back-propagation procedure via gradient descent. The control points act as the core enabler to both local and global adjustability of AF, where the GP interpolation constrains the intrinsic autocorrelation

to avoid over-fitting. This chapter shows that there exists a trade-off between the NN's expressive power and interpretation complexity, under linear KST topology scaling. To demonstrate this, this chapter performs a case study on a binary classification dataset of banknote authentication. The model proposed in this chapter converges at better precision rate than state-of-the-art SVM algorithms which indicates that no performance sacrifices in this approach. Meanwhile, by quantitatively and qualitatively investigating the mapping relationship between inputs and output, our explainable model can provide interpretation over each of the one-dimensional attributes. These early results suggest that our model has the potential to act as the final interpretation layer for deep neural networks.

This chapter proposed a NN structure and new flexible AF to increase the explainability of DL models, which could increase the emotional trust and part of the physical trust mentioned in Chapter 2. As for (swarm) drone scenarios, this AF has the ability to analyse the contribution of each input feature, and the potential to discover the physical law between input features and outputs.

This chapter is published as a research paper by the Proceedings of the AAAI Conference on Artificial Intelligence [28].

3.1 Introduction

Improving human trust in deep neural networks (DNN) can be achieved by developing the statistical and explainable foundations. Trust in deep learning is important, because of the increasing widespread use of commercial DNN based artificial intelligence (AI), especially in the areas that engage with human life such as autonomous vehicles and bank transactions [29]. There are a multitude of potential risks (e.g. discrimination, adversarial attacks, over-fitting) [30] and explainable AI [31] has the potential to both offer insight during the operations and in a post-hoc manner.

3.1.1 Motivation and Related Work

At the heart of the need to add explainability / interpretability to DNNs is the need to build trust in a quantifiable way. Traditional mathematical model-based algorithms have reasonably high clarity in how a model and the input data leads to output decisions. Bayesian methods can quantify the uncertainty both in the forward and inverse problem. Whilst DNNs have been shown to be able to accelerate the solution discovery of many iterative optimisation problems [32], they remain opaque and doesn't tell us the impact of input data and bias on decisions, the reasoning for decisions, and how the DNN logic can reverse teach human experts.

Beyond these technical requirements, the legal framework for AI is still in its infancy, and there are several explicit requirements for XAI in different regions, such as EU GDPR (see Recital 71) requires machine learning algorithms to be able to explain their decisions. The key is that rightly or wrongly, humans can attempt to explain if prompted to, and need DNNs to have that equal capability in order to ensure trust and a legal pathway towards improving safety and reliability.

The most common way to achieving explainability for NN is to evaluate the impact of each input on the output, e.g. Saliency maps[33], DeepLIFT[34], LIME[35] can obtain the approximate solution to provide aforementioned type of explanation by reverse analysis for instances. However, these methods are more focusing on the direct relation between inputs and output rather than the NN structure and inner operations. Authors of [31] propose to demystify black-box models with symbolic meta-models which leads a pathway to split and explicit the inner operations of NN and inspired us to improve transparency in NNs from an activation function and topology perspective.

Flexible Activation Function Conventional NN typically have a fixed, bounded continuous non-linear activation function (AF) at each neuron, which is the key to the overall nonlinear behavior of NN. However, with the fixed AF, the expressive power of each layer is capped [36, 37, 38], thus the neural network can only become deeper in order

to fit complex high dimensional nonlinear data. Therefore, there is a strand of works [39, 40, 41, 42, 43, 44, 45, 46, 47] tried to train the NN by tuning AF for purpose of enhancing the expressive power of nodes so that to reduce the topological complexity. The core idea of flexible AF is using control parameters for the curve shaping, which can be considered as interpolation between control points, while these parameters being optimized during the back-propagation process.

Interpolation methods have applied different AFs: (i) Piecewise linear interpolation [39] has the best flexibility, but easily leads to overfitting. Thus, the penalty function is crucial and sensitive to the model; (ii) Polynomial interpolation [40] avoids the overfitting to some extent due to the constraint of the function, but the local flexibility is sacrificed; (iii) Spline interpolation [41, 42, 43, 48] has both locally and globally flexibility while avoids the overfitting, but the splines are hard to express in terms of symbolic functions, which is the gateway to all explanations [31]; (iv) Gaussian Processes (GP) interpolation [44, 49] is a non-parametric model that can give the symbolic expression of AFs, while offer an expressive power vs. explainability trade-off in training set from the kernel space. Due to the kernel function constrains, GP AFs can effectively prevent overfitting as well as give a symbolic function that conforms to intrinsic autocorrelation which is better than others for explanation purpose. Therefore, in the NN proposed in this chapter, each AF is generated by GP interpolation between control points within it, which can thus be tuned during the back-propagation procedure via gradient descent. By visualizing the activation, difficulty in explaining the role of each node and layer in NN could be decreased.

NN Topology In common NN, topology is chosen from a set of known models (AlexNet, VGGNet, GoogleNet, etc) or customized with few limitations [50, 51, 52]. Moreover, from the field of evolutionary computing, search algorithms for neural network topologies are proposed in [53, 54, 55, 56, 57]. However, dynamic and complex topologies are not suitable for the main purpose – Explainability. In this chapter, the explainability scheme lay on a fixed topology mode. [58, 59, 60] shows that Kolmogorov–Arnold

Superposition Theorem (KST) can offer an approximation to any continuous function in high dimensional space using a finite composition of (a) univariate continuous functions and (b) addition operation. Therefore, based on KST, this chapter establishes a scalable NN topology as the foundation of the explainability since both (a) and (b) are the basic elements of all operations which are easier for understanding.

3.1.2 Novelty and Contribution

This chapter approaches the problem of NN explainability framework by introducing flexible activation functions into Kolmogorov–Arnold Superposition Theorem (KST) based topology NN for interpreting its inner workings in terms of symbolic and visualized functions for each neuron. Under the proposed model, this chapter achieves global transparency to the NN and show the trade-off between NN expressive power vs. explainability. The remainder of this chapter is organised as follows. Section 2 build a system model step by step. Section 3 applies the model to the a binary classification dataset of banknote authentication for case study and conduct the explainability analysis of it. Section 4 concludes this chapter and proposes the ideas for future work.

3.2 System Model

3.2.1 Neural Network Topology

Kolmogorov–Arnold Superposition Theorem Hilbert's 13th problem solved by Kolmogorov–Arnold superposition theorem (KST) [58], neural networks (NN) can prove to be universal approximators for every continuous function mapping [61, 62]. There are many generalizations and refinements of KST. One of these is stated, which uses the primary work in [58]. To be specific, for any $D \in \mathbb{N}$, there exist $R \leq 2D$ and continuous functions $\phi_{rd}(\lambda_d) : \mathbb{I} \to \mathbb{R}$ for $d = 1, 2, ..., D$ and $r = 0, 1, ..., R$, such that: for every arbitrary multivariate continuous function $f(\boldsymbol{\lambda}) : \mathbb{I}^D \to \mathbb{R}$, where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_d, ..., \lambda_D]^{\mathrm{T}}$

there exist continuous functions $\Phi_r : \mathbb{R} \to \mathbb{R}$ for $r = 0, 1, ..., R$, such that it is defined:

$$F(\boldsymbol{\lambda}) = \sum_{r=0}^{R} \Phi_r \left(\sum_{d=1}^{D} \phi_{rd}(\lambda_d) \right) \tag{3.1}$$

as an approximate realization of function $f(\boldsymbol{\lambda})$; that is, given any $\varepsilon > 0$, $|F(\boldsymbol{\lambda}) - f(\boldsymbol{\lambda})| < \varepsilon$ for each $\boldsymbol{\lambda} \in \mathbb{I}^D$, which means functions of the form $F(\boldsymbol{\lambda})$ are dense in $C(\mathbb{I}^D)$. Kolmogorov also showed that the inner functions ϕ_{rd} are independent with the outer functions Φ_r may have latent dependence of the target function f.

KST-Based NN Topology The initial NN topology model is constructed according to the state of KST. Each neural node contains an adjustable activation function which maps the sum of inputs into its output. It is set the $\underline{R} = 0, 1, 2, ...$ in (3.1), the *repetition level*, as the only parameter of the proposed topology, which control scaling as well as the trade-off between potential approximation ability and width of the NNs model while the depth of NN is fixed. Meanwhile, each repetition in topology is represented as a *unit*, and the final estimated output $\hat{\gamma}$ is the sum of $R+1$ units. This enable us to alleviate the complexity of explanation into a fixed mode. Schematic diagrams in Fig.3.1 demonstrate two topology examples and indicate that the KST topology is the key to the trade-off between model expressive power and explainability while flexible activation function is applied to enhance the expressive power of each node – as shown in Section 2.2 latter.

3.2.2 Activation Function

Different from the conventional NN using fixed AF while training the additional weight and bias of each input for the neural node, unmodified input is used for each proposed neural node where an adjustable AF is within. In other words, the weight and bias of each input are embedded into the AF it is designated, and at the same time, tuning the shape of AF.

In order tackle the aforementioned problem (i.e. function discontinuity, local tuning

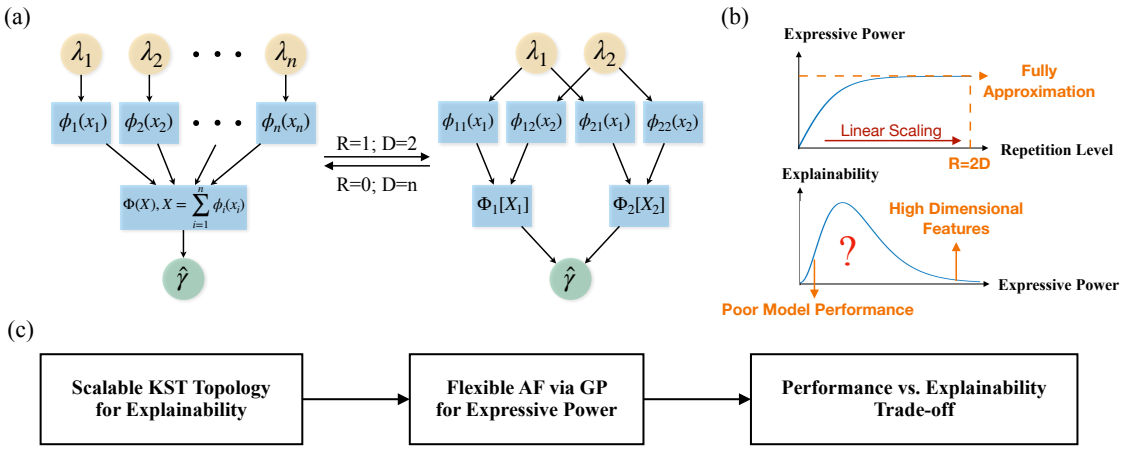


Figure 3.1: (a) The KST based neural network topology. Φ and ϕ are the flexible GP AFs. Left gives an example for one unit with n dimensions inputs while right is two units topology for two dimensions inputs; (b) Qualitative relationship within expressive power, scaling and explainability in the proposed model; (c) The working flow for scalable partial explainability in neural networks via flexible AFs.

difficulty, parameter dependency and model over-fitting), this chapter applies the noise-contained Gaussian Processes (GP) to fit the AF with control points. Specifically, each AF is generated by a GP regression of the control points while the existence of noise on control points is tolerated.

In this case, it is achieved the following objectives for the AF: (1) Ensure of the intrinsic autocorrelation within the function for smoothness and explainability; (2) Gain both local and global function adjustability; (3) Avoid over-fitting.

Priori Gaussian Processes Consider each activation function is assumed to follow a latent GP plus noise which can be expressed as:

$$\phi(x) = \mathrm{GP}(x) + \varepsilon \tag{3.2}$$

where $\mathrm{GP}(x)$ is the random variable (RV) which follows a distribution given by GP, and ε is the additive Gaussian noise with zero mean and variance σ^2. From the continuous AF domain, finite number of control points are taken as $\boldsymbol{x} = [x_1, x_2, ..., x_n]^{\mathrm{T}}$, with $\boldsymbol{y} =$

$\phi(\boldsymbol{x}) = [y_1, y_2, ..., y_n]^{\mathrm{T}}$, can be assumed to follow the multivariate Gaussian as

$$\mathrm{GP}(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})) \tag{3.3}$$

where $\boldsymbol{\mu}(\boldsymbol{x})$ is the mean function and $\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})$ is the covariance matrix given by:

$$\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}) = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \tag{3.4}$$

where $k(x_i,x_j)$ is the covariance between RVs $\mathrm{GP}(x_i)$ and $\mathrm{GP}(x_j)$ represented by the kernel function. Thus, according to 3.2 and 3.3, the priori GP probability model can be expressed as:

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{C}(\boldsymbol{x},\boldsymbol{x})). \tag{3.5}$$

where $\boldsymbol{C}(\boldsymbol{x},\boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}) + \sigma_n^2 \boldsymbol{I_n}$.

Kernel Function In GP, the covariance between every two RVs is quantified by the kernel function which interprets the potential correlation between RVs. Several appropriate kernels can be selected for different priori knowledge, e.g. smooth curve is obtained by fitting with radial-basis function (RBF) kernel and exp-sine-squared kernel is designed for periodic patterns. In the proposed activation function, the chapter aims for two goals, i.e. (1) Global autocorrelation with local adjustability; (2) Continuity and smoothness, so that rational quadratic (RQ) kernel would be a default choice in the experiments:

$$k(x_i,x_j) = \sigma^2(1 + \frac{(x_i - x_j)^2}{2\alpha l^2})^{-\alpha} \quad i,j = 1,2,..,n \tag{3.6}$$

where σ^2 determines the variance magnitude, l is length-scale parameter and α is scale mixture parameter. RQ kernel can be considered as a combination of infinite sum of RBF

kernels with various length-scales hence to vary smoothly across multiple length-scales which makes it more flexible than the RBF kernel to fit local adjustment of control points and to alleviate gradient vanishing problem while maintain the noise resistance property of GP[63]. Further more, various common and customized kernels alternatives may also be applicable in some specific scenario with priori knowledge embedded [64, 65, 66].

Posterior Gaussian Processes Kernel function selected, the hyper-parameters $\boldsymbol{\theta}$ of the kernel can be tuned by maximizing the corresponding log marginal likelihood function which is equivalent to minimizing the cost function:

$$\arg\min_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{y} + \log|\boldsymbol{C}| \tag{3.7}$$

The conventional quasi-Newton and gradient descent methods can be used in this optimization problem while [67] also offers an algorithm for reducing the computational complexity of hyper-parameter learning from $\mathscr{O}(n^3)$ to $\mathscr{O}(n^2)$ without performance loss.

In the tuning process of kernel hyper-parameters, due to the allowance of noise ε in $\phi(\boldsymbol{x})$, the AFs effectively avoid over-fitting, since the GP will follow the covariance constraints given by kernel and result in an curve which is not guaranteed to pass through all the control points. Therefore, the GP can give penalty to a potential over-fitted control point by considering it as noise, so that to keep itself still being an autocorrelation function.

Kernel hyper-parameters tuned and optimized, GP can give the posterior Gaussian distribution for every RV $y_* = \phi(x_*)$ within the AF domain with mean and variance as [68]:

$$\begin{aligned} \overline{\phi(x_*)} &= \boldsymbol{\mu} + \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{x}, x_*)\boldsymbol{C}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \\ V[\phi(x_*)] &= k(x_*, x_*) - \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{x}, x_*)\boldsymbol{C}^{-1}\boldsymbol{k}(\boldsymbol{x}, x_*) \end{aligned} \tag{3.8}$$

and here, it is assumed that $\boldsymbol{\mu} = \boldsymbol{0}$ [69]. Same as Bayesian deep learning, the GP AFs

give not only the mean values but also the variance which lead the pathway to quantify the epistemic and aleatoric uncertainties of the model [70]. In this chapter, only the mean for activation value is used to discuss the explainability.

3.2.3 Back-Propagation for Control Points Tuning

Feed-forward NN structure established, back-propagation algorithm is used for control points' coordinate tuning so that to adjust the AF. Let $L(\boldsymbol{\Theta})$ denote the loss for a batch of instances, where $\boldsymbol{\Theta} = \{x_1, x_2, ...y_1, y_2, ...\}$. In each epoch for a batch of training instances, an update on control points' coordinate is performed with a learning rate η as:

$$\begin{bmatrix} x_i^{t+1} \\ y_i^{t+1} \end{bmatrix} \leftarrow \begin{bmatrix} x_i^t \\ y_i^t \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial L(\boldsymbol{\Theta})}{\partial x_i} \\ \frac{\partial L(\boldsymbol{\Theta})}{\partial y_i} \end{bmatrix} \tag{3.9}$$

Similar to conventional NN, the chain rule still works for loss gradient descent whatever loss function (e.g. cross entropy, MSE loss, etc.) is chosen while the key is to obtain the function slope $\kappa(x_*) = \frac{\partial \overline{\phi_*}}{\partial x_*}$ at each point x_* and the gradient $\boldsymbol{\Delta_i} = [\frac{\partial \overline{\phi_*}}{\partial x_i}, \frac{\partial \overline{\phi_*}}{\partial y_i}]$ for each control point, where $i = 1, 2, ..., n$. Due to the complex optimization processes in GP regression, for calculation reduction, finite difference method in (3.8) is used for $\kappa(x_*)$ and $\boldsymbol{\Delta_i}$ approximate solutions. There are also experiments performed in which $\kappa(x_*)$ and $\boldsymbol{\Delta_i}$ are given precisely, however the accuracy improvement is ambiguous.

3.2.4 Unit Expressive Power

This chapter does not quantitatively discuss the model expressive power but qualitatively instead. This is because the expression power of model is abstract and hard to bu quantified. However, qualitative discussion proposed a general understanding of the different expressive power of conventional activation functions and flexible activation functions under the same network type and task. An example is provided as follows.

It can be easily derived that conventional sigmoid and *tanh* etc. are involved in GP-

based flexible activation functions, which lay the lower bound of the model in this section. With control points' flexibility, each neuron in this model can deal with much more complex high dimensional transforming and contain much more information than the conventional ones. To demonstrate it, a specific target function for model proposed here is represented as: $\gamma = sin[2\pi(\lambda_1^2 + e^{-\lambda_2})]$ where $\lambda_1, \lambda_2 \in [0,1]$.

The KST-based topology is set with $R = 0$ and $D = 2$ to approximate the target function. Fig.3.2.4 shows how inputs are non-linear transformed and combined into the outputs. In this example, the regression $R - square$ value converges at 0.999 with the model including only one unit (three neurons). By comparison, conventional NN with fixed sigmoid activation functions needs at least $2\times$ deeper and $5\times$ wider topology to gain an approximate result to the former in prior experiments. In other words, with GP-based flexible activation functions, the information from huge amounts of conventional neurons is integrated into several units proposed in this section. Therefore, the conventional NN can be compressed with less but more informative neurons, which have more partial explainability.

3.2.5 Symbolic Model and Visualized Explainability

Using all trained AFs, the symbolic representation of the model (3.10) can be directly derived from (3.1) and (3.8) which gives the transparency to the full mapping relationship. Furthermore, with figures of the AFs, how each instance inputs map to the output can be backtracked visually (e.g. Fig.2). i.e. At the last layer, contribution from each unit can be evaluated, so that the reason why model perform well/bad in each instance can be attributed to some specific units. In the same way, which input features are decisive at the first layer could be discovered. Moreover, this model can be easily reversed in order to explore the data group which lead to a concerned output. Therefore, the noise attributes and potential feature discrimination might be dug out. There is a comparison between the explainability provided by conventional methods (such as saliency maps and perturbation-based methods) and the proposed model. Conventional methods could

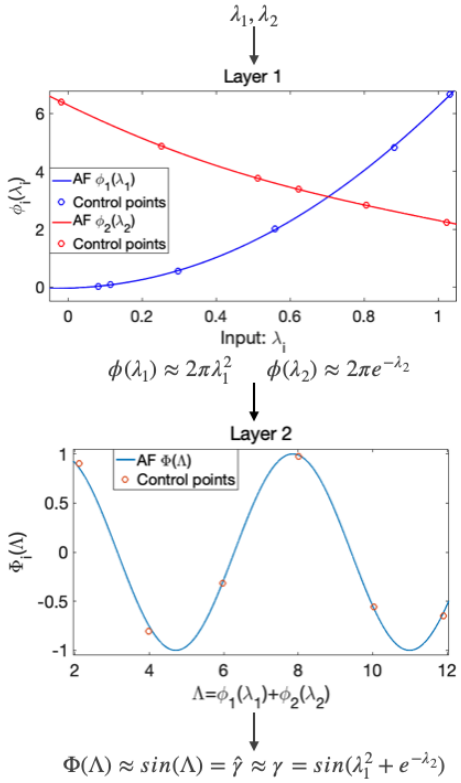


Figure 3.2: The inputs are non-linear transformed and combined into the outputs by adaptive activation functions in each neuron.

provide an intuitive visual explanation of what features are important in recent model inference (local explanations), without analysing the variation pattern of model output with these features. While flexible activation functions (e.g., GP-based activation functions) could provide both visualized and symbolic explanations for the model inference behaviour, which is known as a global explanation. This grant the activation functions the ability to express the interaction of features in inference production, and the potential to discover the statistical or physical law between the input features and the model target. In the next section, a case study to illustrate how the proposed AFs offer transparency and partial explainability for the neural network will be presented.

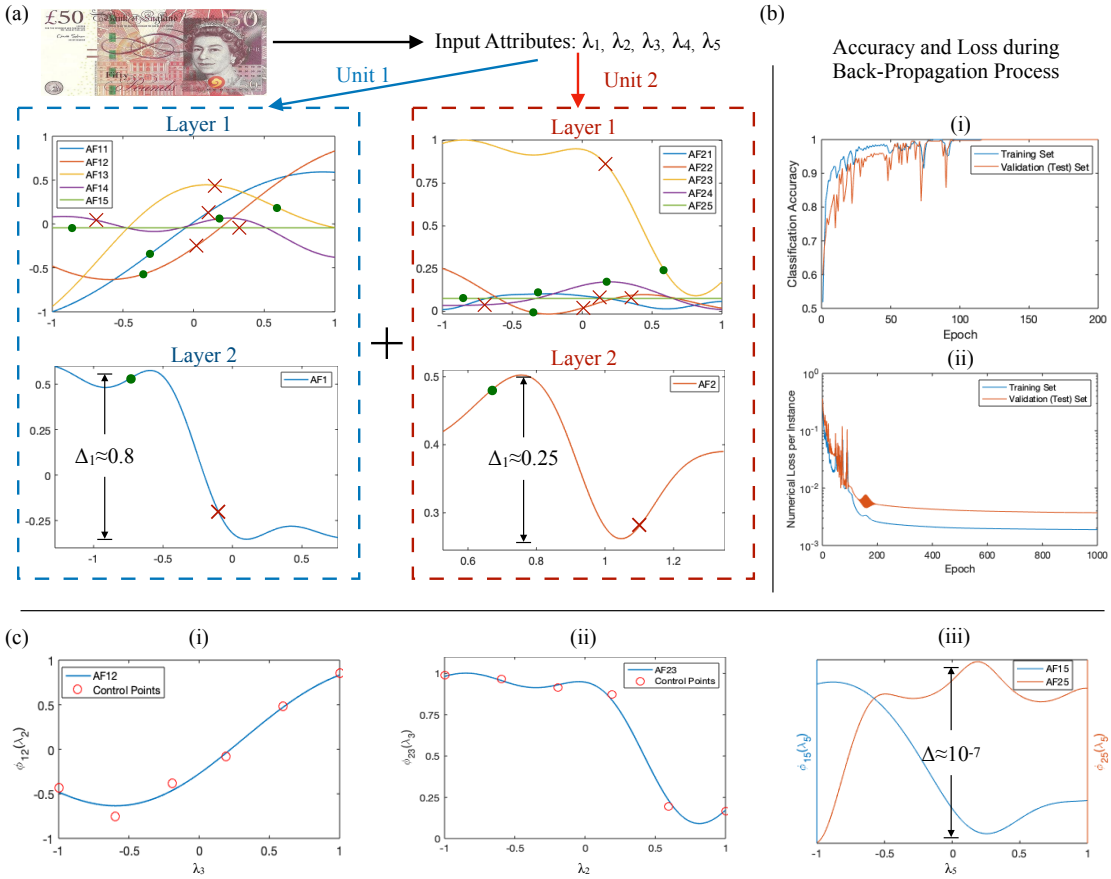


Figure 3.3: Visual Explanation: (a) The AFs in each neuron after training process are demonstrated. The green points and red crosses marked on the AFs show how two typical representative inputs from two classes data map to the output; (b) The model performance are evaluated with both the classification accuracy and the numerical loss during the back-propagation process; (c) Three noteworthy AFs are displayed separately for analysis.

3.3 Case Study and Result Discussion

In this section, a case study is provided on an open-source banknote dataset[1] from UCI , which contains four input attributes namely: variance of Wavelet Transformed image (λ_1), skewness of Wavelet Transformed image (λ_2), curtosis of Wavelet Transformed image (λ_3), and entropy of image (λ_4), while the output attribute is the class information (fake or not).

[1]Data Source: https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Results Overview Symbolic Explainability - After 428s running time, the symbolic trained model is obtained in the form of:

$$F(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5) = \sum_{r=0}^{1} AF_r \left(\sum_{d=1}^{5} AF_{rd}(\lambda_d) \right) \tag{3.10}$$

Fig.3(a) visualizes the trained NN networks with every AF while backtracking two typical representative data from two classes in the AFs with green points for the genuine banknote and red crosses for the forged one. Fig.3(b) shows the classification accuracy achieve 100% for both training and validation set after around 120 epochs, which is higher than the classical SVM (99.2%) on this dataset while the numerical loss is converged within $[10^{-3}, 10^{-2}]$ for each instance and no overfitting has occurred. Fig.3(c) focuses on three noteworthy AFs.

Model Interpretation In the model, the output result is the sum of the outputs from the two units. Consider that each unit represents a feature of the data which act as an additive to the model result, thus analyses are proposed separately on how these units effect the model result. At layer 2, the AF1 and AF2 (Fig.3a) give different value ranges with $\Delta_1 \approx 0.8 > \Delta_2 \approx 0.25$ which indicates that the unit 1 has a more decisive impact than unit 2 on the model result, meanwhile it is worth mentioning that AF1 demonstrate a distinct trend towards binary classification, which such is the case in the previous description of KST – the outer functions may have latent dependence of the target function. Firstly, focus on unit 1, it can be qualitatively observed that the λ_1, λ_2 and λ_3 are the main attributes which effect most on the unit 1 output. Fig.3c(i) gives an example for AF12 with its control points which start with a small decrease followed by an increase. A conclusion could be made that the lower λ_2 make the banknote more easily to be classified as genuine. Turn to unit 2, the λ_3 almost dominate on the output of unit 2. With a rapid change over $[0, 0.5]$ interval in AF23 (Fig.3c(ii)), λ_3 attribute has the ability to distinguish the two classes of data in unit 2. In both units, the impact of attribute λ_4 is not notably on the classification result of the model due to relatively narrow range of AF14 and AF24.

Meanwhile, Fig.3c(iii) shows that the proposed model has the robustness to noise attribute λ_5 – AF15 and AF25 give extremely little contribution to the model result other than overfit the data.

In the proposed model, the symbolic expressions can help data scientists with better understanding of the global mapping relationship within the NN. Furthermore, by analyzing the trained model reversely, scientists can anticipate potential risks in advance about how banknote would be forged to pass the detector under this case. Meanwhile, for AI users, the one-dimensional visual functions flow offer the transparency and partial explainability on how model result come from each attribute input, which can enhance the trust to the model.

Scalability, Expressive Power and Explainability The proposed model is linear scalable with the repetition level in topology and the number of control points in each neuron. The number of control points determines how much details can the function has while the units number determines the maximum number of potential features that can be extracted. Fig.4 shows that more units and more control points lead the model converging at lower numerical loss which means stronger model expressive power. However, the difficulty of model explaining for AI users is increasing at the meantime – The AI users are not sensitive to small variations in the function (Fig.4a) while confuse the role of multiple units (Fig.4b). The ascent of model accuracy and model explaining difficulty are subject to different scales – *log* scale and linear (or even exponential) scale. Thus, there needs a balance between these two or establishing a trade-off based on different scenario requirements.

3.4 Conclusion and Future Work

In this section, it is found that by modelling neural networks as a combination of linear scalable architectures via the Kolmogorov–Arnold Superposition Theorem (KST) and generalised Gaussian Process activation functions, makes it able to demonstrate a new

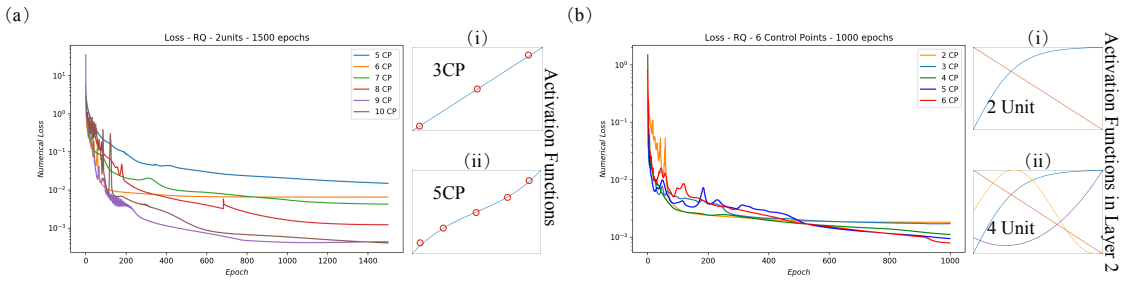


Figure 3.4: More control points in each neural (a) and more units in topology (b) can lead to better model performance, however as such, the difficulty of model explaining for AI users will increase at the meantime.

degree of interpretability. In particular, this section has demonstrated a trade-off between expressive power and explainability of the NN through controlling the KST repetition level. This work expands on current state-of-the-art interpretation methods (e.g. Saliency maps, DeepLIFT, LIME, etc.) by focusing more on the role played by activation functions and the NN structure itself.

To demonstrate applicability, this chapter performs a case study on a binary classification dataset of banknote authentication. The model in this section converges at a better precision rate than state-of-the-art SVM algorithms which indicates that the NN proposed in this section does not make performance sacrifices. Meanwhile, by quantitatively and qualitatively investigating the mapping relationship between inputs and output, the proposed explainable model can provide interpretation over each of the one-dimensional attributes. These early results suggest that the proposed model has the potential to act as the final interpretation layer for deep neural networks.

The feasibility of using flexible activation functions to achieve partial explainability is proven in this chapter. Other substitution AFs will be discovered and join the explainable AF family. However, the author concerns the overfitting problem may occur when given the AF with too much flexibility (e.g. too many control points in GP). Current GP AF could also be applied in currently used conventional neurons, however, the explainability of NN is still highly related to the complexity of layer structure. This means the explainability could significantly be dropped with a larger depth, and degenerate to the

similar level of conventional NN. The GP regression cost much more calculations than conventional AFs, leading to an increase in training and inference speed, which needs more optimisation. How GP AF could be applied in convolutional layers to provide explainability is not learned up to now.

The future work will focus on (1) fitting orthogonal features to different unit in order to compress the network by reducing the repetition level while keeping approximately the same model accuracy [71, 72]; (2) establishing a pattern library of common features using activation functions for easier identifying the feature represented by a unit.

Chapter 4

DL Energy Efficiency: A Transistor Operations Model for Deep Learning Energy Consumption Scaling Law

Deep Neural Networks (DNN) has transformed the automation of a wide range of industries and finds increasing ubiquity in society. The high complexity of DNN models and its widespread adoption has led to global energy consumption doubling every 3-4 months. Current energy consumption measures largely monitor system wide consumption or make linear assumptions of DNN models. The former approach captures other unrelated energy consumption anomalies, whilst the latter does not accurately reflect nonlinear computations. This chapter is the first to develop a bottom-up Transistor Operations (TOs) approach to expose the role of non-linear activation functions and neural network structure in energy consumption. Activation functions in NN require hardware to process complex non-linear operations (e.g., division) and result in energy consumption, where conventional methods (e.g., FLOPs) only consider the energy consumption of linear operations (addition and multiplication) in NN. As there will be inevitable energy measurement errors at the core level, I statistically model the energy scaling laws as opposed to absolute consumption values. The models are offered for both feedforward DNNs and convolution

neural networks (CNNs) on a variety of data sets and hardware configurations - achieving a 93.6% - 99.5% precision. This outperforms existing FLOPs-based methods and our TOs method can be further extended to other DNN models.

(Swarm) drones as energy-sensitive devices, whose QoS will be influenced by endurance. This means, the onboard DL model should have high energy efficiency. This chapter provides a tool to analyse the energy efficiency of different DL structures and settings. The work helps the achieving of high energy-efficiency models, which could limit the NN size (reduce model complexity, increase physical trust in Chapter 2), and more energy-efficient AFs (increase accuracy, increase physical trust in Chapter 2). Higher endurance of drone has the potential to increase the emotional trust proposed in Chapter 2.

This chapter is accepted by IEEE Transactions on Artificial Intelligence, currently awaiting publication. Early access version is available in [73].

4.1 Impact Statement

Deep learning is one of the fastest growth areas for computational resources (300,000x from 2012 to 2018, doubling every 3-4 months). Data centres are predicted to dominate over 20% of global energy consumption by 2030. The proposed TOs model provides developers with a theoretical model to expose the important role of both (1) nonlinear activation functions and (2) DNN model structure in its energy consumption. This enables developers to trade-off between model performance and sustainability with 93.6% - 99.5% precision. Due to the consideration of both linear and non-linear operation in TOs, it can to some extent replace FLOPs/MACs count as a more accurate metric of DNN model complexity.

4.2 Introduction

Rapidly increased Artificial Intelligence (AI) demand has generated a huge increase in computational resource requirement (300,000x from 2012 to 2018) [74]. Energy consumption in data centres around the world to maintain data and learn models will account for 10% of global energy consumption in 2025 and 20.9% in 2030 [75]. Whilst previous attempts in green communications have reduced networking energy consumption [76], Internet-of-Everyting will connect intelligence and it is important to reduce energy consumption across connectivity and autonomy [77]. The endless chasing of higher-precision in DNN spawns ultra-large-scale models, especially in computer vision (CV), natural language processing (NLP) [78, 79], and also communication networks [80] (see - Fig. 4.1*a*: YOLOR-D6 for CV with 174.7 million parameters in May 2021 [81]; openAI GPT-3 for NLP with 175 billion parameters in May 2020 [82]). Researchers in [83] argue that this trend is unfriendly to computational resources, energy and the global environment. Developers should carefully analyze the requirements (e.g., precision, robustness) and backgrounds (e.g., computational hardware, energy supply) of DNN tasks to make a trade-off between the performance and economy of DNN models. The network size of large-scale DNN models also barriers their deployment in energy-sensitive devices (e.g., Drones and remote sensors). Developers urgently need a theoretical method to analyze the scaling law of DNN model energy consumption during model configuration changes to design and select energy-efficient DNN models.

Indeed, there are now widespread efforts to reduce the size of neural network architectures both post-training [84], federation to the edge [85], and more recently during training [86]. Current theoretical DNN complexity metric FLOPs [87] and MACs [88] only consider linear operations (e.g., multiplications and additions) without non-linear operations (e.g., root operation in activation functions). However, non-linear operations are a non-negligible part of some DNN models [89] that the amount of calculation tasks and energy cost for doing a non-linear operation is always higher than linear ones [1, 90]. Thus, using FLOPs/MACs in analysing the scaling of DNN model complexity and en-

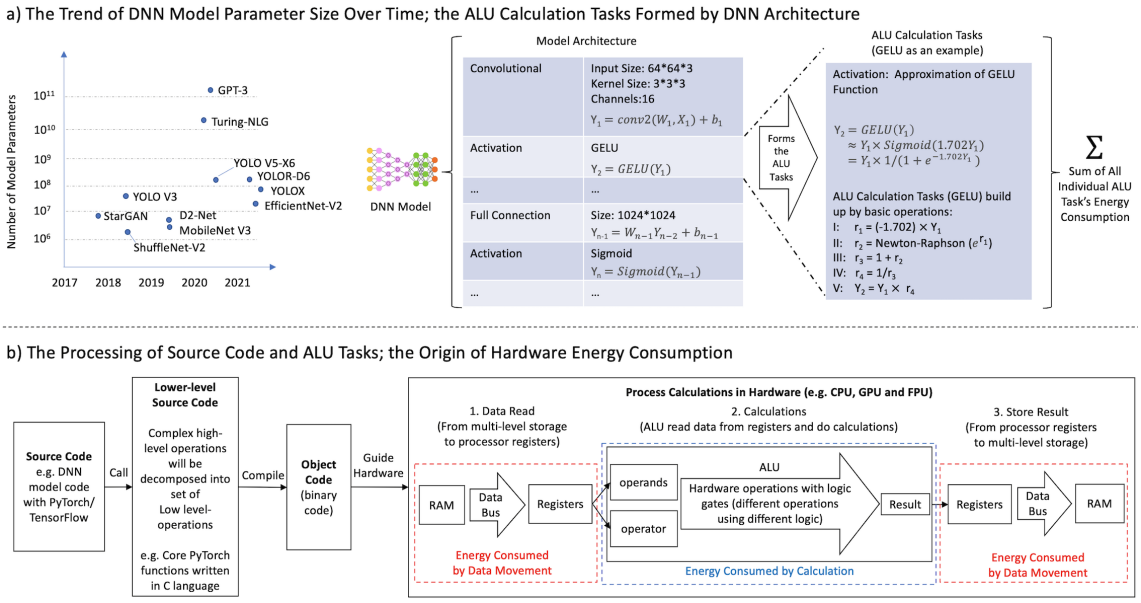


Figure 4.1: The Trend of DNN Model Parameter Size Over Time and the Origin of DNN Energy Consumption

ergy consumption is not precise enough. Simultaneously, binary neural network [91, 92], low precision arithmetic [93], low precision number [94, 95, 96] and high-efficiency operators [97, 98, 99] they affect the DNN energy consumption, without effect in model FLOPs/MACs. To solve the issue above, this chapter proposed a more accurate TOs method to analyze DNN model calculation tasks and energy consumption, which considers linear, non-linear operations and floating-point format. I show the use of TOs as a metric of DNN complexity could boost the precision of DNN energy estimation models. Details are introduced later.

4.2.1 Review on Deep Neural Network Energy Model

DNN energy consumption in training, validation, and testing can all be related to the model complexity, data size, and hardware implementation [100]. As shown in Fig. 4.1*a*, the DNN model architecture (incl. the activation function) determines the resulting execution order of the equivalent arithmetic logic units (ALU) tasks. Within this, the input data affects the DNN model hyper-parameters which is part of the ALU tasks and overall

they all contribute to different energy consumption.

Fig. 4.1*b* (left to right) briefly demonstrates the running of DNN model code in hardware and resulting energy consumption. Once the DNN model source code (e.g. Python) is run, learning frameworks (e.g., PyTorch, TensorFlow) will call the relevant core functions written in lower-level languages (e.g., C/C++). For example, the Gaussian Error Linear Unit (GELU) [101] operation will be decomposed into a set of ALU-supported instructions (see - Fig. 4.1*a*). Lower-level source codes will be further compiled into object codes to guide data reading, calculation and result storage [102]. The calculation in ALU follows a designed process [103]. Firstly, operands will be called from data storage (DRAM etc.) to the processor registers through data bus. The energy for moving one unit data varies with the memory hierarchy. Secondly, do the specific hardware operation (specified in operator) on operands, and generate calculation energy consumption. Thirdly, the result will be temporarily stored in registers and then moved to other levels of storage. As explained above, mainly energy consumption of code running is for data movement and calculation. This chapter provides a review of different processors, please refer to more details in the Appendix.

Based on the information above, any comprehensive analysis of DNN energy model must encompass:

- Analyze the origin of hardware energy consumption at transistor operation (TO) and data movement level
- Translate linear and non-linear DNN activation functions into the real calculation tasks executed by ALU
- Propose a TO based energy metric that is generalised across diverse hardware operations
- Develop a regression model to quantify the energy scaling with model configuration

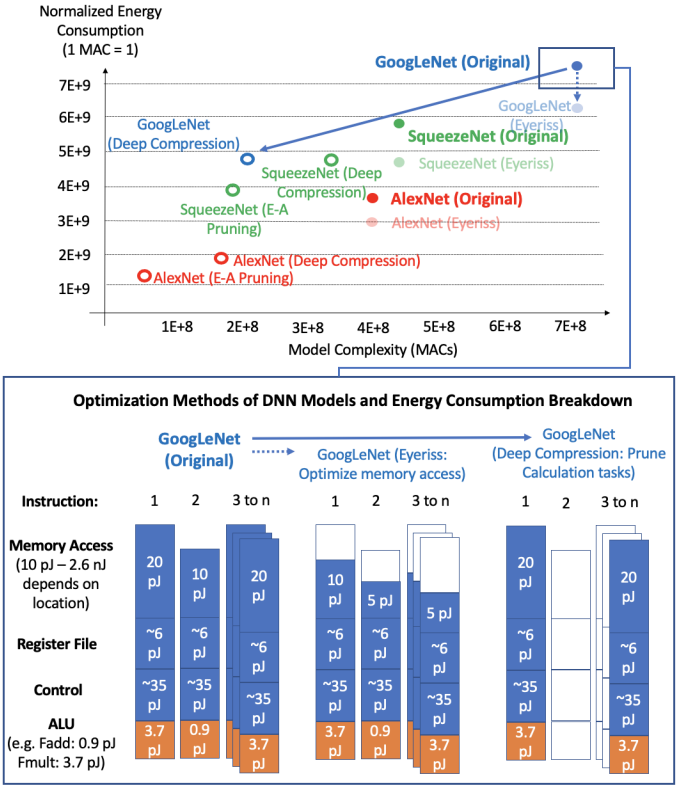


Figure 4.2: (top) Normalized Energy Consumption and (bottom) Energy Breakdown of DNN Models. Data from [1, 2])

4.2.2 Deep Learning's Energy Breakdown

Current literature shows that DNN models primarily consume two types of energy: (1) calculation energy (as described earlier), and (2) data movement and memory access energy [79]. The latter constitutes a significant part of the energy consumption, especially for large data sets [104, 105, 2]. In the execution of each instruction, data calling from hierarchy storage dominate up to *90%* energy consumption while calculations in ALU count less than *30%* [106].

As shown in Fig. 4.2(bottom), the data to be moved depends on operation tasks in instructions. Data movement optimization methods like Eyeriss [106] and MONeT [107] significantly reduce memory access energy consumption with higher data multiplexing rate and low-cost storage usage. Network pruning and compression methods (e.g., energy-aware pruning [108] and Deep compression [109]) directly cut down calculation tasks to slash overall energy consumption. This means the investigation of DNN calculation

energy cost is still beneficial in energy-aware network light-weighting (e.g., using binary neural networks and doing network pruning) in energy-sensitive DNN scenarios, optimizing neuron structure (e.g., efficient activation functions) and designing energy-efficient application-specific integrated circuits (e.g., TPU) for DNN. The maximum energy efficiency of deep learning models can only be achieved through combined optimization of both computation and data movement energy consumption. A review of DNN energy optimization is provided in this chapter, if interested, please refer to more details in Appendix.

4.2.3 Related Energy Quantification Research

There are several works investigating the energy consumption quantification and estimation of machine learning energy consumption. In [112], the authors make a survey on machine learning energy consumption estimation approaches that use simulated hardware or performance monitoring counters (PMC). They find processor plays the main role rather than DRAM in the energy consumption of tree-based models with experimentation. As an extended work, in [113], hardware energy observation tools and state-of-the-art machine learning energy consumption estimation approaches are reviewed and classified according to different techniques they use. Authors in [114] proposed a lightweight code-level energy estimation framework for software applications with limited additional cost in resources and energy. However, it is a general method focused on achieving accurate energy observation of computational hardware without the perspective of learning models.

Synergy, a method proposed in [110] uses linear regression on both the number of SIMD instructions and bus accesses observed from hardware PMC to measure and predict the energy consumption of CNNs at a layer-level. But no theoretical analysis of the relationship between CNN layer configuration and energy consumption is given. *NeuralPower* proposed in [111] uses sparse polynomial regression method to model the power and run-time according to layer configuration parameters (e.g., batch size, kernel size, etc.) of key CNN layers, then applies the model to unseen layers for energy consump-

Summary	Precision (%)	Theory Basis	Experiment Data	Focus Area	Data Movement Energy	Calculation Energy	Energy Metric	Disadvantages
Experimentation of System Level Energy Consumption Increase [110]	63.05-73.7	×	✓	SIMD & No. Data Access	✓	No distinction of operation types	Joule	No analysis of the relationship between the DNN model configuration and energy consumption
Experimentation Layer-based DNN Energy Consumption and Core Usage [111]	97.21 (avg)	×	✓	Total Run-time Power	✓	No distinction of operation types	Joule	Not distinguishing linear and non-linear operations in layer configuration and energy consumption, need information about hardware run time and power
Theoretical Analysis of DNN Complexity based on Only Linear Operations [87]	92.9-99.5	✓	×	Calculation FLOPs	×	Linear Operations	FLOPs	Ignored Non-linear Operations (e.g., activation functions)
Theoretical & Experimental Analysis of DNN Energy Consumption based on Data Movements [105]	96.4-96.5	✓	✓	Data Movement & MACs	✓	Linear Operations	MACs	Data sparsity is hard to estimate accurately; ignored non-linear operations
This Chapter: Proposed Transistor Operations (TOs) Method	93.6-99.5	✓	✓	Transistor Level Operations	×	Linear and non-linear operations	TOs/Joule	No data movement energy analysis

Table 4.1: Comparison between State-of-the-Art and the Proposed Methods for DNN Energy Consumption Estimation. Acronyms in Table: Deep Neural Network (DNN), Single Instruction Multiple Data (SIMD), Transistor Operations (TOs), Floating-point Operations (FLOPs), Multiply–accumulate Operations (MACs).

tion estimation. However, the layer-level analyze can not distinguish linear and non linear operations in layer configuration. Theoretical analysis of the relationship between CNN layer configuration and energy consumption is not given. Floating-point operations (FLOPs) is used as a simple model-structure-based DNN computational complexity measurement in [87] without mapping to DNN energy consumption. The calculation of FLOPs only consider linear floating-point operations (multiplication and accumulation) in full-connection/convolutional layers without non-linear operations and other layers (e.g., non-linear operations in AFs and operations in pooling layers). These ignored parts count a non-negligible part of DNN model complexity (around *5%* in convolutional layers and *15%* in feed-forward layers depends on the model configuration measured by the theoretical TOs model in this chapter). Authors in [105] propose a theoretical DNN energy estimation method that uses the simulation of data movements between multiple layers of storage to quantify and normalize DNN energy consumption with the energy for doing one MAC operation as the unit. However, their method ignore non-linear operations and the difference between hardware (energy for doing one MAC operating varies in different computation hardware, the rate of energy consumption for storage and computation varies with different hardware combinations). No practical energy data is given, and the memory simulation method is not open to readers. As an extension work, they propose *Eyeriss* in [106], which optimizes the data flow of CNN models using higher data reuse rates and less data movement from expensive storage. Authors in [115] review the parameter size, FLOPs and performance metric of several benchmark DNN models. They find the computation and energy efficiency of hardware is affect by the precision of floating-point (FP) numbers (e.g., FP16, FP32), and propose that multiplicative factors take the majority responsibility of DNN energy consumption. However, the work is observation-based without study the energy scaling law led by DNN model configurations.

The comparison between aforementioned DNN energy estimation methods and the proposed TOs method are summarised in Table 4.1, which also list the individual precision of each method.

4.2.4 Gap Summary & Innovation

Currently, as seen from the above review, the relationship between the DNN model configuration and energy consumption is not well established. The energy consumption of nonlinear operations in DNN is still lacking in analysis.

In this chapter, an innovative bottom-up Transistor Operations (TOs) method is developed to expose the role of nonlinear activation functions and neural network structure in energy consumption. I translate a range of feedforward DNN and CNN models into ALU calculation tasks (e.g., basic operations (BOs)). Based on the proposed TOs model, I demonstrate how the calculation energy scales when changing the model structure and activation functions. This chapter also provides a verification experiment that compares the energy consumption estimated with TOs model and the practical energy consumption monitored from general-purpose commercial processors (CPU, GPU).

Compared with energy consumption estimation methods in [113, 111], the proposed TOs method can individually analyze the calculation energy consumption with DNN model structures, and consider non-linear operations that not included in [87, 105, 111]. Latter section will show that the proposed TOs method can achieve a superior 93.61% - 99.51% precision in estimating its energy consumption.

4.2.5 Research Limitations & Applicability

The calculation tasks measured by TOs/FLOPs/MAC have limitations in mapping DNN energy consumption when the DNN model runs with multi-core. The processing of data instances in DNN training follows the same way, suitable for parallel processing [116] in modern multi-core processors. However, plenty of additional energy costs for core communications will be generated [117]. This additional core communication cost is hard to split from overall processor energy consumption and does not influence DNN calculation tasks and complexity metrics (e.g., TOs, FLOPs). Simultaneously, as reported in [118], the core communication energy cost in multi-core processors is hard to model and is an

experimental fact. Based on the aforementioned points, this chapter only consider single thread & CPU running in this chapter. In fact, this is a common issue for all theoretical metrics for DNN calculation tasks. At the same time, recent TOs model does not support piecewise-defined activation functions (e.g., Rectified Linear Unit (ReLU)) processed with comparison operators. As TOs is a theoretical metric that is directly analysed from DNN structures and configurations, it is not applicable to black box DNN models without structure and configuration information.

Ultimately, the practical energy of running the same DNN models on different hardware varies. As such, the proposed TOs model in this chapter calculates the theoretical TOs for each hardware operation with generic processing logic. The verification results show that TOs still outperforms FLOPs in estimating the energy consumption of DNN models on different hardware platforms.

4.3 Transistor Operations (TOs) Model

The calculation of theoretical TOs for a given DNN model is demonstrated as Fig. 4.3*a* (left to right). Suppose the dataset have already been pre-processed (energy for data pre-processing is not considered in this chapter, but the TOs method could be extended to data processing). Firstly, the layer list will be extracted from the model structure and settings (e.g., 2×4 full-connection layer, activation layer with Sigmoid AF). Secondly, extract running steps according to different analysis levels: *training level* involves model forward, loss calculation and backpropagation (BP); *validation level* involves forward and loss calculation; *inference level* only focus on model forward. The reason is that each step has individual calculation logic and resulting in different calculation tasks and energy consumption. Calculation tasks at different analysis levels are calculated by summing the calculation tasks of the relevant steps (e.g., validation level calculation tasks is calculated by summing calculation tasks for model forward and loss calculation). Thirdly, layer-wise analysis of how many calculation tasks are needed for each running step based on their

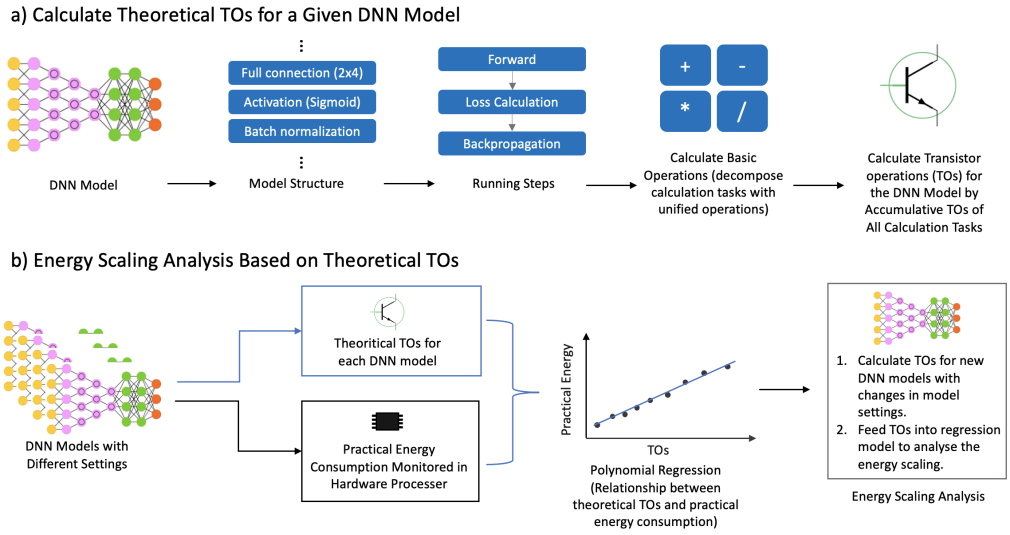


Figure 4.3: Flowchart for Calculating Theoretical TOs of a Given DNN Model and DNN Energy Scaling Analysis. (a) Calculate TOs for a given model (left to right): extract layer structure from the target DNN model; define which steps are going to be analyzed; layer-wise analyze how much BOs are needed in forward/loss-calculation/backpropagation; translate layer-wise BOs into TOs. (b) Energy Breakdown of DNN Models (left to right): prepare a number of different DNN models; apply TOs model to calculate theoretical TOs of each DNN model, collect their practical energy consumption from hardware respectively; use polynomial regression (PR) to analyze the relationship between DNN model TOs and energy consumption for current hardware; input the TOs of new target DNN models into the trained TOs-based PR model to predict their energy consumption

calculation logic, in terms of BOs. Operations addition, subtraction, multiplication, division and root operations are used as the categories of BOs, for the reason that higher-level calculations (e.g., activation of neuron with Sigmoid as AF function - see Function 4.3) are assembled by these five BOs at the software level. Finally, the number of transistor operations theoretically involved in each basic operation processed are analyzed by ALU calculation logic (translate BOs into TOs), and calculate the layer-based TOs based on the layer-based BOs information and data types (e.g., FP-16, FP-32).

Suppose there is a target DNN model to be analyzed, the layer list extracted from q is $L = \{l_1, l_2, ..., l_i, ..., l_o\}$ (l_o is the output layer). C represent the basic operation calculator, which takes a model layer as input and calculate how many basic operations are needed in this layer. T represent the TOs translator, which takes the number of five basic operations as input, and calculate how many TOs are needed to process all the input basic operations.

The calculation of theoretical TOs for DNN model q could be shown as equation 4.1.

$$\text{Target Model TOs} = \sum_{i=1}^{\text{o}} T(C(l_i)) \tag{4.1}$$

The functions used in BOs and TOs calculations are introduced in the following subsections, frequently used notations are listed in Table 4.2.

Table 4.2: List of Notations

L	Layer list of DNN model
l_i, l_{o}	The ith/output layer of DNN model
$C_{\text{f}}, C_{\text{bp}}, C_{\text{loss}}$	BOs calculator for layer forwarding, backpropagation and loss calculation
$n_{\text{add}}, n_{\text{sub}}, n_{\text{mul}}$, $n_{\text{div}}, n_{\text{root}}$	Number of addition, subtraction, multiplication division and root operation
$B_{\text{f}}, B_{\text{bp}}$	Layer-wise basic operations list of DNN model
$b_{\text{fk}}, b_{\text{bpk}}, b_{\text{loss}}$	Basic operations of ith/output layer
$BOs_{\text{Potential}}$ $BOs_{\text{Activation}}$ $BOs_{\text{Convolution}}$	Represent the number of five basic operations for potential/activation/convolution calculation in a layer
x	Neuron input
w	Weight
ξ	Neuron potential
y	Neuron output
$bias$	Bias
$c_{\text{in}}, c_{\text{out}}$	Number of input/output channel
m	Convolutional layer output window width
k	Convolutional kernel width
add/sub/mul/ div/root	One addition/subtraction/multiplication/ division/root operation
I	Number of input dimension
O	Number of output dimension
$T, T_{\text{add}}, T_{\text{sub}}$, $T_{\text{mul}}, T_{\text{div}}, T_{\text{root}}$	Transistor operations calculator (sub-index: calculator for different operations)
$D_{\text{f}}, D_{\text{bp}}$	Layer-wise transistor operations list
$d_{\text{fk}}, d_{\text{bpk}}, d_{\text{loss}}$	Layer-wise transistor operations
$d_{\text{f\_total}}, d_{\text{bp\_total}}$	Total transistor operations for model forwarding/ backpropagation
d_{total}	Total transistor operations for model forwarding, loss calculation and backpropagation
p	Data type

4.3.1 Layer Structure Based Basic Operations (BOs)

The BOs of a DNN model are calculated according to the processing logic of each layer, related to Fig. 4.3*a*: *Calculate basic operations*, details are shown in Alg. 1. For each

layer $l_i \in L$, the BOs calculator $C_{\mathrm{f}}(l_i)$ and $C_{\mathrm{bp}}(l_i)$ analyze the function of the layer l_i by its structure; translate the layer function into calculation tasks (refers to Fig. 4.1*a* - ALU Calculation Tasks); and further translate calculation tasks into the number of five BOs needed for layer forward and backpropagation respectively. $C_{\mathrm{loss}}(l_{\mathrm{output}})$ takes only the output layer l_{output} structure as input to calculate BOs for loss calculation. The calculated BOs information by $C_{\mathrm{f/bp/loss}}$ will be stored and return as a list (shown in equation 4.2). Here, n with sub-index add, sub, mul, div and root means the number of each basic operation (fixed order: addition, subtraction, multiplication, division and root)).

$$C_{\mathrm{f/bp/loss}}(l) = [n_{\mathrm{add}}, n_{\mathrm{sub}}, n_{\mathrm{mul}}, n_{\mathrm{div}}, n_{\mathrm{root}}] \tag{4.2}$$

Algorithm 1 Calculate layer based BOs for DNN model

Require: DNN model structure $L = \{l_1, l_2, ..., l_{\mathrm{o}}\}$
Require: Basic operation calculators: $C_{\mathrm{f}}, C_{\mathrm{bp}}, C_{\mathrm{loss}}$
  Initialise $B_{\mathrm{f}} = [b_{\mathrm{f1}}, b_{\mathrm{f2}}, ..., b_{\mathrm{fo}}]$; $B_{\mathrm{bp}} = [b_{\mathrm{bp1}}, b_{\mathrm{bp2}}, ..., b_{\mathrm{bpo}}]$
  Initialise $b_{\mathrm{loss}} = [0,0,0,0,0]$ /* number of 5 BOs */
  for $l_i \in [1, 2, ..., \mathrm{o}]$ **do**
    $b_{\mathrm{f}i} = C_{\mathrm{f}}(l_i)$
    $b_{\mathrm{bp}i} = C_{\mathrm{bp}}(l_i)$
  end for
  $b_{\mathrm{loss}} = C_{\mathrm{loss}}(l_{\mathrm{o}})$ **return** $B_{\mathrm{f}}, B_{\mathrm{bp}}, b_{\mathrm{loss}}$

Algorithm 1 shows how to extract layer-wise BOs information from the target DNN model, the algorithm will return: layer-wise forward BOs in B_{f}; layer-wise BP BOs in B_{bp}; and BOs for loss calculation b_{loss}. This section provides two examples that calculate the forward BOs for a full-connection layer in feedforward DNN, and a convolutional layer in CNN respectively to show how BOs is analyzed from layer structure. The calculation of BOs for loss and model backpropagation is similar to model forward but follows their individual calculation logic.

Feedforward DNN

The calculation logic for full connection layers is proposed in [119]. Suppose full-connection layer $l_{\text{Full-connection}}$ have I inputs; O outputs; use Sigmoid as AF, the calculation of output y_j on input x, weight w and bias *bias* could be demonstrated in equation. Here, ξ represents neuron output before activated by AF, and S is Sigmoid function. The BOs for forward on one data instance could be calculated by equation 4.4:

$$\begin{aligned} &\text{Potential of } j\text{-th Neuron:} \xi_j = \text{bias}_j + \sum_{i=1}^{I} (w_{i,j} \times x_i) \\ &\text{Sigmoid AF of } j\text{-th Neuron:} y_j = S(\xi_j) = 1/1 + e^{-\xi_j} \end{aligned} \tag{4.3}$$

$$\begin{aligned} &C_{\text{f}}(l_{\text{Full-connection}}) = BOs_{\text{Potential}} + BOs_{\text{Activation}} \\ &BOs_{\text{Potential}} = O \times I(\text{mul} + \text{add}) \\ &BOs_{\text{Activation}} = O \times (\text{sub} + \text{add} + \text{div} + \text{root}) \\ &C_{\text{f}}(l_{\text{Full-connection}}) = [(I+1)O, O, IO, O, O] \end{aligned} \tag{4.4}$$

Here, the counting of BOs is analyzed from the logic of equations. For example, the calculation of ξ_j in equation 4.3 involves an accumulation of multiplication results, and an addition operation for adding bias. The multiplication operation happens I times; addition operation by accumulation repeat $I-1$ times; add bias need one additional addition operation; the total operations are I multiplications and $(I-1)+1$ additions. The calculation of ξ will repeat the process above O times, so the BOs for linear operations are $O \times I$ multiplications and additions. In equation 4.4, 'mul + add' means 1 multiplication operation and 1 addition operation. The items in the return list from $C_{\text{f}}(l_{\text{Full-connection}})$ means the number of each basic operation needed for forwarding this layer (the order is detailed before).

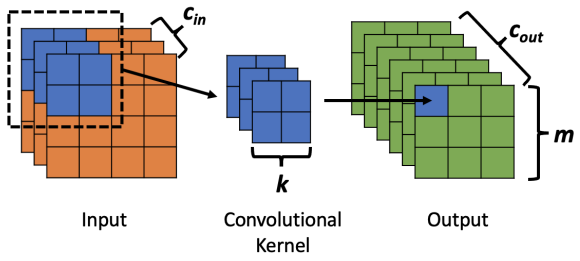


Figure 4.4: Convolutional Layer

CNN

The working process of convolutional layers could be seen as Fig. 4.4. Suppose $m, k, c_{\text{in}}, c_{\text{out}}$ are the output window width, convolutional kernel size, number of input channels and number of output channels in convolutional layer $l_{\text{Convolutional}}$. With GELU (approximate as $\text{GELU}(x) = 0.5 \times x \times (1 + \text{Tanh}(sqrt(2/\pi) \times (x + 0.044715 \times x^3)))$ [101] in PyTorch) as AF, the convolutional layer forward BOs on one input instance (e.g., one image) could be calculated as Eq. 4.5:

$$
\begin{aligned}
C_{\text{f}}(l_{\text{Convolutional}}) &= BOs_{\text{Convolution}} + BOs_{\text{Activation}} \\
BOs_{\text{Convolution}} &= m^2 \times c_{\text{out}} \times c_{\text{in}} \times k^2(\text{mul} + \text{add}) \\
BOs_{\text{Activation}} &= m^2 c_{\text{out}}(\text{sub} + \text{add} + \text{div} + \text{root} + 2 \times \text{mul}) \\
n_{\text{add}} &= m^2 c_{\text{out}}(1 + c_{\text{in}} k^2) \\
n_{\text{sub}} &= m^2 c_{\text{out}} \\
n_{\text{mul}} &= m^2 c_{\text{out}}(2 + c_{\text{in}} k^2) \\
n_{\text{div}} &= m^2 c_{\text{out}} \\
n_{\text{root}} &= m^2 c_{\text{out}} \\
C_{\text{f}}(l_{\text{Convolutional}}) &= [n_{\text{add}}, n_{\text{sub}}, n_{\text{mul}}, n_{\text{div}}, n_{\text{root}}]
\end{aligned}
\tag{4.5}
$$

4.3.2 Basic Operations, Data Type and Theoretical TOs

The BOs information of the target DNN model will be further decomposed into transistor level to unify the energy metric of different BOs with TOs. As shown in Alg. 2, TOs calculator $T(b,p)$ takes both the BOs extracted by Alg. 1 and the data type p used in DNN model as inputs. The reason is that at different data precisions, different numbers of transistor operations are required to do the same operation. T opens the design logic of ALU, analyze the theoretical TOs for each operation modules (e.g., 32-bit adder, 16-bit multiplier) with their individual integrated circuit (IC) designs (the analysis method is introduced in the following subsections). With the information of layer BOs, T is used to translate BOs into theoretical TOs for DNN model in different steps (e.g., validation). Function T could be universally applied on the return lists of $C_{\mathrm{f}}(l_i)$, $C_{\mathrm{bp}}(l_i)$ and $C_{\mathrm{bp}}(l_i)$ due to their similar data format (number of five basic operations with fixed order). Suppose b is a piece of BOs information extracted from the target DNN model by C, equation 4.6 demonstrate how b is processed into TOs with T. The calculation of DNN TOs from BOs information could be seen in Alg. 2, the algorithm will return layer-wise model forward TOs in D_{f} and BP TOs in D_{bp}, as well as cumulative TOs for model forward $d_{\mathrm{f\_total}}$, BP $d_{\mathrm{bp\_total}}$, loss d_{loss}, and overall TOs d_{total}.

$$
\begin{aligned}
b =& [n_{\mathrm{add}}, n_{\mathrm{sub}}, n_{\mathrm{mul}}, n_{\mathrm{div}}, n_{\mathrm{root}}] \\
T(b,p) =& T_{\mathrm{add}}(n_{\mathrm{add}},p) + T_{\mathrm{sub}}(n_{\mathrm{sub}},p) + T_{\mathrm{mul}}(n_{\mathrm{mul}},p) \\
&+ T_{\mathrm{div}}(n_{\mathrm{div}},p) + T_{\mathrm{root}}(n_{\mathrm{root}},p)
\end{aligned}
\tag{4.6}
$$

BOs, Operation Units in ALU and Calculation of TOs

In TOs calculator, the calculation of TOs from BOs depends on the inner logic of ALU (open ALU and get number of logic gates, and open logic gates with transistors to count TOs). The various IC design for ALU inner logic follows the role of making a trade-off

Algorithm 2 Calculate theoretical TOs based on BOs

Require: Layer based BOs set $B_{\mathrm{f}} = [b_{\mathrm{f1}}, b_{\mathrm{f2}}, ..., b_{\mathrm{fo}}]; B_{\mathrm{bp}} = [b_{\mathrm{bp1}}, b_{\mathrm{bp2}}, ..., b_{\mathrm{bpo}}]$
Require: BOs for loss calculation b_{loss}
Require: Data type p
Require: TOs calculator T
  Initialise $D_{\mathrm{f}} = [d_{\mathrm{f1}}, d_{\mathrm{f2}}, ..., d_{\mathrm{fo}}]; D_{\mathrm{bp}} = [d_{\mathrm{bp1}}, d_{\mathrm{bp2}}, ..., d_{\mathrm{bpo}}]$
  Initialise $d_{\mathrm{f\_total}}, d_{\mathrm{bp\_total}}, d_{\mathrm{loss}}, d_{\mathrm{total}} = 0, 0, 0, 0$
  for $i \in [0, 1, ..., \mathrm{o}]$ **do**
    $d_{\mathrm{f}i} = T(b_{\mathrm{f}i}, p)$
    $d_{\mathrm{bp}i} = T(b_{\mathrm{bp}i}, p)$
    $d_{\mathrm{f\_total}} += T(b_{\mathrm{f}i}, p)$
    $d_{\mathrm{bp\_total}} += T(b_{\mathrm{bp}i}, p)$
  end for
  $d_{\mathrm{loss}} = T(b_{\mathrm{loss}}, p)$
  $d_{\mathrm{total}} = d_{\mathrm{f\_total}} + d_{\mathrm{bp\_total}} + d_{\mathrm{loss}}$ **return** $D_{\mathrm{f}}, D_{\mathrm{bp}}, d_{\mathrm{f\_total}}, d_{\mathrm{bp\_total}}, d_{\mathrm{loss}}, d_{\mathrm{total}}$

between time complexity (the time delay) and space complexity (number of transistors). The achieve of lower-delay ALU needs additional hardware logic gates which usually cost more transistors and energy [120] (e.g., adder: ripple-carry and carry look-ahead adder [121]; multiplier: Wallace and Booth-Wallace multiplier [122]). The logic of IC in ALU support different types of basic operations, however, not all the basic operations have their independent operation unit (e.g., adder is designed for addition, but subtraction is processed in adder by 2's complement). As multipliers are always built with adder as basic units, the changing of adder IC has a limited impact on the relationship between transistors in adder and multiplier, similar for other operation units. Although independent root operation units exist, they are not widely embedded in current PC processors. Root operation is always simulated with the Newton-Raphson method [123] by operation units in ALU. The TOs model in this chapter uses the basic IC design to calculate the theoretical transistors needed by BOs. Please note, the design of IC is not focused in this chapter, using the specific hardware IC may increase the analysis accuracy of model energy scaling on that hardware. Theoretically, transistors needed in adder follows the linear relationship to bit-length, multiplier and divider follow exponential relationship. Ten transistors are used for a NOR-XNOR gates based 1-bit full-adder as proposed in [124]. And according to the IC design in [90], transistors used for the 64-bit Booth-Wallace multiplier and SRT

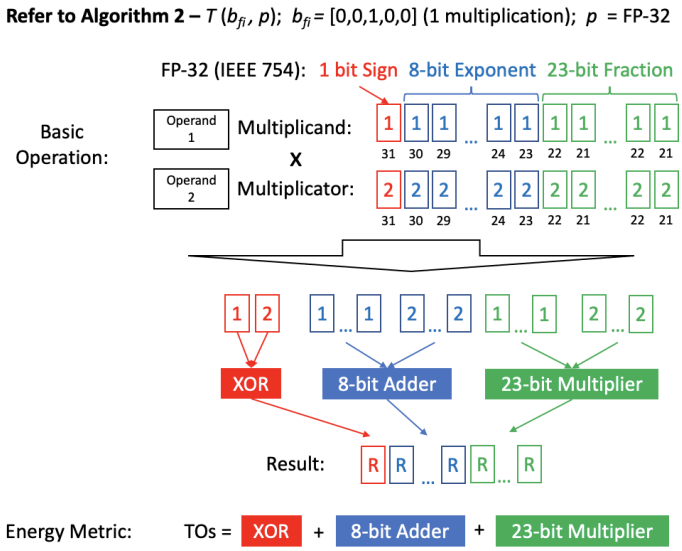


Figure 4.5: Calculation of Theoretical TOs based on BOs (Example: Multiplication Operation on Two IEEE 754 FP-32 Numbers)

divider are 90k and 110k respectively.

Floating-point Numbers in TOs Calculation

Floating-point numbers (FP) are the most generally used data type in DNNs. As FP used in majority programming language follows IEEE 754 standard, the calculation of theoretical TOs should consider the structure and calculation logic of of binary FP with different precision (e.g., FP-32 known as single-precision floating-point, FP64 as double-precision floating-point). According to IEEE 754, each FP-32 number contains an 1-bit sign, an 8-bit exponent and a 23-bit fraction (FP-64: 1-bit sign, 11-bit exponent and 52-bit fraction). Theoretically, adding two FP-32 numbers will need a 24-bit adder (23 full-adder and 1 half-adder without considering bit shift in the exponent). As shown in Fig. 4.5, the multiplication/division of two FP-32 numbers will need a XOR gate (for sign), a 24-bit multiplier/divider (fraction calculation) and an 8-bit adder (exponent calculation). It means the TOs for a FP-32 multiplication is theoretically be calculated by the sum of transistors for a 24-bit multiplier, a 8-bit adder and a XOR gate. As a theoretical model, transistors redundancy in practical hardware are not considered (e.g., multiplication of two 24-bit number on 32-bit multiplier).

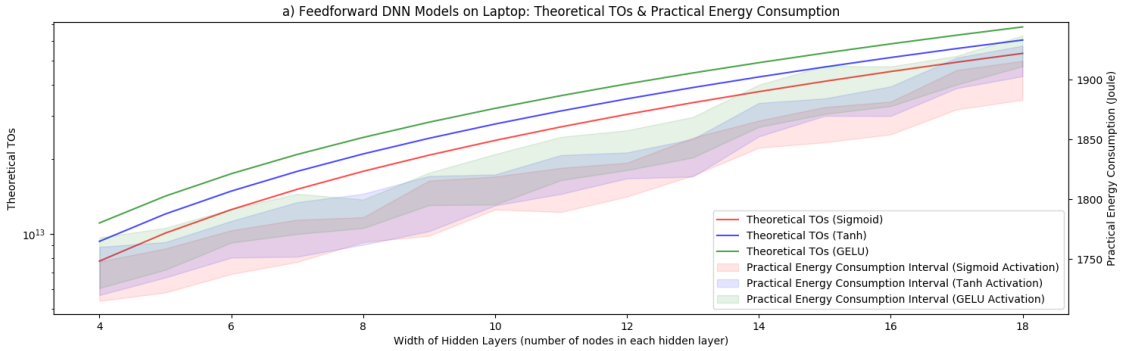
4.3.3 TOs Model and Energy Scaling

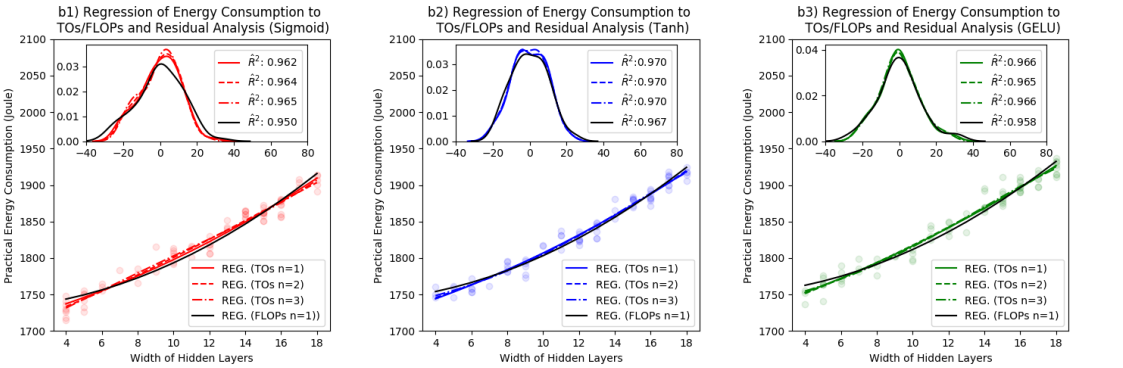
Theoretical TOs for a given DNN model could be calculated by Alg. 1 and Alg. 2. The process of mapping the scaling of DNN model TOs to its practical energy consumption scaling is demonstrated in sub-figure Fig. 4.3*b*. Firstly, a list of DNN models with different configurations will be established. Secondly, calculate individual TOs of DNN models respectively with the TOs model, and deploy them on practical hardware for their practical energy consumption data. According to the theory in this chapter, polynomial regression (PR) with different number of coefficients will be used to fit the relationship between the practical energy consumption and TOs of DNN models. To analyze the scaling of energy with a DNN design factor (e.g., width of hidden layers), a list of models will be generated by gradually changing the factor (e.g., models with 4,5,6 and 7 hidden layer width). Then, calculate their TOs and estimate their individual energy consumption by the previously fitted PR model. I demonstrate the energy scaling of a feedforward DNN model with different hidden layer widths and AFs in the next section, followed by that of a CNN model with different layer configurations. Please note that the practical energy consumption of models depends on different choices of hardware. If the hardware platform changes, the PR model should be retrained on energy data collected from the new hardware platform.

4.4 Method Verification

To verify the TOs model, experiments are designed to analyze the training energy scaling of 1) a feedforward DNN set by changing the AFs and width of hidden layers; 2) a CNN set by changing the number of convolutional layers and kernel size in each layer. The structural plausibility of DNN models (overfitting may occur with large network size) is not considered, as they do not affect energy consumption. The experiment settings could be seen from Table 4.3, data statement please refer to the Appendix.

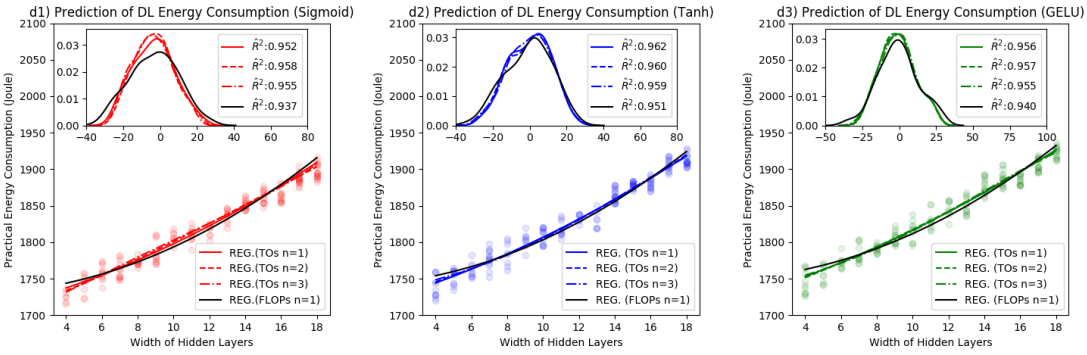
The practical energy consumption of DNN models is collected with *Intel Power-*





c) Training of Polynomial Regression (Energy to TOs/FLOPs): Coefficient and Performance Metrics

	n = 1	RMSE	$\hat{R}^2$	n = 2	RMSE	$\hat{R}^2$	n = 3	RMSE	$\hat{R}^2$
TOs (sigmoid)	1.71e+3, 3.65e-12	10.857	0.962	1.69e+3, 5.00e-12, -2.30e-26	10.480	0.964	1.69e+3, 5.55e-12, -4.31e-26, 2.15e-40	10.332	0.965
FLOPs (sigmoid)	1.73e+3, 8.42e-8	12.518	0.950	1.72e+3, 1.31e-7, -2.21e-17	10.852	0.962	1.71e+3, 1.64e-07, -5.69e-17, 1.03e-26	10.449	0.964
TOs (tanh)	1.71e+3, 3.34e-12	9.302	0.970	1.70e+3, 4.03e-12, -7.74e-27	9.273	0.970	1.71e+3, 3.23e-12, -1.80e-26, 2.43e-40	9.180	0.970
FLOPs (tanh)	1.74e+3, 8.62e-8	9.718	0.967	1.72e+3, 1.30e-7, -1.81e-17	9.178	0.970	1.72e+3, 1.39e-07, -2.76e-17, 2.83e-27	9.175	0.970
TOs (gelu)	1.72e+3, 3.04e-12	10.598	0.966	1.71e+3, 3.51e-12, -6.46e-27	10.567	0.965	1.73e+3, 2.01e-12, -3.59e-26, 3.53e-40	10.429	0.966
FLOPs (gelu)	1.75e+3, 8.84e-8	11.719	0.958	1.73e+3, 1.29e-7, -1.96e-17	10.454	0.966	1.73e+3, 1.31e-07, -2.19e-17, 6.88e-28	10.440	0.966



e) Validation of Polynomial Regression (Energy to TOs/FLOPs): Performance Metrics

	RMSE (n=1)	$\hat{R}^2$ (n=1)	RMSE (n=2)	$\hat{R}^2$ (n=2)	RMSE (n=3)	$\hat{R}^2$ (n=3)
TOs (sigmoid)	11.632	0.952	10.865	0.958	11.157	0.955
FLOPs (sigmoid)	13.349	0.937	10.921	0.957	11.090	0.956
TOs (tanh)	11.146	0.962	11.313	0.960	11.408	0.959
FLOPs (tanh)	12.610	0.951	11.403	0.960	11.427	0.959
TOs (gelu)	11.731	0.956	11.555	0.957	11.779	0.955
FLOPs (gelu)	13.610	0.940	11.731	0.955	11.853	0.954

Figure 4.6: Method Verification (Feedforward DNNs on Laptop CPU)

Exp. Settings	**Feedforward DNN**	**CNN**
Dataset	Banknote [125]	Drone images
Dataset Length	1372 instances	300 images
Data Structure	4 inputs & 1 output	256*256*3 (RGB)
Model Type	Feed-forward	Convolutional
Model Depth	3 hidden layers	2-10 C-layers
DNN Width	4-18 nodes per layer	-
CNN Kernel (h&w)	-	layer 2-6: 3
	-	layer 7-10: 4,5,6,7
CNN Channels	-	32
Activation Functions	Sigmoid, Tanh, GELU	GELU
Batch Size	64	64 (GPU: 16)
Epochs	2k (laptop CPU)	25 (laptop CPU)
	5k (desktop CPU)	20 (desktop CPU)
		100 (desktop GPU)
DNN Framework	PyTorch	PyTorch
Hardware	**Laptop**	**Desktop**
CPU	Intel Core i9	AMD RYZEN 9
GPU	-	Nvidia 3080
OS	macOS Monterey	Windows 10

Table 4.3: Experiment Settings and Hardware

Gadget software [126] (for Intel Architecture CPU[127]), *OpenHardwareMonitor* (for AMD CPU) and *TechPowerUp GPU-Z* (for GPU). These tools are designed using on-chip energy sensors to collect the instantaneous power of processor cores, DRAM, and overall CPU/GPU package separately with timestamps [126]. The energy consumption is calculated based on the integration of the instantaneous power consumption over time. Certain errors could exist due to the accuracy of on-chip sensor for instantaneous readings reported in [128]. However, as the aim of this chapter is to analyze only calculation energy cost, it is the only way for collecting on-chip component energy data [112, 129] and reported to be fairly accurate by NVIDIA (+-5% error rate [130]) and authors in [131, 132]. As the problem does not influence theoretical TOs/FLOPs, this chapter will not discuss the accuracy of hardware energy monitoring. During the experiment, the program is forced to run with a single CPU/GPU core as the reason mentioned in research limitations - core communications energy cost generated by parallel running could significantly influence the relationship between DNN calculation tasks and practical energy

consumption.

4.4.1 Energy Consumption Scaling of Feedforward DNNs on Laptop CPU

As shown in Fig. 4.6*a*, this experiment calculates TOs for feed-forward DNNs with 4-18 width (number of nodes in each hidden layer) and uses *Sigmoid*, *GELU* and *Tanh* as AFs respectively. 50 experiments are made (cool the hardware between each run to maintain the repeatability of experiment) on each model setting with laptop CPU as introduced in Table 4.3, drop the highest and lowest 5 data, and demonstrate the practical energy consumption interval of each DNN. The collected energy data for each DNN are randomly split into training sets (60%) and validation sets (40%). They are used to train and validate the PR models mapping the relationship between DNN TOs/FLOPs and practical energy consumption. I demonstrate the training of PR models based on FLOPs (FLOPs-based PR model) and TOs (TOs-based PR model) and with 1-3 coefficients respectively. I conduct a residual analysis to different PR models, the result could be seen in Fig. 4.6*b1-b3* (only demonstrate 1 coefficient FLOPs-based PR in figures for higher clearness and readability, statistics for 2/3 coefficients FLOPs-based PR are listed in on-figure tables). In these figures, each point is one data instance collected from one experiment. Both FLOPS-based and TOs-based PR models perform fairly excellent - their residuals following Gaussian distribution. To provide statistical performance analysis of PR models, metric adjusted R-square and RMSE are used, as shown in Fig. 4.6*c* (will be explained in Result Discussion). I further demonstrate the validation of PR models in Fig. 4.6*d1-d3*, and list the performance metric in Fig. 4.6*e*. I also run the DNN set on a desktop CPU, please refer to Fig. 1 in the Appendix.

According to the coefficients shown in Fig. 4.6*c* and statistic of DNN on desktop CPU (Fig. 1*c* in the Appendix), use data of Sigmoid with 1 coefficient (*n=1*) for example, the relationship between DNN model's TOs and the practical energy consumption E in

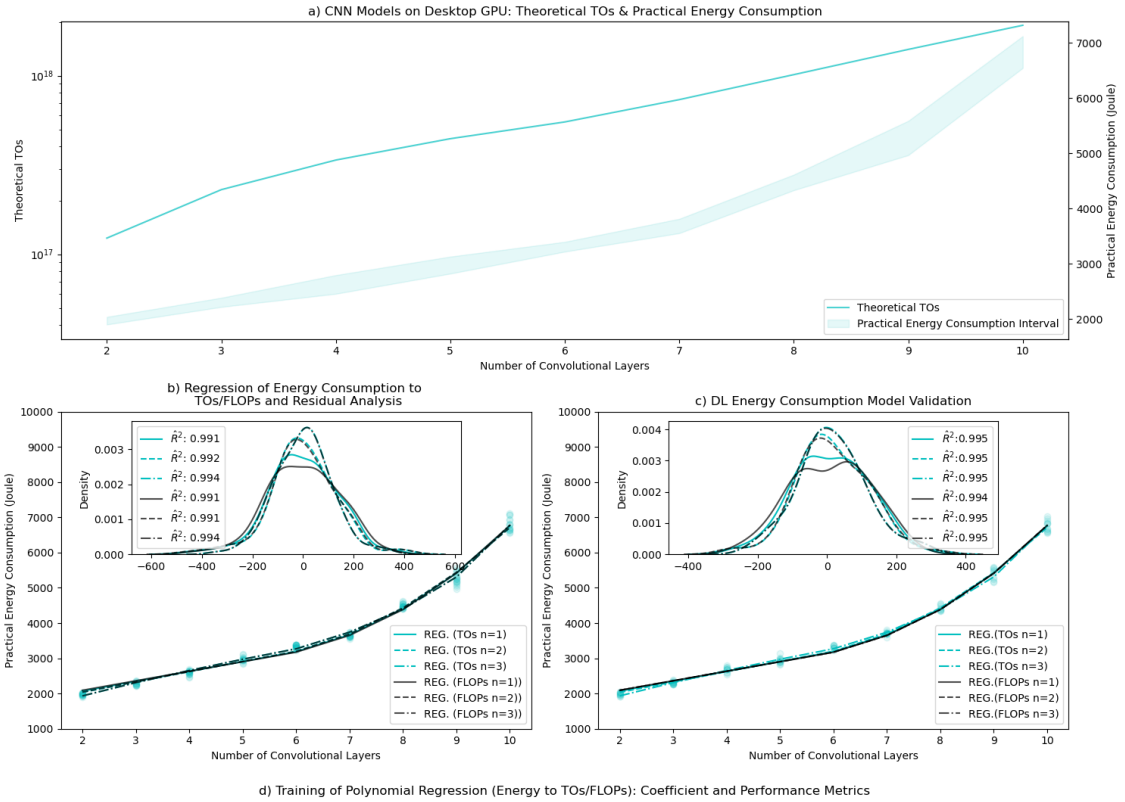
current laptop CPU and desktop CPU could be demonstrated by Eq.4.7.

$$\begin{aligned} &\text{DNN on laptop CPU:} \\ &\mathrm{E} = 1.71 \times 10^{3} + 3.65 \times 10^{-12} \times \mathrm{TOs} \\ &\mathrm{E} = 1.73 \times 10^{3} + 8.42 \times 10^{-8} \times \mathrm{FLOPs} \\ &\text{DNN on desktop CPU:} \\ &\mathrm{E} = 9.96 \times 10^{2} + 2.47 \times 10^{-12} \times \mathrm{TOs} \\ &\mathrm{E} = 1.04 \times 10^{3} + 5.71 \times 10^{-8} \times \mathrm{FLOPs} \end{aligned} \tag{4.7}$$

4.4.2 Energy Consumption Scaling of CNN on Desktop GPU

As shown in Fig. 4.7*a*, I calculate TOs for CNN models with different depths (number of convolutional layers) and apply *GELU* as the AF for each convolutional layer. The configuration of each CNN model could be seen from Table 4.3. 50 experiments are made for each model, drop the highest and lowest 5 data, and demonstrate the practical energy consumption interval of each model. The energy data is split randomly into training set and validation set with a rate of $60/40\%$. The training and validation of PR energy models with residual analysis are demonstrated in Fig. 4.7*b-c*. I also summarise the PR model performance metric (adjusted R-square and RMSE) in Fig. 4.7*d-e*. I also run the same CNN set on a desktop CPU and a laptop CPU separately, please refer to Fig. 2 and Fig. 3 in the Appendix for your interest.

According to the coefficients showed in Fig. 4.7*c* and statistic of CNN on desktop CPU (Fig. 2*c* in the Appendix), the relationship (PR models with *n=1*) between CNN model's TOs and the practical energy consumption *E* in current desktop processors could



d) Training of Polynomial Regression (Energy to TOs/FLOPs): Coefficient and Performance Metrics

	n = 1	RMSE	$\hat{R}^2$	n = 2	RMSE	$\hat{R}^2$	n = 3	RMSE	$\hat{R}^2$
TOs	1.76e+3, 2.61e-15	134.266	0.991	1.69e+3, 2.82e-15, -1.04e-34	131.907	0.992	1.53e+3, 3.74e-15, -1.29e-33, 4.00e-52	111.769	0.994
FLOPs	1.81e+3, 3.49e-11	139.645	0.991	1.72e+3, 3.89e-11, -2.68e-26	134.598	0.991	1.55e+3, 5.19e-11, -2.56e-25, 1.05e-39	111.672	0.994

e) Validation of Polynomial Regression (Energy to TOs/FLOPs): Performance Metrics

	RMSE (n=1)	$\hat{R}^2$ (n=1)	RMSE (n=2)	$\hat{R}^2$ (n=2)	RMSE (n=3)	$\hat{R}^2$ (n=3)
TOs	107.923	0.995	103.700	0.995	100.391	0.995
FLOPs	113.994	0.994	105.624	0.995	100.812	0.995

Figure 4.7: Method Verification (CNN on Desktop GPU). The potential reason why statistics for FLOPs and TOs are close when n=3: Although several runs are made for each DL model structure, the statistics are still scarce, and higher-flexibility PR models with larger n could easily fit the statistics. Roughly, I think the workload quantified by any metric (FLOPs/MACs/TOs) should follow a linear relationship to the energy consumption, and the result of n=1 is more convincing.

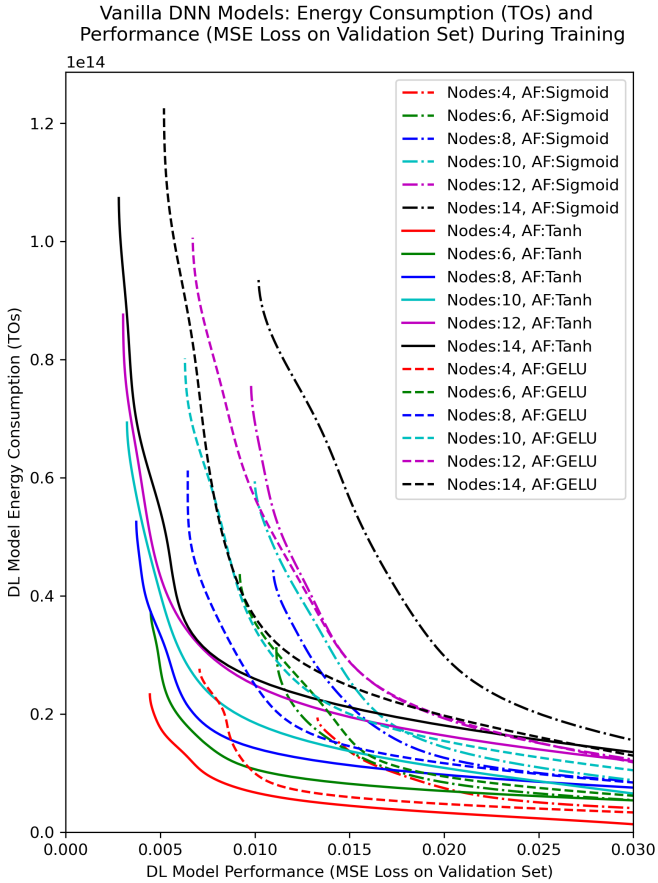


Figure 4.8: TOs Energy Consumption and Performance of Feedforward DNNs

be demonstrated by Eq.4.8.

$$
\begin{aligned}
&\text{CNN on desktop GPU:} \\
&\mathrm{E} = 1.76 \times 10^{3} + 2.61 \times 10^{-15} \times \mathrm{TOs} \\
&\mathrm{E} = 1.81 \times 10^{3} + 3.49 \times 10^{-11} \times \mathrm{FLOPs} \\
&\text{CNN on desktop CPU:} \\
&\mathrm{E} = 2.51 \times 10^{3} + 4.29 \times 10^{-14} \times \mathrm{TOs} \\
&\mathrm{E} = 2.67 \times 10^{3} + 5.73 \times 10^{-10} \times \mathrm{FLOPs}
\end{aligned}
\tag{4.8}
$$

4.4.3 Result Discussion

During the verification, each experiment with individual settings (feedforward DNNs/CNNs on different hardware processors) is run 50 times, to find the practical energy consumption interval and analyze the robustness of energy scaling models (see - sub-figure *a* in Fig. 4.6, 4.7 and Fig. 1, 2, 3 in the Appendix).

This chapter analyzes and compares the performance of TOs-based and FLOPs-based PR models in estimating DNN energy consumption by Adjusted R-square Error ($\hat{R}^2$) and Root Mean Square Error (RMSE). $\hat{R}^2$ represents the fraction of variance of the actual value of the response variable captured by the regression model, with penalty in the number of variables[133]. RMSE represents the differences between values predicted by a model or an estimator and the values observed [134]. From the statistic of $\hat{R}^2$ and RMSE (see sub-figures c, e in Fig. 4.6, and d, e in Fig. 4.7), both TOs-based and FLOPs-based PR models could accurately fit DNN energy metric and practical energy consumption ($\hat{R}^2$ for FLOPs/TOs-based PR model: 0.95-0.99/0.96-0.99), while TOs-based PR models have equal or better performance in $\hat{R}^2$ and RMSE than FLOPs-based PR models. This means, TOs is more accurate to be used as a metric for analysing the scaling law of DNN energy consumption. As validation results apply polynomial regression with 1-3 coefficients to fit the relationship between FLOPs/TOs and practical energy consumption, according to $\hat{R}^2$, sometimes the relationship shows more linear features and is more explainable. For example, as in function 4.7, 4.8, the energy cost of each TO in CNN models on the GPU can be approximated to be an order of magnitude lower than that of CPU (desktop GPU: 2.61×10^{-15} Joule; desktop CPU: 4.29×10^{-14} Joule). Simultaneously, the performance of FLOPs-based and TOs-based PR models for DNN energy estimation in Joule could be summarised in Table 4.4. Compared with FLOPs-based PR models, TOs-based PR models achieve $0.14 - 2.56\%$ higher precision on DNN energy consumption estimation, and a 10% lower average estimation error in Joule.

This chapter also demonstrates DNN energy consumption over performance in Mean Squared Error (MSE) with different configurations (banknotes identification [125]) in

Desktop		
DNN: AFs	**FLOPs**	**TOs**
Sigmoid (%)	96.93-99.99 (avg 99.05)	97.46-99.99 (avg 99.14)
Tanh (%)	96.81-99.97 (avg 99.04)	96.95-99.99 (avg 99.22)
GELU (%)	97.11-99.97 (avg 98.94)	97.52-99.99 (avg 99.26)
Sigmoid (avg/max, J)	11.16/36.05	10.19/31.29
Tanh (avg/max, J)	11.33/41.28	9.33/39.51
GELU (avg/max, J)	12.78/35.55	9.04/29.45
CNN: Processor		
CPU (%)	78.41-99.92 (avg 94.13)	80.97-99.97 (avg 94.82)
GPU (%)	91.50-99.92 (avg 97.15)	92.61-99.98 (avg 97.36)
CPU (avg/max, J)	383.39/797.24	341.46/747.77
GPU (avg/max, J)	101.65/364.33	95.95/362.66
Laptop		
DNN: AFs	**FLOPs**	**TOs**
Sigmoid (%)	98.14-99.99 (avg 99.41)	98.52-99.99 (avg 99.49)
Tanh (%)	97.95-99.98 (avg 99.45)	98.44-99.99 (avg 99.50)
GELU (%)	97.99-99.99 (avg 99.46)	98.40-99.99 (avg 99.51)
Sigmoid (avg/max, J)	10.59/31.80	9.31/25.56
Tanh (avg/max, J)	9.90/35.27	9.19/26.77
GELU (avg/max, J)	9.85/34.57	9.19/30.47
CNN: Processor		
CPU (%)	74.03-99.90 (avg 92.93)	76.45-99.99 (avg 93.61)
CPU (avg/max, J)	501.09/1134.58	453.25/1029.06

Table 4.4: The Performance of FLOPs-based and TOs-based PR Models in Estimating the Energy Consumption of Different DL Models on Different Hardware Platforms (Precision, Average Error and Max Error)

Fig. 4.8, from where the energy efficiency of DNN model configurations (number of nodes and the choice of AFs) could be analyzed. From the figure, DNNs with Tanh as AF can converge to lower MSE loss within a certain energy cost than with GELU and Sigmoid (e.g., Tanh/GELU/Sigmoid can achieve: 0.004, 0.008, 0.013 MSE loss in 0.2×10^{14} TOs). At the same time, with AF fixed, the energy cost to train a DNN model to a certain MSE loss increases with network size (e.g., DNN with 4/8/12 width need: $0.17/0.32/0.45 \times 10^{14}$ TOs to 0.005 MSE loss). However, larger DNNs can converge to a lower MSE loss than light networks, with a significantly increased energy cost.

FLOPs/MACs/TOs are theoretical metrics for DNN complexity/energy consumption analysed from DNN structures and configurations. With support to nonlinear operations, TOs is more complex to be calculated than FLOPs/MACs. Compared with methods in [110, 111], TOs method is more efficient due to no practical experiment required, but not applicable to black box models.

4.5 Conclusion and Further Works

The energy consumption of nonlinear operations in DNNs has not been well analyzed and modelled, resulting in an incomplete understanding of how DNN energy consumption scales with model complexity. In this chapter, I propose a bottom-up theoretical TOs method to expose the role of nonlinear activation functions and neural network structure in DNN energy consumption. This chapter shows that 1) with single core running, theoretical TOs of DNN shows a strong empirical polynomial relationship with its practical energy, and could be used for analysing the energy scaling of DNN models; 2) the proposed method (average precision 93.61-99.51%) outperforms FLOPs-based method with $0.14-2.56\%$ higher precision on DNN energy consumption estimation, and lower 10% of the average estimation error; and 3) the scaling relationships in this chapter are less prone to measurement errors than absolute energy consumption estimates.

The impact of proposed TOs-based approach is that developers can analyze the energy scaling of different operations in DNNs, thus developing more energy-efficient DNN structures and configurations. It is believed that TOs could be extended to all DNNs through more comprehensive research in different algorithm logic (e.g., automatic differentiation in PyTorch [135]) and processing mechanism of other operations, e.g., comparison operators.

In future work, if I combine the proposed TOs-based and the data movement-based [105] energy estimation methods, I can build a more holistic and accurate DNN energy consumption framework. Furthermore, by combining the proposed TOs with state-of-the-art multi-core energy consumption modelling approaches, I can map the scaling law of DNN energy consumption in multi-core environments. I intent to apply this to machine learning techniques used in widespread communication [80] and IoT architectures [76] to have a significant impact.

4.6 Appendix

4.6.1 Review of Calculation Hardware - Data Storage and Processors

Mathematical operations involve two main types of hardware: multiple-levels of data storage (e.g., Disk, DRAM, Cache) and processors (e.g., CPU, GPU, TPU) for data storage/access and calculation respectively. Commercial processors use ALU as the fundamental calculation unit, with different configuration of inner components (e.g., ALU, control units) to achieve different features [136]. An operation could be processed in different types of processor, and resulting different time cost and energy consumption. This means, when changing network configuration, the scaling law of calculation TO for different processor types is the same. However, the TO-determined energy consumption varies with the choice of processor type.

4.6.2 Review of DNN Energy optimization: Data Movement Energy and Calculation Energy

The hardware-level energy consumption of a DNN model is composed by calculation and data movement energy consumption [105]. Source code of a DNN model could indirectly control the order and number of hardware operations, thereby determining the model energy consumption. Suppose the source code of a given learning model contains a set of calculation tasks $N = \{n_1, n_2, ..., n_k\}$. Task n_i calls data d_i from storage level l_i (determined by data storing strategy), and operation type is t_i. The energy consumption level of the current data movement strategy is represented by θ_m. Hardware calculation energy consumption level is represented by θ_c depending on hardware-related factors (e.g., circuit logic and materials). As function $f_{\text{move}}(d_i, l_i, \theta_m)$ and $f_{\text{calculate}}(d_i, t_i, \theta_c)$ calculates the energy consumption for data movement and calculation respectively, the energy consumption E of doing one prediction with the given DNN model could be summarised by

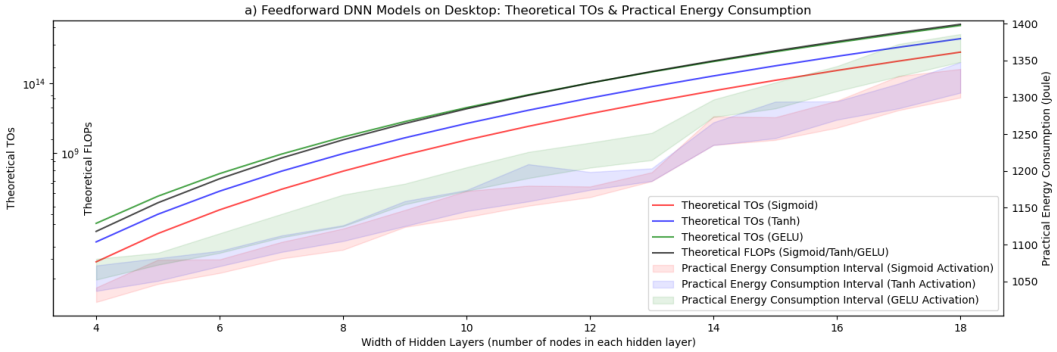
equation 4.9.

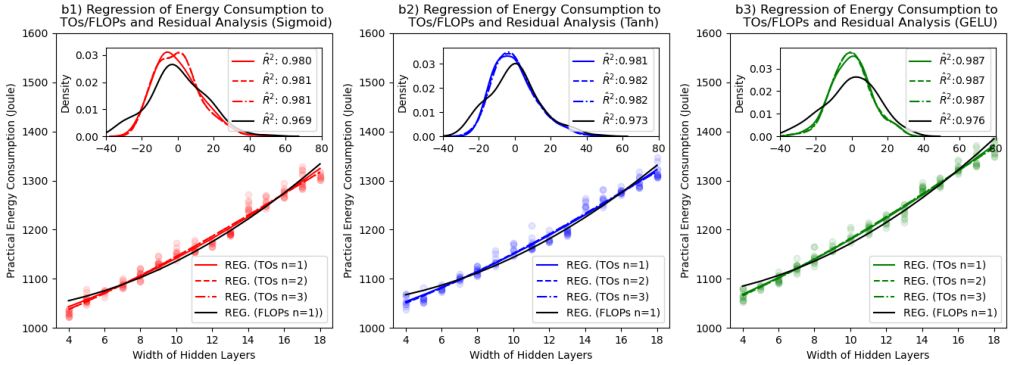
$$E = \sum_{i=1}^{k} (f_{\text{move}}(d_i, l_i, \theta_m) + f_{\text{calculate}}(d_i, t_i, \theta_c)) \tag{4.9}$$

$$\min_{m \in M} \text{trade-off} = \alpha E_m + (1 - \alpha) loss_m \tag{4.10}$$

The analysis of data movement energy and calculation energy are in different significance for developers. As data movement logic is pre-defined at the system level, it can only provide developers with limited assistance in model designs. However, it helps the learning frame developers to optimize the data strategy to improve data movement efficiency (optimize θ_m) as method proposed in [106]. By contrast, the analysis of calculation energy allows developers to intuitively understand the relationship between DNN model architecture and hardware operations. They can further optimize the learning model architecture (e.g., use energy-efficient AFs, reduce k [84]) to develop energy-efficient DNNs. At the same time, hardware engineers can design specialised hardware (e.g., TPU) for DNN to increase the energy efficiency (optimize θ_c) [137]. Federated learning [85] with TPU in edge can make profit from both high energy efficiency and high learning efficiency [138]. As shown in equation 4.10, developers could make a trade-off between DNN model energy consumption and performance by a selected α, and select the most appropriate model m from candidate model set M.

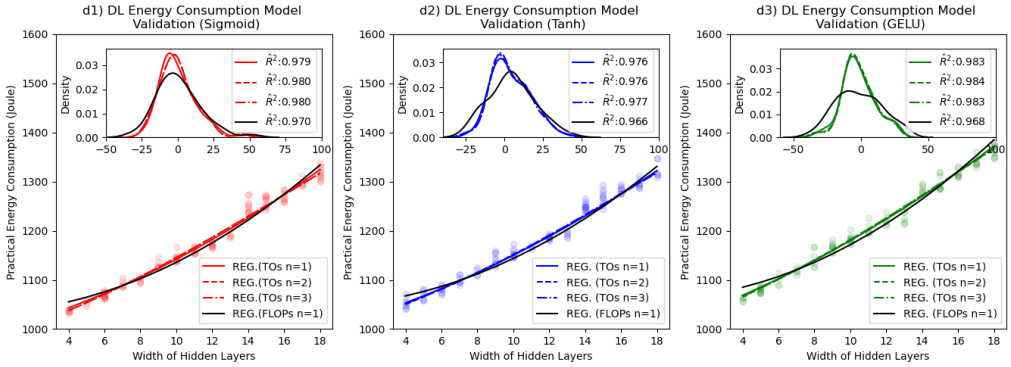
4.6.3 Extra Results of Method Verification





c) Training of Polynomial Regression (Energy to TOs/FLOPs): Coefficient and Performance Metrics

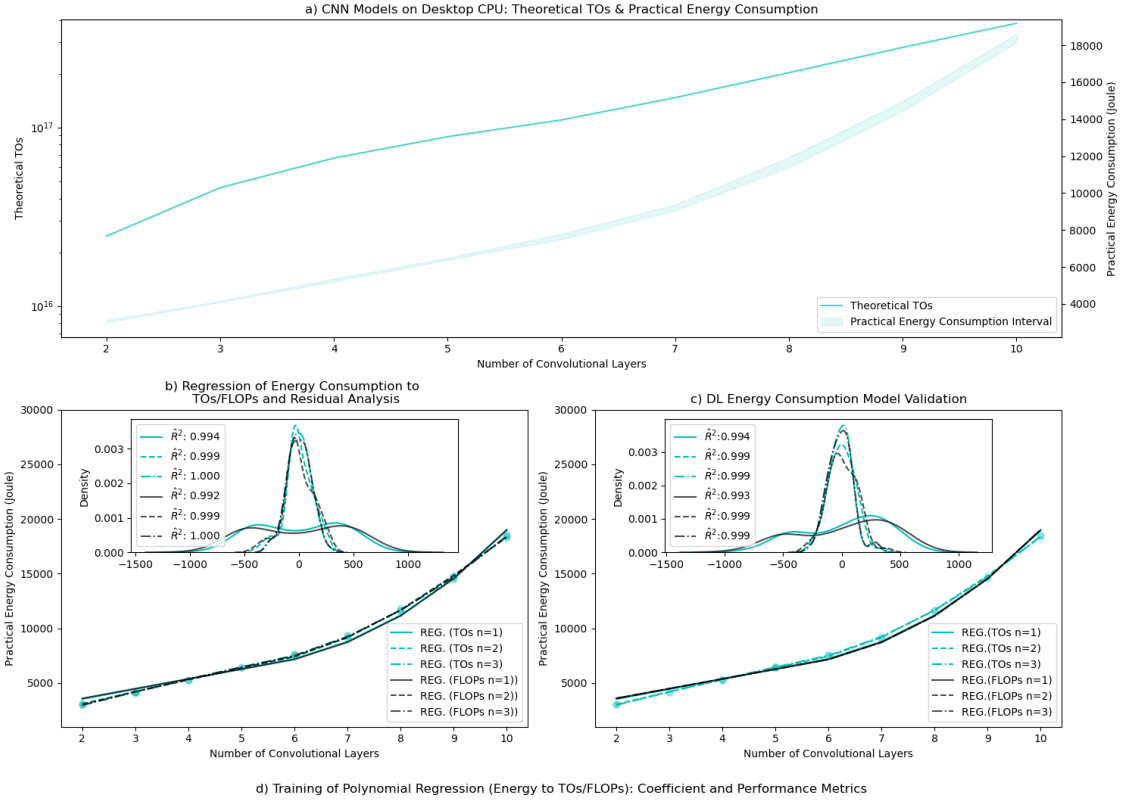
	n = 1	RMSE	$\hat{R}^2$	n = 2	RMSE	$\hat{R}^2$	n = 3	RMSE	$\hat{R}^2$
TOs (sigmoid)	9.96e+2, 2.47e-12	12.683	0.980	9.81e+2, 2.95e-12, -3.12e-27	12.158	0.981	9.92e+2, 2.35e-12, 5.77e-27, -3.84e-41	12.153	0.981
FLOPs (sigmoid)	1.04e+3, 5.71e-08	15.564	0.969	1.01e+3, 8.21e-08, -4.57e-18	12.461	0.980	1.01e+3, 8.37e-08, -5.26e-18, 8.20e-29	12.431	0.980
TOs (tanh)	1.00e+3, 2.15e-12	11.772	0.981	9.92e+2, 2.45e-12, -1.67e-27	11.636	0.982	9.97e+2, 2.23e-12, 1.14e-27, -1.08e-41	11.588	0.982
FLOPs (tanh)	1.05e+3, 5.59e-08	14.240	0.973	1.03e+3, 7.95e-08, -4.37e-18	11.696	0.981	1.02e+3, 8.76e-08, -7.97e-18, 4.39e-28	11.680	0.981
TOs (gelu)	1.01e+3, 2.12e-12	10.951	0.987	9.97e+2, 2.47e-12, -1.82e-27	10.764	0.987	1.01e+3, 2.03e-12, 3.25e-27, -1.70e-41	10.721	0.987
FLOPs (gelu)	1.06e+3, 6.17e-08	14.682	0.976	1.04e+3, 9.07e-08,-5.46e-18	10.950	0.987	1.03e+3, 9.98e-08, -9.55e-18, 4.96e-28	10.888	0.987



e) Validation of Polynomial Regression (Energy to TOs/FLOPs): Performance Metrics

	RMSE (n=1)	$\hat{R}^2$ (n=1)	RMSE (n=2)	$\hat{R}^2$ (n=2)	RMSE (n=3)	$\hat{R}^2$ (n=3)
TOs (sigmoid)	12.916	0.979	12.509	0.980	12.447	0.980
FLOPs (sigmoid)	15.418	0.970	12.358	0.981	12.423	0.980
TOs (tanh)	13.374	0.976	13.196	0.976	12.992	0.977
FLOPs (tanh)	15.721	0.966	12.950	0.977	13.020	0.977
TOs (gelu)	12.565	0.983	12.088	0.984	12.150	0.983
FLOPs (gelu)	16.947	0.968	12.560	0.983	12.350	0.983

Figure 4.9: Method Verification (Feedforward DNNs on Desktop CPU)



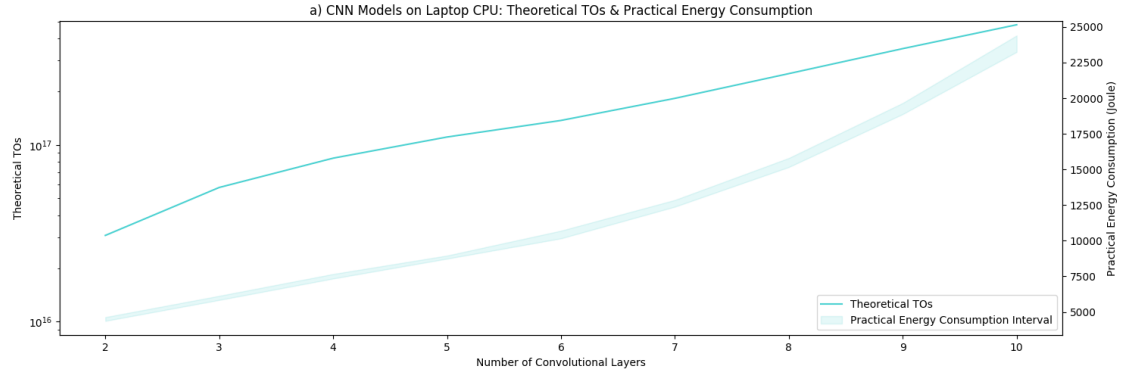
d) Training of Polynomial Regression (Energy to TOs/FLOPs): Coefficient and Performance Metrics

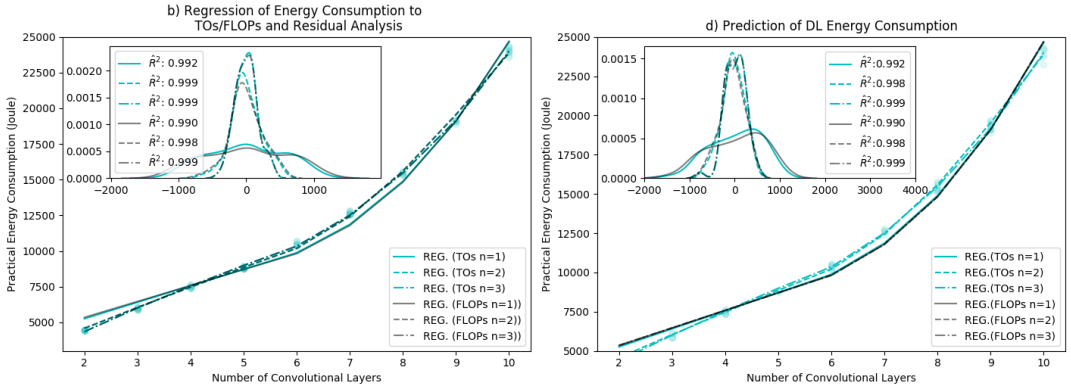
	n = 1	RMSE	$\hat{R}^2$	n = 2	RMSE	$\hat{R}^2$	n = 3	RMSE	$\hat{R}^2$
TOs	2.51e+3, 4.29e-14	387.184	0.994	1.71e+3, 5.55e-14, -3.14e-32	125.673	0.999	1.56e+3, 5.96e-14, -5.68e-32, 4.15e-50	106.433	1.000
FLOPs	2.67e+3, 5.73e-10	432.821	0.992	1.82e+3, 7.59e-10, -6.28e-24	138.951	0.999	1.61e+3, 8.37e-10, -1.28e-23, 1.45e-37	107.959	1.000

e) Validation of Polynomial Regression (Energy to TOs/FLOPs): Performance Metrics

	RMSE (n=1)	$\hat{R}^2$ (n=1)	RMSE (n=2)	$\hat{R}^2$ (n=2)	RMSE (n=3)	$\hat{R}^2$ (n=3)
TOs	350.802	0.994	112.994	0.999	101.406	0.999
FLOPs	394.248	0.993	124.393	0.999	103.563	0.999

Figure 4.10: Method Verification (CNN on Desktop CPU). The potential reason why statistics for FLOPs and TOs are close when n=3: Although several runs are made for each DL model structure, the statistics are still scarce, and higher-flexibility PR models with larger n could easily fit the statistics. Roughly, I think the workload quantified by any metric (FLOPs/MACs/TOs) should follow a linear relationship to the energy consumption, and the result of n=1 is more convincing.





c) Training of Polynomial Regression (Energy to TOs/FLOPs): Coefficient and Performance Metrics

	n = 1	RMSE	$\hat{R}^2$	n = 2	RMSE	$\hat{R}^2$	n = 3	RMSE	$\hat{R}^2$
TOs	3.88e+3, 4.35e-14	541.757	0.992	2.80e+3 5.78e-14, -2.88e-32	222.127	0.999	2.40e+3, 6.67e-14, -7.44e-32, 6.10e-50	160.202	0.999
FLOPs	4.08e+3, 5.80e-10	598.106	0.990	2.93e+3 7.90e-10, -5.71e-24	241.145	0.998	2.47e+3, 9.32e-10, -1.56e-23, 1.78e-37	161.292	0.999

e) Validation of Polynomial Regression (Energy to TOs/FLOPs): Performance Metrics

	RMSE (n=1)	$\hat{R}^2$ (n=1)	RMSE (n=2)	$\hat{R}^2$ (n=2)	RMSE (n=3)	$\hat{R}^2$ (n=3)
TOs	555.347	0.992	246.992	0.998	225.373	0.999
FLOPs	610.213	0.990	263.715	0.998	227.920	0.999

Figure 4.11: Method Verification (CNN on Laptop CPU)

Chapter 5

DL Learning Efficiency: Scarce Data Driven Deep Learning of Drones via Generalized Data Distribution Space

Increased drone proliferation in civilian and professional settings has created new threat vectors for airports and national infrastructures. The economic damage for a single major airport from drone incursions is estimated to be millions per day. Due to the lack of balanced representation in drone data, training accurate deep-learning drone detection algorithms under scarce data is an open challenge. Existing methods largely rely on collecting diverse and comprehensive experimental drone footage data, artificially induced data augmentation, transfer and meta-learning, as well as physics-informed learning. However, these methods cannot guarantee capturing diverse drone designs and fully understanding the deep feature space of drones. Here, this chapter shows how understanding the general distribution of the drone data via a Generative Adversarial Network (GAN) and explaining the under-learned data features using Topological Data Analysis (TDA) - can allow us to acquire under-represented data to achieve rapid and more accurate learning. This chapter demonstrates the results of the proposed method on a drone image dataset, which contains both real drone images as well as simulated images from computer-aided design.

When compared to random, tag informed and expert informed data collections (discriminator accuracy of 94.67%, 94.53% and 91.07% respectively after 200 epochs), the proposed GAN-TDA informed data collection method offers a significant 4% improvement (99.42% after 200 epochs). It is believed that this approach of exploiting general data distribution knowledge from neural networks can be applied to a wide range of scarce data open challenges.

This chapter provides a model to analyse the generalisation ability of convolutional layers on drone images, and uncover the relationship between CNN model performance and the training data. The result uncovers drone model(s) that are difficult for CNNs to learn, which means, CNN drone discriminators are highly possible to make wrong inferences on images of these drones. Accordingly, collecting additional efficient drone images for (re)training the CNN drone discriminator resulted in raising the DL model physical trust proposed in Chapter 2. The using of TDA also explains the hard-to-learn drone topological features, which contributes to the improving of emotional trust proposed in Chapter 2. Also, the efficient dataset will resulting in less training epochs, saving the energy cost according to later research in Chapter 5. Finally, the outcome of this chapter supports the discovery of efficient drone informatics, and directly supports the design of neuron stealth drone proposed in Chapter 6.

This chapter is published as a research paper by Neural Computing and Applications [12].

5.1 Introduction

Increased proliferation of drones and autonomous air vehicles can disrupt critical national services (*e.g.* Gatwick Airport 2018). The economic damage for air transport is estimated to be millions per day for airports and airlines [139, 140, 141]. The growth of drone industry generates high contributions to the economy (1.9% of UK GDP and supports over 600,000 jobs, 5 million consumer drone shipments worldwide in 2020 [142]), but

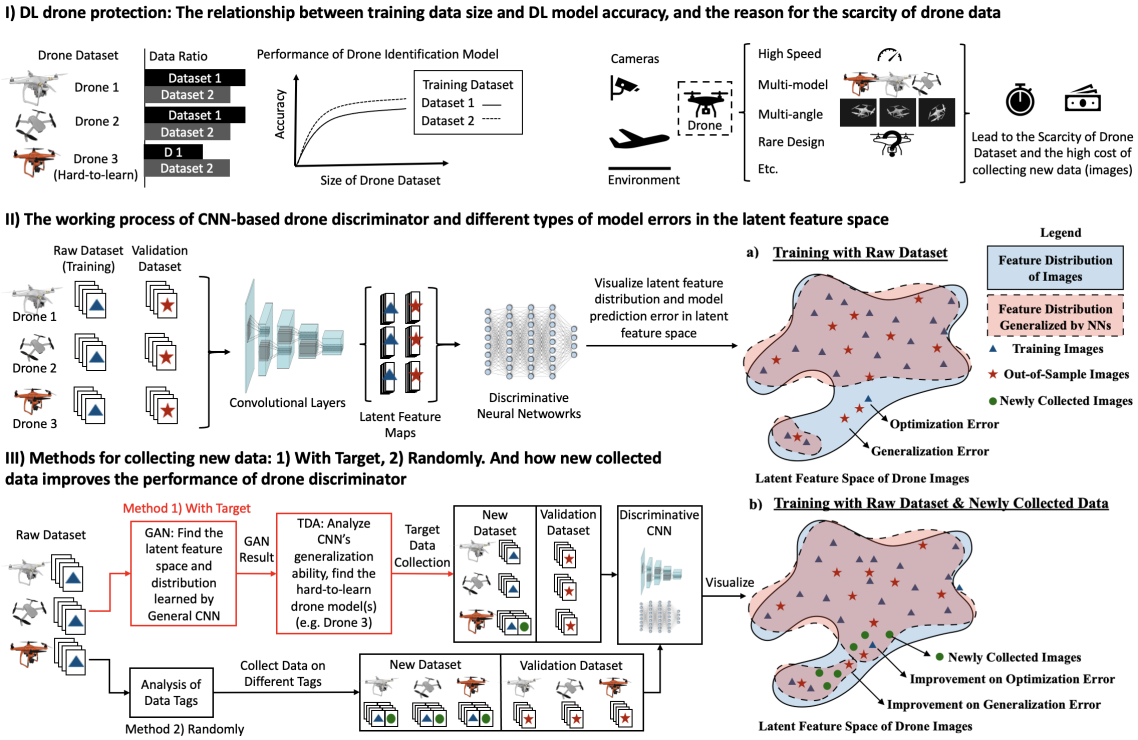


Figure 5.1: Reasons for the scarcity of drone data, errors in CNN-based drone discriminators, and methods for collecting new data. (I) The upper limit of DL drone classification accuracy is affected by the training data size, but various factors of drones lead to scarcity of drone data and high acquisition costs. (II) the nature of discriminative NNs' inference is to map high-dimensional input features into a label class using the feature distribution (FD) generalized from the training data (convolutional layers extract latent features maps of input images for further discriminative work in NN). While lack of training data will result in the generalization error on out-of-sample data. (III) The aim is to detect the data that important for model training (*e.g.* hard-to-learn drone), reduce the amount of new data required for model improvement, and outperform randomly data collection method in both learning speed and predication accuracy of trained model.

also brings new threat factors to air transport [7, 8]. Whilst many are amateur drones that pose no malicious intention, some may carry deadly capability and cause severe economic damage to critical infrastructure. Protection against drones is critical to ensuring smooth operation of services, whilst safeguarding it against the most severe threats. High resolution cameras can classify drones using deep learning (DL), but accurate identification is critical for not disrupting normal day-to-day operations and maintaining an efficient economy [143].

As shown in Fig. 5.1, the feature of drones led to a high data scarcity, and significantly challenged the accuracy and reliability of data-driven DL drone identification. As shown in Fig. 5.1.*I*, whilst the upper limit of accuracy for image classification has been increased by more complex deep learning architectures, the upper accuracy of DL model also limited by its logarithmic growth to the size of the training dataset [144]. This often means a large amount of resources and time is dedicated to broad data collection and (re)training DL-based system models to achieve higher drone discriminate accuracy. Simultaneously, high speed and small-size of drone challenge the image capture, while multi-model and numbers of shooting angle also increase the data scarcity of drone dataset. This scarce drone dataset lead to optimization error and generalization error in DL systems (see - Fig. 5.1.*II*). Recent methods for scarce data learning (*e.g.* data augmentation, meta-learning) artificially create new data based on existing ones, or transfer the knowledge learned from other domains. But, these methods can not solve generalization errors related to out-of-sample data. However, collect extra training data could address these errors as shown in Fig. 5.1.*III*. (Re)training the model on additional drone data can improve system accuracy, but is also more costly than other methods that do not require new data collection. However, a target of data collection can reduce the amount of new data to be collected, saving overall DL performance improvement costs. Accordingly, it is a key open challenge to achieve extremely high accuracy by sourcing sufficient, relevant but rare training data sets (*e.g.* rare drone design) [145].

This chapter shows how understanding the general distribution of the drone data via

a Generative Adversarial Network (GAN) and explaining the under-learned data features using Topological Data Analysis (TDA) - can allow us to acquire under-represented data (data instances with under-learned data features) to achieve rapid and more accurate learning. The aim is to demonstrate how to find the under-represented data for the DL model training by understanding DL learning behaviour, benefiting the efficiency of improving model performance by more targeted data collection.

5.1.1 Related work

One of the universal challenges in deep neural network training is when there is a lack of data, the out-of-sample performance can not be guaranteed [146]. As shown in Fig. 5.1.*II*, the nature of discriminative NNs' inference is to map an high-dimensional input features into a label class using the feature distribution (FD) generalized from the training data [147] (convolutional layers extract latent features maps of input images for further discriminative work in NN). While lack of training data will result in the generalization error on out-of-sample data [148, 149, 150], data scarcity can also lead to the optimization error on known training data (see - Fig. 5.1.*a*). However, collect extra training data could address these error as shown in Fig. 5.1.*b*. In general discriminative NNs, the performance increases logarithmically based on volume of training data size [144], which means the marginal cost of training data for model performance improvement boosts exponentially. Due to the diminishing returns, exhaustive or randomly searching for data is not applicable to the scenarios where data collection is expensive (*e.g.* aerospace, military). Therefore, there is a need to create a method to identify which specific new data would be important for discriminative NNs' performance improvement based on an existing dataset, as the guidance to the new data collection work, so that to reduce data collection cost. This is the motivation for this chapter.

As shown in Fig. 5.1.*III*, the aim of this chapter is to detect the data that is important for model training (*e.g.* hard-to-learn drone), reduce the amount of new data required for model improvement, and achieve a faster speed and higher accuracy training of DL-

based drone discriminator then other data collection methods. There are related papers addressing the aforementioned challenges in the discriminative NNs training with limited data (scarce data learning). Several research achievements are summarized and compared below in table 5.1.

Data augmentation improve the training set by adding slightly modified (*e.g.* translations, rotations and flips) copies of existing data to strengthen the invariance of NNs to the aforementioned modified data [151]. However, data augmentation is only based on raw training data, hence cannot offer additional generalization on the variation of the object itself other than its position to NNs. Transfer learning reduce the training cost of new DL model by reusing the convolutional kernels from other related well-trained DL models [152]. By doing so, NNs can partially transfer the generalization ability got in the former relevant training into the latter target inference. Similarly, meta-learning tries to abstract more universal generalization ability from multiple training domains and "learn to learn" fast [153], which lays the foundation for the few-shot learning [154]. However, the performance gain via these two methods are not guaranteed, since the transferred generalization ability is context-agnostic (*i.e.* does not focus on the properties of the new data), which may not match the need to specific requirements. Physical informed learning is designed to embed given laws of physics (*e.g.* general nonlinear partial differential equations) into NNs to inform its generalization [155, 156]. Hence, there is less need for the diversity of the training data and more training data can be generated numerically from the given laws of physics. However, in most discriminative NNs applications, physical law is unknown.

Although data augmentation enhances the training set by adding augmented data based on observed data (auxiliary variables method), there is no effect on un-observed data (*e.g.* Drone images by un-observed angle). However, newly collected training set could address this problem. Transfer learning and meta-learning can save the training time for new tasks, but can not enhance a trained model and are lack of explainability. Physical informed learning needs expert experience which is abstract and uncontrollable. Thus, this chapter proposes a method to reveal the relationship between NN's generaliza-

	Data Augmentation [151]	Transfer Learning [152]	Meta Learning [153]	Physical Informed Learning[155]	**Proposed GAN-TDA Method**
Methodology	Add modified data into dataset	Reuse Layers from other well-trained DL models	Reuse layers trained in previous tasks	Apply physical properties in model training	Purposefully collect new data
Generalization Ability	Modified dataset	Other related data	Other related data	Data physical properties	Newly collected data
Advantage	No additional data required	Wide applicability; fast deploy speed	Learn to learn; learn fast for new tasks	Controllable generalization	Explainability; less data to be collected;
Disadvantage	Limited improvement; lack of explainability	Context-agnostic, lack of explainability	Context-agnostic, lack of explainability	Difficulties in processing physical properties	Need additional cost for collecting targeted new data

Table 5.1: A comparison of methods for training deep learning models with scarce data

tion ability and the composition of training data, so that to provide a feature-based target for new data collection to save the cost of collecting new data. Although GAN-TDA method is with higher complexity in calculations (needs training of new DL models) and workloads (needs collecting new data) compared with the aforementioned methods, it is still necessary for several situations: when training dataset is not representative enough, new data collection is necessary to achieve higher DL performance; when collecting certain types of data is expensive (*e.g.* drone data); an explanation of DL errors related to data is needed.

5.1.2 Innovation: GAN-TDA framework

In conventional work, methods focus on the generalization ability of the model itself, which are general methods with versatility for various applications. By contrast, the generalization ability brought by the training data attracts less attention. In this chapter, the aim is to identify the required data for discriminative NNs' generalization error reduction, by analyzing the existing training data through its potential feature distribution (FD) and that generalized by NNs (see - Fig. 5.1.*b*).

Generative adversarial network Generative models are designed to generate data with the same FD as that learned by NNs. In principle, most common generative models, including variational auto-encoder (VAE) and variations of generative adversarial network (GAN), are trained to convert the initial distribution of latent variables into that learned by NNs with training data [157, 158, 159, 160, 161]. For scarce data problems such as in drone detection application, one might be interested in using GAN with the following reasons: (1) GAN is proven to be asymptotically consistent in FD approximation while VAE may have bias due to the variational lower bound, thus the generated data from GAN would be more precise in representing the FD learned by the NNs [162]; (2) With the same training data, the discriminator in GAN gives similar but more smooth convolution kernels as that in CNN [163]. More specifically, kernels in GAN have the same generalization way but weaker ability as CNN. Accordingly, the analysis in GAN can be considered as the representation of general CNN. (3) The discriminator in GAN can be considered as the pre-trained target discriminative NN for methodology validation so as to eliminate generalization ability bias caused by another arbitrary NNs [164].

In the experiment, color information in each pixel of drone images are use as inputs, the viewing of raw feature space built by pixel information will be over-dimensioned and lacks practical interpretability. Thus, to build a more informative feature space for image data, latent feature learned and processed by convolutional layers in deep learning models helps. Here, each latent feature is represented as the degree of response to a certain kernel feature in different receptive fields of the image. However, GAN does not give explicit expressions of the distribution, hence Monte Carlo synthetic data from the generator would be taken for the further FD analysis in the next step.

Topological data analysis For high-dimensional data analysis, dimension reduction approaches, such as PCA, MDS, t-SNE and *etc.*, are commonly applied [165, 166]. However, due to existing of generalization in NNs, the generated data may have more complicated topology than raw dataset in the high-dimensional feature space, while conventional

methods fail to capture any structure from the data, which cause catastrophic lose in high-dimensional distance information for the analysis after dimension reduction. In the framework, topological data analysis (TDA) mapper is proposed to address this issue. With the key idea of multidimensional persistence, TDA can capture data structure and then preserve the high-dimensional distance information with simplicial complex [167, 168, 169].

In the experiment, TDA is applied to tell the difference between the potential FD of data from raw dataset and that of synthetic data from the generator in GAN. With the high-dimensional clustering in TDA, discrete nodes are used to represent the original continuous feature distribution, while the connection between nodes indicates the distance in feature space. Before TDA, a step is needed that mix these two datasets into one with data labels attached (*i.e.* real, synthetic), so that to maintain the consistency of the topological space in TDA. Then, by analyzing the proportion of real/synthetic data in each node, one can discover the weak nodes, which lack the synthetic data, and then identify the required data by the real data tags in these nodes to guide the new data collection.

It is worth noting that, in the proposed methodology, the description for required training data cannot exceed human knowledge about the data. Although GAN-TDA approach works on the feature space in NNs, the result is still interpreted using the human feature space (tags), which may not match the need of neurons. What researchers can do here is to tag data with their best knowledge, so that to make the data collection guidance more targeted.

Contribution and novelty In this chapter, GAN-TDA is proposed to identify which specific new data would be important for discriminative NNs' performance improvement based on an existing drone dataset, as the guidance to the new data collection work. To my best knowledge, this chapter is the first to reveal the relationship between NN's generalization ability and the composition of training data, so that to improve the model performance via newly collected data.

There are three major contributions:

(i) GAN-TDA framework is proposed to guide the new data collection. Specifically, GAN to capture the feature distribution in inference generalized by discriminative NNs from the training data, and TDA to identify the generalization weakness on the training data.

(ii) A drone image dataset using both real drone images as well as simulated images in CAD is established for experiments. Each image is tagged with the drone's features (*e.g.* model, color, frame shape, camera position...).

(iii) A validation experiment of the proposed GAN-TDA method is demonstrated on a drone image dataset, which contains both real drone images as well as simulated images from computer-aided design. When compared to random, tag guided and expert guided data collections (discriminator accuracy of 94.67%, 94.53% and 91.07% respectively after 200 epochs), the proposed GAN-TDA informed data collection method offers a significant 4% improvement (99.42% after 200 epochs).

The remainder of this chapter is organised as follows. In Section II, the working flow of the proposed GAN-TDA framework is demonstrated. In Section III, the model is applied to a drone picture dataset for evaluation and validation. Section IV concludes this chapter and proposes ideas for future work.

5.2 Method

Given a dataset with clear properties labelled in tags that are detailed enough to guide the direction of new data collectivity (*e.g.* images in a vehicle dataset labelled with the vehicle's maker, model, type, color, number of wheel *etc.*), the methodology in this chapter is to identify which kind of data the discriminative model has weak generalization on by viewing the data distribution in the dataset and distribution learned by deep learning (DL) models.

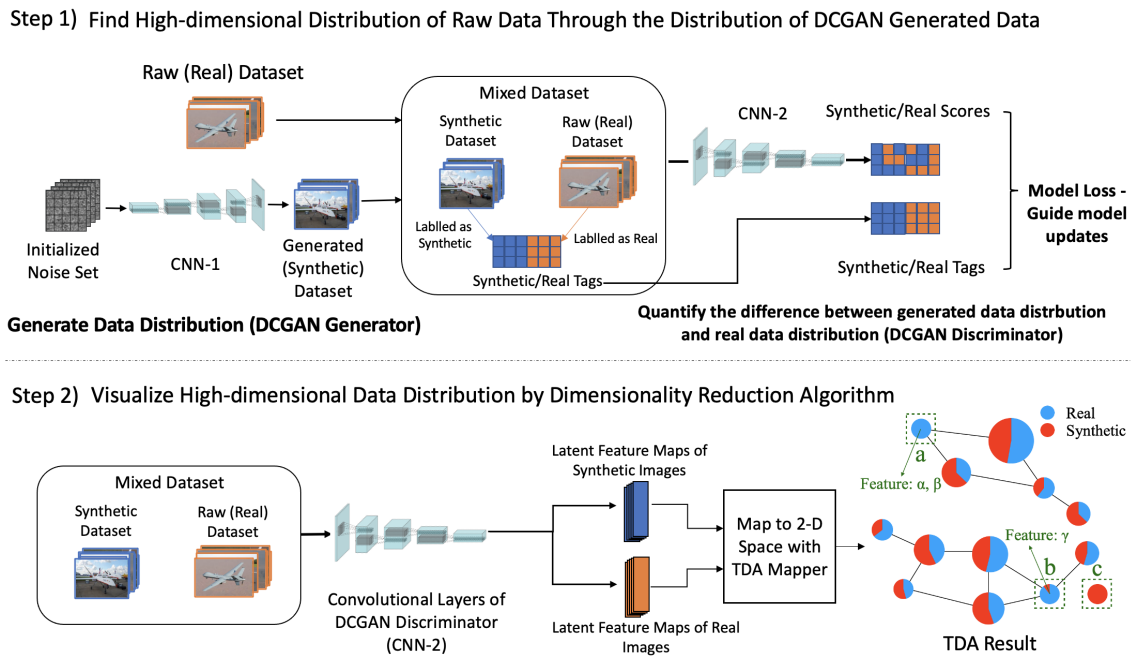


Figure 5.2: Demonstration of Method: Step 1) Find high-dimensional (latent features) distribution of raw dataset by DCGAN; Step 2) Use the convolutional layers of the DCGAN discriminator as the latent feature maps extractor of both the synthetic dataset and the raw dataset, and use TDA to visualize their differences in data distributions.

5.2.1 Step 1) - Learn data distribution by GAN

The first step is to find the high-dimensional distribution of the raw dataset (drone images). According to the demonstration of *Step* 1) in Fig. 5.2, two networks named Generator G and Discriminator D with different network structures will be established and initialized with individual aim to generate synthetic images and to recognise the input image is real or synthetic. Before the training of GAN, all the image from the raw dataset will firstly be pre-processed into the same size (*e.g.* 64*64 pixels) and secondly be normalized into the same scale. These steps are to ensure each input parameter (pixel intensities in each color channel) has a similar data distribution to guarantee the convergence of DL models[170].

Processed images from the raw dataset (real image dataset) will be divided into batches $B = \{B_1, B_2 ... B_i\}$ with a fixed size (*e.g.* 64 pictures per batch). During the training, a batch of initialized noise set n that follows a certain distribution (*e.g.* Gaussian Noise) will be generated whose batch size is the same as data batch size (*e.g.* 64*[100 samples from

Gaussian Noise]). The initialized noise set will be re-produced at the beginning of each training iteration and then be processed into a set of synthetic images $G(n)$ by fractionally-strided convolutions [171] in G. The optimization of G is expressed by minimizing the generative loss, which could be quantified by the discriminative result from D. And the optimization of D is to minimize the discriminative loss on both synthetic images and raw images.

During the quantification of model loss, raw images B_i will be labeled as $True$ and generated synthetic data $G(n)$ will be labeled as $False$, these information are stored into the real and synthetic tags matrix $T_{tag}(G(n),B_i)$ (mixed dataset). D will scoring the real and synthetic rate on both B_i and $G(n)$, the generated scores are stored in the real or synthetic score matrix as $T_{prediction}(G(n),B_i) = [D(G(n)),D(B_i)]$. The generative and discriminative loss could be further expressed by the divergence between $T_{tag}(G(n))$ and $T_{prediction}(G(n))$ and that between $T_{tag}(G(n),B_i)$ and $T_{prediction}(G(n),B_i)$. The divergence quantification uses Wasserstein distance to guarantee the stability of model training and avoid the collapse mode issue[160].

The processes of training a DCGAN with Wasserstein distance is shown as Algorithm 5. During the model training, optimizing model parameters by backpropagation in both G and D will let the distribution of generated synthetic data $G(n)$ gradually approaching that of the real dataset B. After the model converges, the distribution of the raw dataset is considered to have been learned and captured by GAN that the distributions of the GAN generated synthetic image set and the raw dataset is in a high similarity. Simultaneously, the ability to convert the given certain distribution noise set into real-enough synthetic data which follows the distribution of the raw dataset will be hiddenly stored with the form of model parameters in the GAN generator.

5.2.2 Step 2) - Latent feature maps extraction and TDA

In CNN, convolutional kernels are designed with weight sharing property, and each multidimensional kernel representing a unique hidden feature. So, in the convolution process,

Algorithm 3 Training of DCGAN with Wasserstein distance

Require: Training set $B = \{B_1, B_2 ... B_i\}$
Require: Generator G, Discriminator D, Gaussian Noise n, Wasserstein distance calculator W, Optimizer Opt
Require: Number of total epochs j
  Initialize G, D
  for epoch $<= j$ **do**
    for $B_i \in B$ **do**
      Initialize Gaussian Noise n
      $T_{tag}(G(n), B_i) =$
        $[False * G(n), True * B_i]$
      $T_{tag}(G(n)) = [False * G(n)]$
      $T_{prediction}(G(n), B_i) =$
        $[D(G(n)), D(B_i)]$
      $T_{prediction}(G(n)) = [D(G(n))]$
      $loss\_D =$
        $W(T_{tag}(G(n), B_i), T_{prediction}(G(n), B_i))$
      $loss\_G =$
        $W(T_{tag}(G(n)), T_{prediction}(G(n)))$
      $D \leftarrow Opt(loss\_D, D)$
      $G \leftarrow Opt(loss\_G, G)$
    end for
    epoch = epoch + 1
  end forreturn D, G

the output of each convolutional kernel means the degree of activation of an image property in a series of adjacent receptive fields controlled by stride.

The latent feature map of an image contains a set of activation degrees on different high-dimensional features and each value in the latent feature map can be seen as the activation intensity of different convolution kernels. It can also be regarded as the expression of high-dimensional image features to a low-dimensional space like the encoded latent features by generative models like VAE. The latent feature map can express the characteristics of the image to a certain extent, and be used to reduce the image dimension to facilitate the analysis of the image.

Certain data distribution will be expressed differently by latent feature spaces formed from different convolutional layers. To appropriately express the feature space for the raw dataset distribution, a proper multi-dimensional scale needs to be confirmed with a set of latent features learned by convolutional kernels. According to the fact that front convolutional layers tend to learn lower-dimensional visual features (*e.g.*line, Polyline, arc), later convolutional layers trying to extract high-dimensional latent features which contain complex information about positioning and relationship based on features extracted by the front layers. The feature space formed by kernels from the last convolutional layer is chosen to keep the deeply high-dimensional global information used for feedforward layers inferences. And this space could clearly express the distance between distributions of the synthetic dataset and raw dataset.

As shown in Algorithm 6, the latent feature map of a given image is defined as the output of GAN-discriminator's last convolutional layer after feeding the image into D (Step 2 of Fig. 5.2). Suppose the convolutional layers in D is defined as a layer set $C = \{c_1, c_2 ... c_k\}$ and the generated synthetic image from G as $S = \{G(n_1), G(n_2) ... G(n_i)\}$ (i is the batch number in B: to guarantee the generated synthetic images have the same amount of data as raw dataset). The tags of all synthetic image is set to $False$ as $T_{tags}(S) = [S * False]$, and $True$ for all images from real dataset B as $T_{tags}(B) = [B * True]$. As shown in Algorithm 6, the algorithm will return a dataset that contains both the latent feature

maps for synthetic images L_S and real images L_B with their tags $T_{tags}(S)$ and $T_{tags}(B)$ for further TDA use.

Algorithm 4 Extraction of latent feature maps

Require: Real image dataset $B = \{B_1, B_2 ... B_i\}$, Synthetic image dataset $S = \{G(n_1), G(n_2) ... G(n_i)\}$
Require: Convolutional layers of DCGAN's discriminator $C = \{c_1, c_2 ... c_k\}$
Require: Tags for synthetic images $T_{tags}(S)$, and for real images $T_{tags}(B)$
  Define $L_S, L_B = \{\}, \{\}$
  for $i <= len(B)$ **do**
    $l_B = B_i$
    $l_S = G(n_i)$
    for $j <= len(C)$ **do**
      $l_B = c_j(l_B)$
      $l_S = c_j(l_S)$
      $j = j + 1$
    end for
    $L_S.append(l_S)$
    $L_B.append(l_B)$
    $i = i + 1$
  end forreturn $\{L_S, L_B, T_{tags}(S), T_{tags}(B)\}$

The viewing of data distribution in the chosen feature space is barricaded by its high dimensionality, where TDA makes reasonable dimensionality reduction representation to help the discovery of distance between two high-dimensional data distributions. Kepler Mapper is selected for TDA visualization [169].

5.2.3 TDA interpretation

As shown in Fig. 5.2 Step 2), the TDA Mapper generates a network representation of the feature space, in which each node corresponds to a set of data with similar features. The size of the node indicates the quantity of the data assigned in it while the color is used to distinguish whether the data is real or synthetic in the node. In ideal conditions, the synthetic data would have roughly equivalent proportion in each node. Accordingly, the categories of nodes are listed as follows:

- (a) No synthetic data are generated in this node

- (b) both synthetic data and real data are placed in this node;
- (c) Synthetic data are generated anomalously outside of the real data feature distribution

While (a) indicates the GAN (representation of DNN) is failing to generalize the data with the feature α, β, (c) indicates that the GAN is still incomplete convergence, hence the GAN will be further trained while (a) or (c) nodes occur. Part of (b) nodes containing few generated synthetic data with feature γ, thus the data collection procedure will be processed on data with γ.

In perfect training, the percentage of synthetic data in each node should be close, which means discriminative NNs give roughly equivalent generalization to every models. In practical, there always has bias on the generalization to different models, which can be observed from uneven percentage in each node. Low percentage indicates the lack of generalization and vice versa.

5.2.4 Experimental setup

Recent deep learning drone detection models make inferences mainly based on images and video information captured by surrounding cameras. These captured visual data will be processed to complex high-dimensional latent features by convolutional layers based on image properties of different receptive fields for further inference steps completed by feed-forward layers. Therefore, various appearance and shape factors of drones designed according to different working environments and purposes challenge CNN based drone identification models in recognition precision, generalization and robustness a lot.

To investigate which design property of drone is hard for general CNN-based classifiers to learn and discriminate, an experiment is established to apply the GAN-TDA method into a collected drone image dataset (raw image dataset). This experiment aims to prove that the drone discriminator trained using additional images collected with the GAN-TDA method performs better than the model using new images collected with the

randomly method. During the experiment, the GAN-TDA model will analyze the raw dataset and generate the guidance on which kind of data should be collected additionally. Four models with the same net settings will be trained using four different newly collected datasets (see *Method 1)* and *Method 2)* in Fig. 5.1.*III*) in a control experiment to evaluate the feasibility of GAN-TDA guidance. The datasets for four groups are:

Group GAN-TDA - data from the raw dataset and additional data collected under guidance from GAN-TDA (additional data for several drone models).

Group Random - data from the raw dataset and additional data collected averagely for all data categories (additional data for all drone models).

Group Tag - data from the raw dataset and additional data collected under the guidance of labels for misclassified data (additional data for several drone models).

Group Expert - data from the raw dataset and additional data collected under guidance from experts (additional data for several drone models).

During the validation, four models trained by Group GAN-TDA, Group Random, Group Tag and Group Expert datasets respectively will be tested on the same validation dataset which contains images for all drone models distinct from the images in training sets. According to the performance comparison between the models trained with different Groups, the superiority of GAN-TDA guided data in improving general target CNN-based model generalization could be viewed. Details are listed as follows.

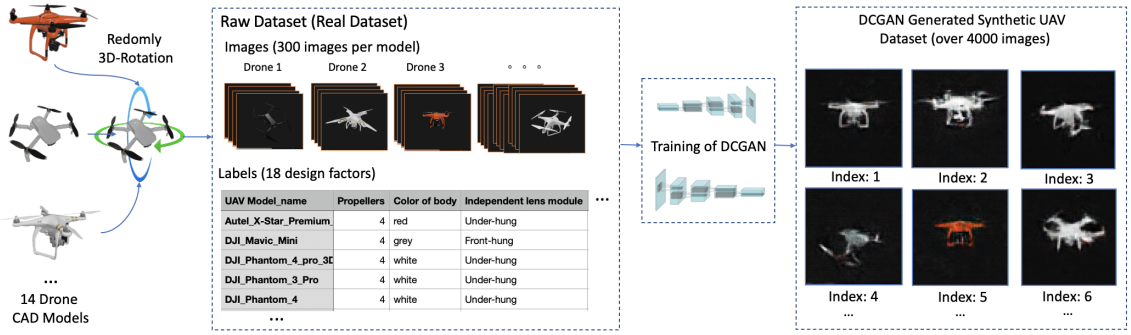


Figure 5.3: The Generating of the Raw Dataset and Synthetic Dataset. Raw dataset is collected from 14 CAD drone models, with 18 design factors and drone model name as labels. The synthetic dataset is generated by DCGAN trained on the raw dataset, with the same amount of data as the raw dataset.

5.2.5 The dataset and hardware

The raw dataset contains over 4000 pictures averagely collected from 14 popular commercial drones' 3D models[1] (*e.g.*, DJI Phantom 3, Phantom 4 and Phantom 4 pro). As shown in Fig. 5.3, to simulate the real pictures of flying drones caught by monitors and cameras with different angles, I randomly rotate these 3D models on x, y, z axis and take screenshots with drone centre-placed and 1800*1500 pixels resolution. During the data collection, the background of each drone model is set into black without ambient light effect to remove the high-frequency information from the background. These images are store in different folders named by their drone model's name, with an additional label file that contains a unique image ID for each collected drone image with 18 different hardware and appearance characteristics of these drones (*e.g.*, shape of propellers, number of propellers, position of floor stand). The synthetic dataset is generated by DCGAN trained on the raw dataset, with the same amount of data as the raw dataset. Each image in the synthetic dataset is labelled with an index, for identification (synthetic image) and trackback purpose.

The experiment environment is split into two part: the training of DCGAN is transferred into Cloud served by Google Cloud Platform with a VM (Ubuntu 16.04) established and 1 Tesla V100 GPU (NVIDIA-SMI 450.102.04, CUDA 11.0, 16G Memory) embedded; the evaluation experiment is processed by 8-Core Intel Core i9 (16G Memory).

5.2.6 DCGAN settings

To accelerate the training speed of GAN and avoid the missing of meaningful details in the appearance, each image in the real dataset is resized into a resolution of 3*64*64 (R, G, B channels, 64*64 pixels) and processed with pixel value normalization in each channel (mean = 0.5, standard deviation = 0.5). As shown in Table 5.2, the DCGAN-generator is designed with five fractionally-strided convolution layers to generate synthetic images

[1]The datasets generated during and/or analysed during the current study are available in the *figshare* repository, DOI: https://doi.org/10.6084/m9.figshare.21905094.v1

with the same resolution as the pre-processed real image (3*64*64). The discriminator is designed with five convolutional layers which accept 3*64*64 images as input, and the activation function of the last layer (Sigmoid in original DCGAN [171]) is removed to meet the requirements of use Wasserstein Loss [160]. During the training of DCGAN on the raw dataset, images from the dataset will be split into batches with 64 images each and shuffled before each training epoch. Gaussian noise is chosen to provide GAN-generator with original input, and in each batch training, a randomly initialized 64*100 noise set will be processed in GAN-generator into 64 synthetic images. According to the indications from [172], discriminator should be trained before generator and with more epochs than generator. The training rate is set to five that generator will be trained once after five times training of discriminator [160] with the same training rate $3e^{-4}$ using Adam Optimizer. The generating quality of generator will be checked every 400 epochs training, and part of the synthetic images are sampled and shown in Fig. 5.3 to demonstrate the generating quality of generator after 5000 epochs training.

5.2.7 TDA settings

As raw TDA metrics cannot be directly visualized, the Kepler mapper is developed to reveal the topological features of the space by constructing a graph [173]. The Kepler mapper is used in this chapter to aid visual exploration. TDA takes the latent feature map (4096 dimensions) of each drone image as input. Within the mapper, a customised 2-D length is established with two individual distances (Isolation Forest and $\hat{L2}$-Norm) [169]. The simplicial complex is created with the customised 2-D length as well as the latent feature map set, with number of intervals set to 15 and the overlap is 20%. K-means cluster is used in this chapter, but any clustering algorithm could be used as advised in [173]. The number of intervals influence the number of TDA nodes, while overlap influence the overlapping among TDA nodes.

Generator			Discriminator		
Layer	Type	Size	Layer	Type	Size
Input	Gussian Noise	100	Input	Image	3*64*64
ConvTranspose 1	4*4 Fractionally-strided Convolutions	512	Convolution 1	4*4 Convolutions	32
-	Batch Normalization	512	-	Leaky ReLU	32*32*32
-	ReLU	512*4*4	-	-	-
ConvTranspose 2	4*4 Fractionally-strided Convolutions	256	Convolution 2	4*4 Convolutions	64
-	Batch Normalization	256	-	Batch Normalization	64
-	ReLU	256*8*8	-	Leaky ReLU	64*16*16
ConvTranspose 3	4*4 Fractionally-strided Convolutions	128	Convolution 3	4*4 Convolutions	128
-	Batch Normalization	128	-	Batch Normalization	128
-	ReLU	128*16*16	-	Leaky ReLU	128*8*8
ConvTranspose 4	4*4 Fractionally-strided Convolutions	64	Convolution 4	4*4 Convolutions	256
-	Batch Normalization	64	-	Batch Normalization	256
-	ReLU	64*32*32	-	Leaky ReLU	256*4*4
ConvTranspose 5	4*4 Fractionally-strided Convolutions	3	Convolution 5	4*4 Convolutions	1
Output	Tanh	3*64*64	Output	None Activation Function/Sigmoid (validation)	1

Table 5.2: DCGAN network training/validation settings

5.2.8 Validation settings

The evaluation of the method is to show the performances of models trained with different additional datasets on distinguishing the synthetic and real drone data.

To control the experiment variable (avoid influences brought by different initialization ways), the network in DCGAN-discriminator is chosen to be the identical network initialization whose convolutional layers are pre-trained on the raw dataset. During the training of DCGAN, the increasing of discriminative ability in discriminator is suppressed by the gradually increasing adversarial power from DCGAN-generator. Once the generator is fixed, the training of the discriminator will no longer be limited and be seen as the training of a general discriminative DNN model.

Based on the result from GAN-TDA (details are listed in the Section: TDA Result), four datasets are collected for Group GAN-TDA, Group Random, Group Tag and Group Expert respectively. The additional data for Group GAN-TDA is evenly collected from the 3 detected hard-to-learn drone models (DJI Phantom 3 Pro, DJI Phantom 4 and DJI

Phantom 4 pro); Group Random additional data is evenly collected from all drone models; Group Tag additional data is evenly collected from 3 drone models (3DR Solo, DJI Phantom 3 Pro and DJI Inspire 2) which have the highest discrimination error rate on the DCGAN discriminator (13.3%, 7.3% and 6.3% respectively); Group Expert additional data is evenly collected from 3 drone models (Autel X-Star, DJI Inspire 1 and DJI Spark) which are with higher complexity in canopy structures other than others. In the practice environment, data accessibility for different models varies. Therefore, the evenly collected method is used in this experiment to represent the general random collection of new data. To control variables, 100 additional images per drone model are collected for Group GAN-TDA, Group Tag and Group Expert respectively. 21 additional images among all 14 drone models are collected for Group Random. With the use of discriminator as the initialization, the network structures of models trained by different Groups are the same as shown in Table 5.2, but Sigmoid activation function is added to the last feed-forward layer for the binary classification task.

The validation set is collected on all drone models evenly (25 images per drone model) and independent from the raw dataset and any additional dataset. Each image in the validation set will be pre-processed as the training sets before model inference. During training, model performance on the validation set will be supervised in each epoch.

5.3 Results

5.3.1 TDA result

The outcome by TDA shows a topological analysis result to the distribution of both synthetic data and real data. By analyzing the TDA diagram shown in Fig. 5.4, it is found that three drone models out of 14 are more difficult for deep learning models to learn. The TDA result analysis is with two steps:

Step 1 - Summarize weak TDA nodes: According to the *mapper summary*, none (a) or (c) type TDA node (clarified in Fig. 5.2) occurs that the trained GAN do capture the

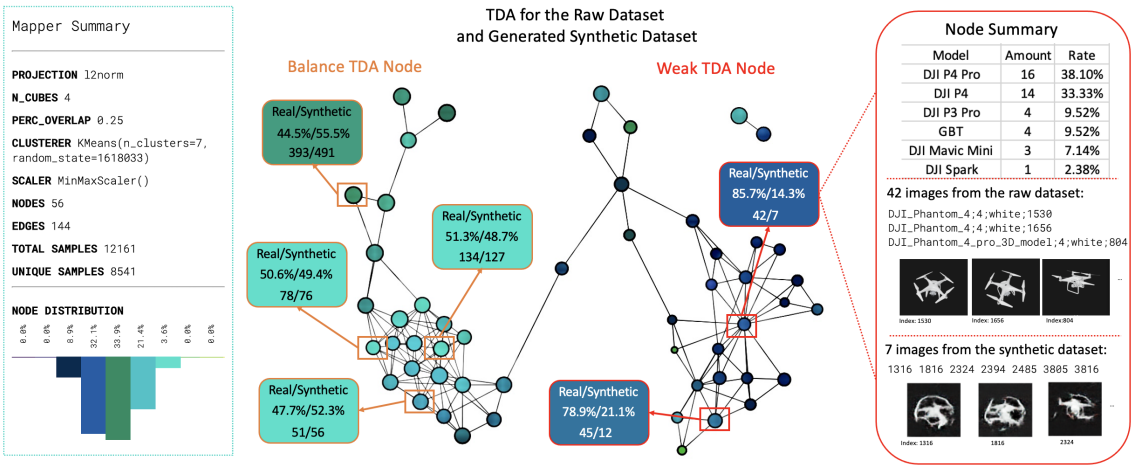


Figure 5.4: TDA output and analysis

data distribution of the raw dataset. There are two kinds of type (b) TDA nodes: Balance TDA node (the amount of real data and synthetic data is balance) and Weak TDA node (few synthetic data in this node compared with real data). The color of nodes reflects the internal balance of data that dark color refers to weak TDA nodes with low internal data balance. The rate of real data and synthetic data in some selected TDA nodes are listed in Fig. 5.4 for demonstration. According to the node distribution from the mapper summary, 8.9% TDA nodes are in dark blue color so the TDA analysis will focus more on these weak nodes.

Step 2 - Trace back to data labels: Tracing the tag of origin data whose latent feature map is placed in weak TDA nodes (which drone model are these data from) could provide clear guidance for new data collecting. The collection guidance in this chapter only focused on drone model names, while guidance by other information in tags (*e.g.* color and number of propellers) still remains for further research (due to these pieces of information in the current dataset being scarce). The details of an example weak TDA node are listed as shown in Fig. 5.4 - *Node Summary*. From here, the proportion of real data from different drone models will be summarized (*e.g.*16 out of 42 real data are from drone model DJI Phantom 4 Pro, and count 38.1% of real data in this node). By analyzing the data in all weak TDA nodes (with dark blue color), it is found that the majority of

real data placed in these nodes come from 3 drone models - DJI Phantom 3 Pro (22.7%), DJI Phantom 4 (27.6%) and DJI Phantom 4 Pro (33.5%). The result shows the DCGAN generalization ability on these drone models is weak. The TDA result further forms the GAN-TDA guidance that new data collection should focus on these models. During the experiment, it is found that the TDA is with low sensitivity to the parameters (intervals and overlap) under current experiment settings (during the changing of TDA parameters, DJI Phantom 4 and 4 Pro are always recognized as hard-to-learn, while sometimes DJI Mavic Pro replace DJI Phantom 3 Pro). However, the parameters should be adjusted for the TDA result clarity.

5.3.2 Discriminator result

The discriminator result shows that, on the designed validation dataset, the drone detection ability of the model trained with GAN-TDA guided additional data (Group GAN-TDA) is better than the model trained by random, tag and expert guided additional data (Group Random, Group Tag and Group Expert).

The demonstration of the discriminator result is made in two parts: *A) Validation Loss in BCE; B) Discriminative precision on the validation dataset*.

As the models' BCE losses on the validation dataset are shown in Fig. 5.5 *A)*, the performance of model trained by Group GAN-TDA dataset shows a significant advantage than that of models trained by Group Random, Group Tag and Group Expert dataset in BCE loss (Final loss: 0.0305, 0.1740, 0.2025 and 0.2755 respectively). This means the models' generalization ability on unseen new data could be affected by the quality of additional data for training. In other words, although different collection methods guarantee the learning of instances from the raw dataset, GAN-TDA guided additional data performs better in boosting the model's learning of data distribution, which leads to the reinforcing of model generalization ability.

The result in *B)* demonstrates the models' discriminative precision on validation dataset. The model trained with GAN-TDA guided additional data shows a quicker rising in

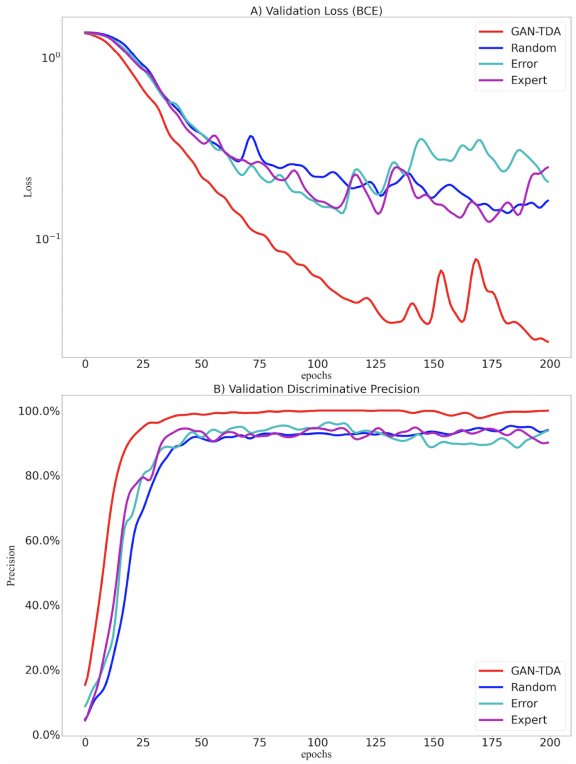


Figure 5.5: The performance comparison of targeted data collection method, random data collection method, tag-informed data collection method and expert-informed data collection method

inference precision of validation dataset, compared with additional data collected with other methods (Group Random, Group Tag and Group Expert). And by the end of training, GAN-TDA guided model's final precision on the validation dataset achieve a 4.75%/4.89%/8.35% increase on that of model trained by Group Random, Group Tag and Group Expert additional data (99.42% - 94.67%/94.53%/91.07% on 350 images from verification dataset). GAN-TDA guided additional data could make the deep learning model achieve high accuracy faster than other methods.

5.3.3 Influences on model QoT

According to the equations 5.1, 5.2, 5.3 (AI Trustworthiness, Physical Trust, Emotional Trust) proposed in Chapter II (2.1, 2.2, 2.3), the trustworthiness of AI is changed by retraining the discriminator on different dataset. However, there are two fine-grained level to analysis the system QoT, which are the discriminator itself, and the whole discriminator-data-method-system.

$$T(m) = \alpha P(m) + (1-\alpha)E(m) \tag{5.1}$$

$$\begin{aligned} P(m) &= R(m)\tau(m)\frac{A(m)}{C(m)} \\ &= R(m)\tau(m)\frac{a_m^d(\prod_{m'_n \in M} a_{m'_n})}{\sum_{m'_n \in M} \Omega_{g_n}(\omega(m'_n))/n}, \end{aligned} \tag{5.2}$$

$$E(m) = \frac{1}{q}\sum_{t \in T} l(t)\gamma(t) \tag{5.3}$$

QoT of discriminator

The majority of trust factors are not influenced. In detail, the task of the discriminator model m is still the discrimination of drone images, thus the ratio α of physical trust

$P(m)$ and emotional trust $E(m)$ in the overall AI trustworthiness $T(m)$ is not changed. The network structure of discriminator is without any changes, no additional surrogate models to increase the model's explainability, so the model complexity $C(m)$ and transparency $\tau(m)$ remains the same. As the amount of new data collected in different dataset are the same, and there are no additional augmentation method to increase the robustness of model, I assume the robustness remains the same (additional dataset could slightly increase the robustness, but ignore this influence in this chapter). Thus, the changes of model physical trust only depends on the changes in model accuracy.

Thus, a summary could be made that, the physical trust of discriminator that re-trained with GAN-TDA method is higher then that with other methods. Assume the complexity of model is the number of convolutional layers, $R(m)$ and $\tau(m)$ are 1 for demonstration purpose, and use the success rate of discriminator on drone image discrimination (at 200 epochs training) as the accuracy metric. The calculation of physical trust for discriminators trained with different method ($m_{\text{GAN-TDA}}$, m_{Random}, m_{Error}, m_{Expert}) could be seen in 5.4.

$$\begin{aligned} P(m_{\text{GAN-TDA}}) &= 1*1*99.42\%/5 = 0.1988 \\ P(m_{\text{Random}}) &= 1*1*94.67\%/5 = 0.1893 \\ P(m_{\text{Error}}) &= 1*1*94.53\%/5 = 0.1891 \\ P(m_{\text{Expert}}) &= 1*1*91.07\%/5 = 0.1821 \end{aligned} \tag{5.4}$$

At the same time, as the explainability of model is not changed, human willingness of trust this AI might be influences by the accuracy. Accordingly, the ranking of re-trained discriminator QoT will be $T(m_{\text{GAN-TDA}}) > T(m_{\text{Random}}) > T(m_{\text{Error}}) > T(m_{\text{Expert}})$.

QoT of discriminator-data-method-system

Similar to the QoT analysis of only discriminator, the ratio α, transparency $\tau(m)$, and robustness $R(m)$ remain the same. However, the GAN-TDA is used as a surrogate mod-

els to explain the AI behaviour, which will influence $C(m)$. There are two models in GAN-TDA method, which are GAN and TDA. The complexity of GAN considers both generator complexity and discriminator complexity. We consider the TDA complexity to be the number of TDA nodes divide by 10, as more nodes raise the complexity of TDA graph.

Thus, similar to the QoT analysis of only discriminator, $R(m)$ and $\tau(m)$ are set to 1 for demonstration purpose, and use the success rate of discriminator on drone image discrimination (at 200 epochs training) as the accuracy metric. The calculation of physical trust for discriminators trained with different method ($m_{\text{GAN-TDA}}, m_{\text{Random}}, m_{\text{Error}}, m_{\text{Expert}}$) could be seen in 5.5.

$$\begin{aligned} P(m_{\text{GAN-TDA}}) &= 1*1*99.42\%/(5+56/10+(5+5)/2) = 0.1912 \\ P(m_{\text{Random}}) &= 1*1*94.67\%/(5) = 0.1893 \\ P(m_{\text{Error}}) &= 1*1*94.53\%/(5) = 0.1891 \\ P(m_{\text{Expert}}) &= 1*1*91.07\%/(5) = 0.1821 \end{aligned} \tag{5.5}$$

We can see that, GAN-TDA slightly raises the complexity of the whole system, but still leads in system physical trust because of the high lead in system accuracy. At the same time, as different method provide different explanations, human willingness of trust this AI might be influenced accordingly. However, I predict that the end-users are more satisfied to the explanations provided by Expert-method; developers are more satisfied to explanations provided by Error-based-method; experts are more satisfied to explanations provided GAN-TDA; none of them satisfied to random methods because no explanations are given. Thus, the ranking of emotional trust $T(m_{\text{GAN-TDA}}), T(m_{\text{Error}})$ and $T(m_{\text{Expert}})$ will be influenced by the ratio of different types of users who are involved in the test. But, all emotional trust for these three method will be higher compared with $T(m_{\text{Random}})$.

5.4 Conclusion

High resolution cameras using deep learning is challenged by the lack of training data sets. This often means a large amount of resources and time is dedicated to broad data collection and (re)training the neural network - without a guaranteed convergence in improving accuracy. This chapter has used explainable deep learning to identify the under-represented data and guide data collection and generation.

For high-dimensional image data, the GAN-TDA provide a solution to extract the latent features of each data instance as feature maps and generate a demonstration of the generalization ability of the convolution kernels on different latent features. With the mapping relationship among images, latent features and labels, the generalization ability of kernels on latent features could indicate that on different image properties (according to data labels). During model training, the training of hard-to-learned kernels with slow improvement in generalization abilities needs more training epochs and additional date feedings, which means an image instance with properties of these hard-to-learned kernels are more difficult for DL models to learn (in this chapter: images with these properties are hard for GAN to generate). Afterwards, analysing these learning-hardly images with their tagged properties can indicate the direction of new data collection. However, the GAN-TDA method is not effective while tags are scarce - the result is not representative (*e.g.* GAN-TDA result: 2 out of 2 canopy colors are hard-to-learn). Meanwhile, the complexity of the GAN-TDA method is much higher than other data collection methods (training the GAN model used in this chapter cost over 100 hours; random data collection and expert informed data collection do not require new models; tag informed data collection need to train a DL discriminator which cost much less time than GAN training). But GAN-TDA is still worthwhile when collecting new data is expensive in both time and money, or an explanation of DL errors related to data is needed.

By applying GAN-TDA proposed in the chapter, it achieves a 4.75-8.35% precision boosting (99.42%) on drone discriminative NN compared with control models which use random, tag informed and expert informed collection methods (94.67%, 94.53% and

91.07%). Simultaneously, GAN-TDA guided data make the discriminative NN achieve the same inference performance with less training time.

Declarations

Funding: This work is supported by the Department of Transport under the S-TRIG program 2020-21; and the EPSRC/UKRI Trustworthy Autonomous Systems Node in Security [grant number EP/V026763/1].

Conflict of Interest

Chen Li, Schyler C.Sun, Zhuangkun Wei, Antonios Tsourdos and Weisi Guo are with Digital Aviation Research Technology Centre (DARTeC), Cranfield University, Bedford, United Kingdom. Weisi Guo is also with the Alan Turing Institute, London, United Kingdom.

Data Availability

The datasets generated during and/or analyzed during the current study are available in the *figshare* repository, DOI: https://doi.org/10.6084/m9.figshare.21905094.v1

Method Verification on Fashion-MNIST Dataset

To verify the effectiveness of the GAN-TDA method on other datasets, this chapter proposed a verification experiment that applies the GAN-TDA method on the open-source *Fashion-MNIST*[2] dataset. The dataset contains grayscale images (size: 28x28 pixels) of clothes and boots, with a label from 10 classes, 60k images for training and 10k images

[2]The Fashion-MNIST is an open-source dataset, available at: https://www.kaggle.com/datasets/zalando-research/fashionmnist

for verification. Here, the validation set is divided into two parts: the first part (0-5000 data instances) is used as a data source for collecting additional datasets, while the second part (5001-10000 data instances) still works as validation dataset.

The GAN-TDA result shows classes 1, 8, 3 and 5 are difficult to learn by convolutional layers (data rate: 35.8%, 32.3%, 9.5% and 7.4% respectively in weak TDA nodes). From the test result of the GAN-discriminator, the wrong discriminated data are mainly from classes 1, 6, 9 and 0 (data rate: 26.7%, 20%, 20% and 20% respectively). Based on the information above, four groups of additional datasets are design for comparison purpose, which are:

Group GAN-TDA - data from the raw dataset (training set) and additional data collected (from the validation set 0-5000) under classes 1, 8, 3 and 5.

Group Random-all-classes - data from the raw dataset (training set) and additional data collected (from the validation set 0-2000) under all classes.

Group Tag - data from the raw dataset (training set) and additional data collected (from the validation set 0-5000) under classes 1, 6, 9 and 0.

Group Random-selected-classes - data from the raw dataset (training set) and additional data collected (from the validation set 0-5000) under random 4 classes (in this chapter: 4, 5, 6 and 7).

The discriminator result is demonstrated in Fig. 5.6. From the figure, it is found that the discriminator re-trained with GAN-TDA guided additional data still shows a higher converge speed (see - BCE loss in sub-figure *A)*). At the same time, from sub-figure *b)*, the discriminator trained with GAN-TDA guided data reaches 100% accuracy slightly faster than that trained with additional data collected by other methods. However, the difference between GAN-TDA methods and other methods is not as large as that on the drone image dataset. The reason may be related to the ratio of the original training set to the additional data (the amount of additional data for model training is about 7% of the original training set in the drone experiment, while that of the Fashion-MNIST experiment is only 2%). The low ratio of additional data limits the difference in representativeness among data

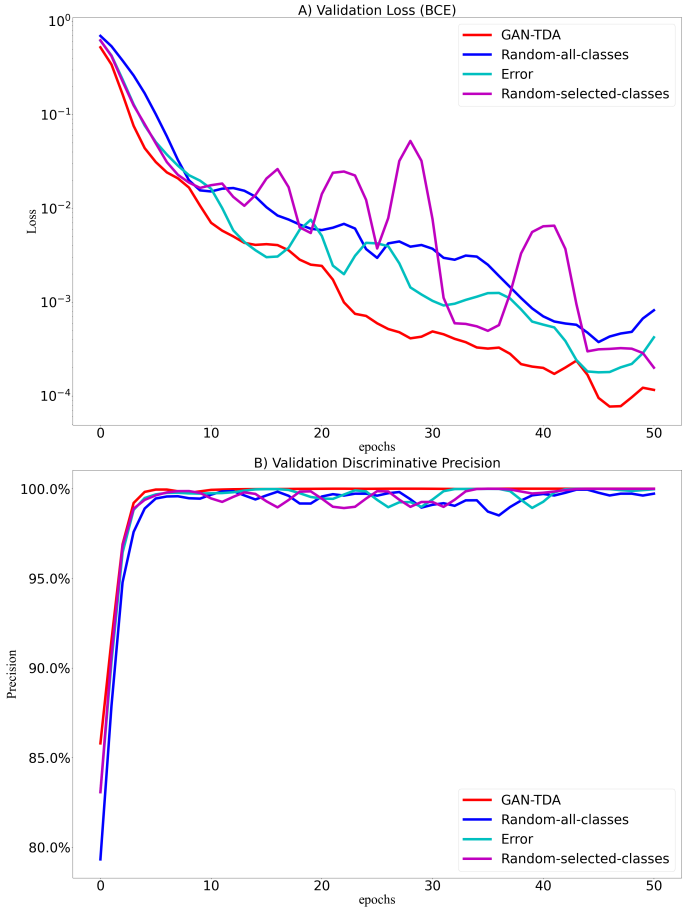


Figure 5.6: The performance comparison of targeted data collection method, random data collection method (on all/selected classes) and tag-informed data collection method.

collected with different methods.

Chapter 6

Designing AI Stealth Drones using Adversarial Topology and Imagery

Deep learning (DL) is widely used for automated surveillance of incursions into large sensitive areas (e.g., airports, prisons, docks, power plants). In counter drone systems, Convolution Neural Networks (CNNs) are widely used to pre-filter horizon scan (e.g., initial detection and threat classification). Recent research on adversarial attacks, especially evasion attacks are effective against general CNNs. Most work focus on planar perspectives (satellite images), but real mobile drones need robust perspective invariant neural stealth designs. Here, I use 2 novel approaches: (1) explainable AI to retrofit a neural stealth canopy that exhibit difficult to learn deep features, and (2) evasion patterns that are perspective invariant. Using simulated and real-world drone data, I show that the proposed topological and imagery stealth can have a combined reduction in drone detection accuracy from 85% to 46% and drone classification accuracy 100% to 0% against general and dedicated CNN algorithms.

This chapter uses the efficient drone informatics analysed in Chapter 5 to design a neural stealth drone, which significantly challenge the trustworthiness (Chapter 2) of CNN based drone classifier and objector. The drone canopy is then further printed with evasion patterns to reduce the trustworthiness of CNN in ground drone detect systems. This work

is also extended into a swarm drone scenario, achieves stealth of the target drone to the ground detectors (reduce trustworthiness) while maintaining visible communication with drones in the same swarm (none trustworthiness sacrifices).

This chapter is submitted to Nature Communications Engineering and is currently under major revision.

6.1 Introduction

Drone detection and classification is an important task in the prevention of low-slow-small (LSS) aerial threats in counter-unmanned aerial systems (C-UAS) [7, 8]. The visual features is predominant determined by the topological shape of the drone and the colour distribution. Convolutional neural networks (CNN) have proven to be accurate and robust to detecting drones even when they are distant, low-flying, and co-exist in clutter [9, 8]. However, as reported in [174] and [175], neural networks are sensitive to specific distortions and have generalisation errors. Current adversarial attack methods can mislead CNN inference by modifying the latent features by adding a colour perturbation. However, these methods are not suitable for drones currently because they do not consider the impact of 3D drone orientation during flight and the practicability of imagery noise.

As such, designing the canopy shape of drones with neural stealth properties is very important, but still lacks research. There are two main challenges in designing a neural stealth drone:

- identify which drone design latent features (or combination of features) that are difficult to learn for CNNs. Then, it is possible to explain and use these to retrofit a drone to exhibit these features.

- painting perspective invariant evasion attack patterns - ideally these patterns are robust against imagery noise, and have the potential to account for future soft-body or morphing capabilities.

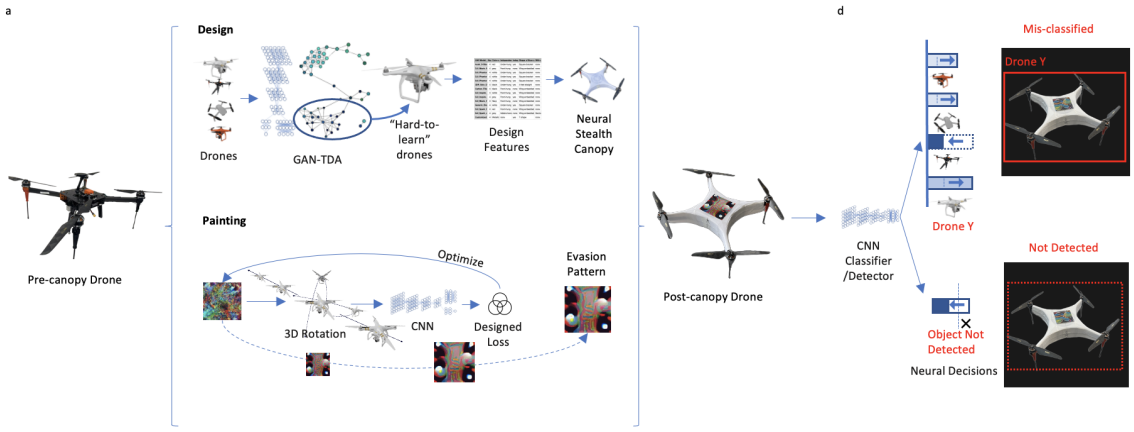


Figure 6.1: Designing AI Stealth Drone using Adversarial Topology and Imagery. **(a)** The design process is divided into canopy design and painting design. GAN-TDA is used to identify which latent features (or combination of features) make a drone difficult to learn for CNNs. And then, the next step is to train and paint perspective invariant evasion attack patterns on the neural stealth canopy. **(b)** The design of the canopy can manipulate the latent features extracted by the convolutional layers, thereby interfering with CNN inferences on object detection and drone classification.

6.1.1 Explaining Latent Feature Space for Drones

CNN-based drone classifiers and detectors use the learned data distribution in latent space for inference. Common amongst all CNNs is their convolution layers [176], which transform low-level semantic features (e.g., curvature of body, wingspan, no. of propellers) to latent feature spaces for full-connection layers task-specific processing.

GANs for Generating Representative Data

Generative Adversarial Networks (GANs) are powerful at generating representative latent space distributions for challenging data [177]. I analyse the generalization ability of convolutional layers on different visual-features of drones by training an unsupervised Deep Convolutional Generative Adversarial Networks (DCGAN) [171] model on drone image sets. The first reason is DCGAN can both learn the data distribution of drone images, and express the learned data distribution in feature space by generating enough synthetic data (drone images). The second reason is DCGAN have a similar generalization ability to convolutional layers in CNN-based supervised learning models (e.g. CNN drone de-

tector and classifier) using the same dataset, reported in [178]. The generalization ability of proposed DCGAN on a specific drone is reflected by the ability to generate synthetic images of this drone model. This means, under the premise of enough synthetic data, low generalization ability means DCGAN cannot or can hardly generate synthetic data for a certain data type (e.g., drone models). I comparatively visualize the data distribution of synthetic dataset and original dataset, as low generalization ability of DCGAN will lead to blank areas in the distribution of the synthetic data compared with the original data distribution.

TDAs for Explaining & Visualising Latent Feature Maps

I use topological data analysis (TDA) [179] on latent feature maps of both synthetic and original data to produce their dimension-reduction expressions, cluster data with similar topological features, and visualise them in a uniform TDA space. Here, each TDA node contains images from original and synthetic datasets with similar topological features, which means they have a similar position in data distribution [180]. The connections between TDA nodes tell their distance. I use the drone labels for images from the original drone dataset in each TDA node to describe data features in this node, as synthetic data are not with labels. I call a TDA node 'weak' when it contains none or not sufficient synthetic drone images. A weak TDA node represents the DCGAN model has low generalisation ability on this kind of data. "Hard-to-learn" drones (DCGAN has low generalization ability) can be found by summarizing images from which drones play the main role in each weak TDA node.

I leverage on aerospace design experts to judge which difficult to learn labels have common aircraft design features (e.g., wingspan, propeller configuration,...etc.). It was found that (see details in Method) drones with curved body that hide sharp edges were more difficult to learn. As such, I design a curved canopy to retrofit onto existing drones and test whether this reduces the detection and classification accuracy.

6.1.2 Perspective Invariant Evasion Pattern

Adversarial evasion attack methods are well researched to mislead CNN inference by modifying the latent features of objects in images with a human visible or invisible colour perturbation noise [181, 182, 183]. However, an invisible attack directly adds digital perturbation to images after capturing processes, which needs access to the attacked system data portal and is not applicable to the design of drone canopies. Adversarial patch methods [184], attack CNNs by applying a visible evasion pattern with evasion patterns on an image or object, for different purposes (e.g., undermine CNN performance in drone classification, detection or both). The challenge with computer generated evasion pattern is that real world lighting and imagery noise, as well as 3D rotation/scaling should be taken into account.

I review specific recent developments in evasion patterns. Authors in [185] propose a basic patch method to fool a YOLO v2 detector, and provide a demo that a person escapes from CNN detection by holding a physical patch. Authors in [186] apply this method to military plane images captured by unmanned aerial surveillance to let planes in image escape from a YOLO v2 object detector, but the performance is analysed based on the virtual experiments. There are several novel variants of patch training such as using GAN [187] to generate multiple patches at the same time; generating butterfly-shape patches with GAN [188]; data independent method to generate patches [189]; train face-mask-shape patches [190]; train deformable patches [191]. However, these methods ignore the perspective of patches caused by the rotation of the objects, and resolution changes (clarity of optical features) for objects and patches in images caused by different camera distances.

The authors in [192] train evasion patterns with simulation on camera angles and distances to demonstrate how sensitive evasion pattern attacks are to the above factors. However, statistics are collected from virtual validation of super-resolution evasion patterns. High-resolution evasion patterns can easily reach a higher performance during virtual training compared with lower ones, as they have the ability to store more neural

stealth features. However, these meticulous features are easily corrupted by distortions caused by ambient lighting and camera quality during image capture, leading to failure in practical use. As drones are with high flexibility in flying behaviour, a practical evasion pattern should have invariance and robustness to the rotation of drones (perspective of evasion pattern), optical sensor distance, and distortion caused by environment lighting and camera quality.

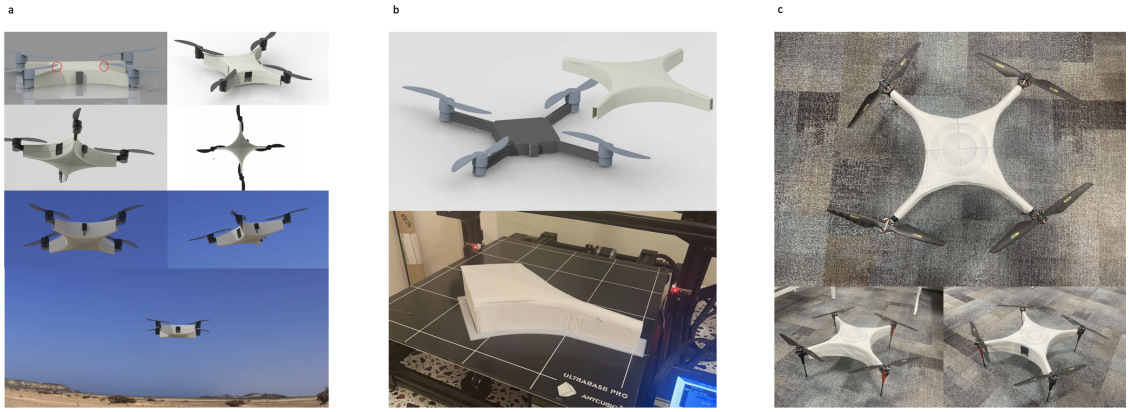


Figure 6.2: Design of Neural Stealth Drone Canopy and 3D Printing Process.

To solve the aforementioned concerns, I improve the method from [185] by adding a few additional simple augmentation tricks to the evasion pattern during training. The original method augments the evasion pattern with random 2D rotation, location, noise, and adjustments in image quality (e.g., brightness, contrast). To simulate different shooting angles, I add random perspective effects on evasion patterns before virtually sticking them on drone images. I then manually add blur to the images (both object and evasion pattern) by randomly downsizing and then resizing them to their original size. This is to simulate the simultaneous changes in the resolution of objects and evasion patterns in the images to different camera distances. I also added a random scale to the evasion pattern to simulate different size ratios between the drone and the physical evasion pattern. The idea is to reduce the sensitivity of the evasion pattern to its relative size in the image. Random elastic transformation on the evasion patterns during training is optional, which allows the evasion patterns to adapt to soft-body drones. Indeed, these additional augmentations will

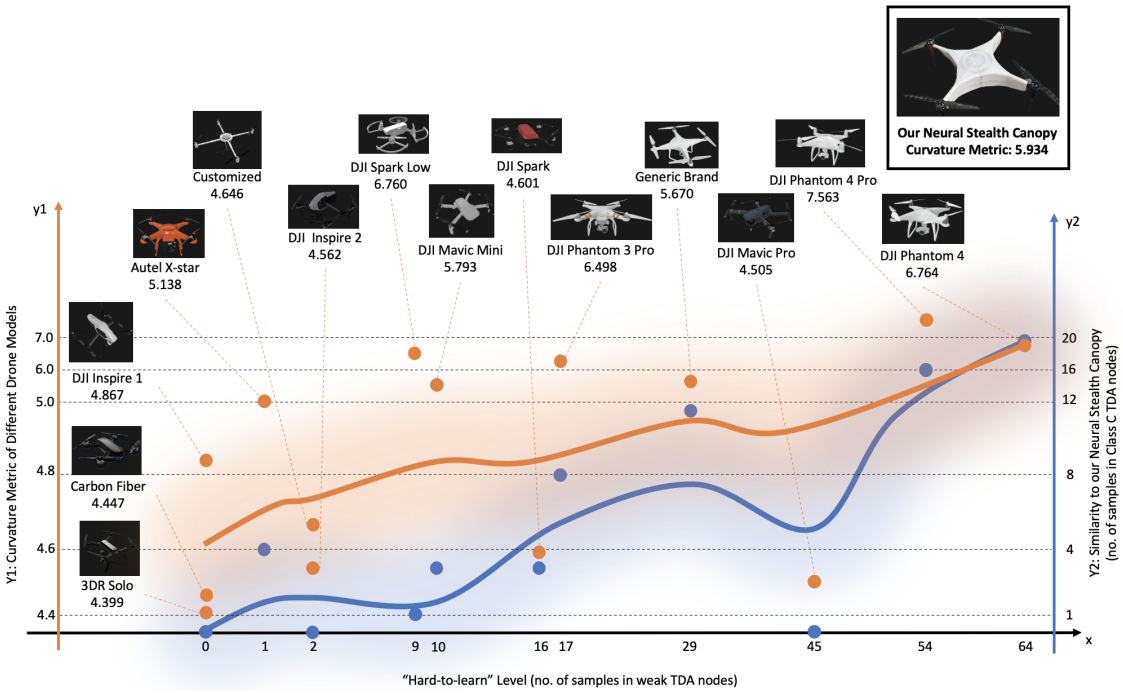


Figure 6.3: The "Hard-to-learn" Drone Analysis by GAN-TDA, and the Verification of Proposed Canopy Design. The x-axis is the "hard-to-learn" level of the 14 drone models, represented by the number of samples of each drone model in weak TDA nodes. The y1-axis is the curvature metric for each drone model, indicating the richness of curves in their canopy design. The y2-axis is the similarity of each drone model to the canopy design in TDA space, represented by the number of samples of each drone model in (Class C) TDA nodes. From the result, I have: 1) the curvature metric of drone models is positively correlated to their "hard-to-learn" level; 2) in the TDA space of image latent feature maps, proposed canopy design is close to the top "hard-to-learn" drone models.

significantly affect the upper bound of the evasion pattern performance, as more evasion pattern capacity is occupied by augmentation information while keeping the total capacity constant. However, I think they are worthwhile and necessary, that a evasion pattern performs well in the virtual environment and fails in practical ones is unreasonable.

6.2 Results

In the results section, I wish to convey the step by step results from generating an effective neural stealth canopy to the perspective-invariant evasion patterns, to the final validation testing.

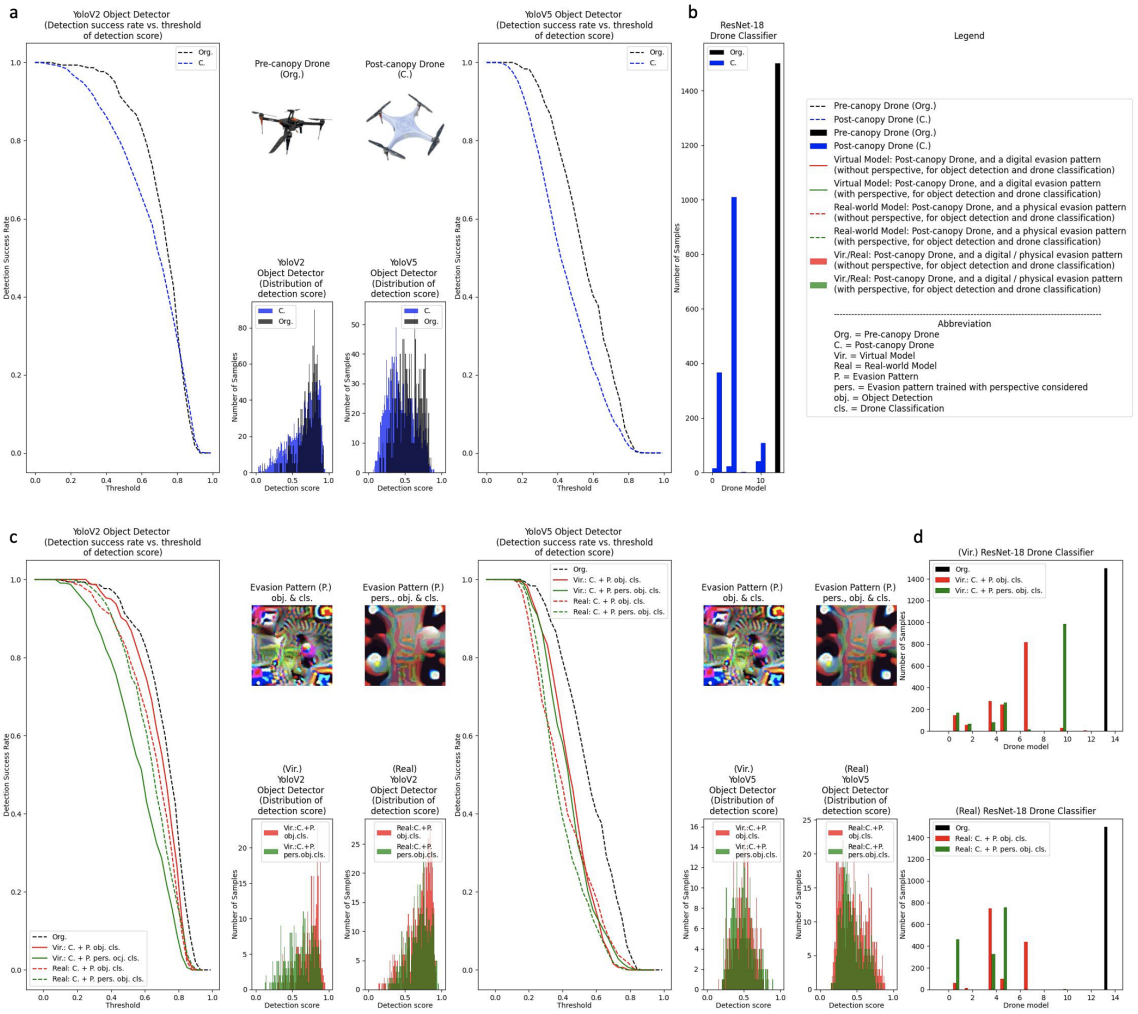


Figure 6.4: Canopy Design Validation: **(a)** The validation of the canopy design only (without evasion attack patterns) on interfering CNN object detection. Canopy performance is verified by the detection success rate of Yolo V2 and V5 detectors under different confidence thresholds (hyper-parameter for all Yolo family models, the detection box with higher detection score over the threshold will be seen as containing an object). The distribution of the highest detection scores in Yolo V2/V5 detection boxes for each image is also provided. **(b)** The performance validation of the proposed canopy design on interfering CNN drone classification is based on a ResNet-18 dedicated drone classifier. As all images for the pre-canopy drone are correctly classified into drone model 13, while only 1 out of 1571 images (0.06%) of post-canopy drone is classified into drone model 13. Most are wrongly classified into drone models 1 and 4 (30.3% and 64.2%). As the output of the ResNet drone classifier is decided by a softmax of class scores of all possible drone models, the distribution of model output is not demonstrated because it is not as meaningful as that of the Yolo output. **(c, d)** The validation results of the evasion attack patterns on interfering CNN object detection and drone classification. Two evasion patterns are trained comparison (without and with perspective during training, for interfering with both CNN detection and classification). Both of them are stuck digitally or physically on the canopy to generate a virtual attack model or a real-world attack model. **(c)** the evasion pattern trained with perspective works better than the evasion pattern trained without perspective in the virtual and real-world attack of Yolo V2. And, both evasion patterns showed transfer ability to attack Yolo V5, while the evasion pattern trained with perspective is better in both virtual and real-world attacks. **(d)** Shows the virtual and real-world attack performance of both evasion patterns.

6.2.1 Neural Stealth Canopy Results

As explained earlier in this chapter (details in Methods), I use a GAN to learn the general drone feature distribution in the convolution layers of any CNN. I then leverage on TDA to expose how the ”hard-to-learn” connected features relate back to aircraft design. In Fig. 6.2 I show the step by step process of a) designing the canopy (curvature informed design - details in Methods), b) rendering and 3D printing, and c) fitting onto an existing drone.

Here, in Fig. 6.3, I show how ”hard-to-learn” is quantified by the number of drone data samples in weak TDA nodes corresponds to both:

1. aircraft design - no. of curvature features in the drone air frame (see curvature kernel convolution in Methods), and

2. validation - similarity in TDA space between neural stealth canopy and ”hard-to-learn” drone designs

As one can see in Fig. 6.3, there is a strong relationship between the number of curvature features and the difficulty in recognising a drone using any CNN convolution layer. I then design the stealth canopy to increase the number of curvature features by shielding sharp edges and this corresponds well with the ”hard-to-learn” TDA space.

In Fig. 6.4a-b I show the performance of the canopy in a real drone setting. In subplot a), I show how different general Yolo model object detection is degraded by the stealth canopy (black is pre-canopy, blue is post-canopy). In subplot b), I show how a dedicated ResNet-18 model drone classification is degraded by the stealth canopy (black is pre-canopy, blue is post-canopy). In both cases, I vary the threshold of the representation layer, I can see that the maximum difference can be up to 35% loss.

6.2.2 Evasion Pattern and Canopy Combined Results

In Fig. 6.4c-d I show the performance of the impact of evasion pattern on top of the aforementioned canopy. In subplot c), I show how different general Yolo model object detec-

Table 6.1: Performance Evaluation of AI Stealth Drones on Yolo V2 and Yolo V5 Generic Object Detectors (threshold: 0.6 and 0.3 respectively) and ResNet-18 Dedicated Drone Classifier.

	Virtual Models			**Real-World Models**		
Model	Detection (Yolo V2)	Classification (ResNet-18)	Overall	Detection (Yolo V2)	Classification (ResNet-18)	Overall
Pre-Canopy (Original) Drone	84.6%	100%	84.6%	84.6%	100%	84.6%
Post-Canopy	62.5%	0.06%	0.04%	62.5%	0.06%	0.04%
Post-Canopy; +Evasion Pattern obj. & cls. (pers.)	74.7% (45.6%)	0%	0%	68.2 (58.7)%	0%	0%
Model	Detection (Yolo V5)	Classification (ResNet-18)	Overall	Detection (Yolo V5)	Classification (ResNet-18)	Overall
Pre-Canopy (Original) Drone	91.7%	100%	91.7%	91.7%	100%	91.7%
Post-Canopy	76.6%	0.06%	0.5%	76.6%	0.06%	0.5%
Post-Canopy; + Evasion Pattern obj. & cls. (pers.)	73.2% (66.5%)	0%	0%	55.3% (52.7%)	0%	0%

tion is degraded by the stealth canopy and evasion pattern (black is pre-canopy/pattern, red is post-canopy/pattern, green is post-canopy/pattern with perspective invariance). In subplot d), I show how a dedicated ResNet-18 model drone classification is degraded by the stealth canopy and evasion pattern (black is pre-canopy/pattern, red is post-canopy/pattern, green is post-canopy/pattern with perspective invariance). In both cases, I vary the threshold of the representation layer, I can see that the maximum difference can be up to 45% loss. I can see that the perspective invariant evasion patterns significantly degrade performance of detector and classifier.

I summarize key findings across all combinations of canopy and evasion patterns in

Table 6.1. The highlight results are:

1. Neural stealth canopy informed by ”hard-to-learn” features is effective at degrading detection accuracy (20%) and as one would expect, highly effective in evading classification due to the new design (99%). This is true for any CNN based detector/classifier as the GAN is designed for convolution layers.

2. Evasion pattern needs to be designed with perspective invariance (pers.), as I can see there is up to 25% difference in performance degradation.

6.3 Discussions

In this chapter, my primary goal was to develop practical neural stealth techniques to ensure that drones can evade both the detection and classification of general convolution layer neural networks - that make up most of current image processing algorithms. In this process, the main challenges I resolved are related to: (1) uncovering deep features that are difficult to detect and relating them back to interpretable aircraft design concepts, and (2) perspective invariant evasion patterns due to the 3D mobile nature of drones. In this process, I discovered that many challenges remain open due to realistic operating conditions.

First, partially obscured drones may cause the canopy or evasion pattern to be less effective. Naturally, I can add different degrees of partial visibility in the data augmentation stage after GAN training process. Second, many drones now have either soft body or morphing body capabilities, which means any evasion patterns need to be elastic. I discussed recent advances in this area in the introduction and provide my own prototype work in Supplementary Information (SI). Thirdly, as shown in SI, high resolution evasion pattern can overfit to achieve a good performance in a particular perspective but is not robust to errors in printing, lighting variations, and perspective changes. Fourthly, the image capture quality (e.g. aperture) of ground drone detection systems and the DL model used also matters. Similar to the effect of environment lighting on the evasion pattern,

the aperture could influence the features in the captured image, thus the effectiveness of evasion patterns. This needs appropriate augmentation methods to enhance the robustness of evasion patterns. Also, methods to generate high transferability evasion patterns for generic CNN backbones should be discussed. Finally, I acknowledge that there are a number of ways to combat the proposed design from a detection/classification, such as using an information entropy detector that is sensitive to neural evasion pattern designs, or detecting evasion patterns through various forms of adversarial training.

I leave these topics to future research. For us, I want to use GAN on evasion pattern augmentations to simulate the complex effect of aforementioned practical environmental factors [193]. I believe this will grant the trained evasion pattern with higher ability to transfer from virtual to practical use, as I found the influence of environment light on the physical evasion pattern is hard to be approximated by conventional augmentation methods (e.g., hue, saturation and brightness). And, to against multi-sensor drone detection system, considerations are also needed to eliminate the sound sensors (e.g. high altitude route and muting ability). 3D evasion patterns are also a good research direction which could have higher capacity to fool CNNs.

6.4 Extend the Work to Swarm Drone Scenarios

As shown in Fig. 6.5 a. Suppose there is a drone swarm flying with a stable trajectory (perpendicular to the ground, or nearly), and a hostile drone detection system whose position is known. I want to achieve the drones are stealth for the detection system, but not for other drones in the swarm. Because the view angles of cameras from the drone detection system and from other drones are different, the evasion patterns could be trained to target specific view angles to achieve flexible neural stealth features.

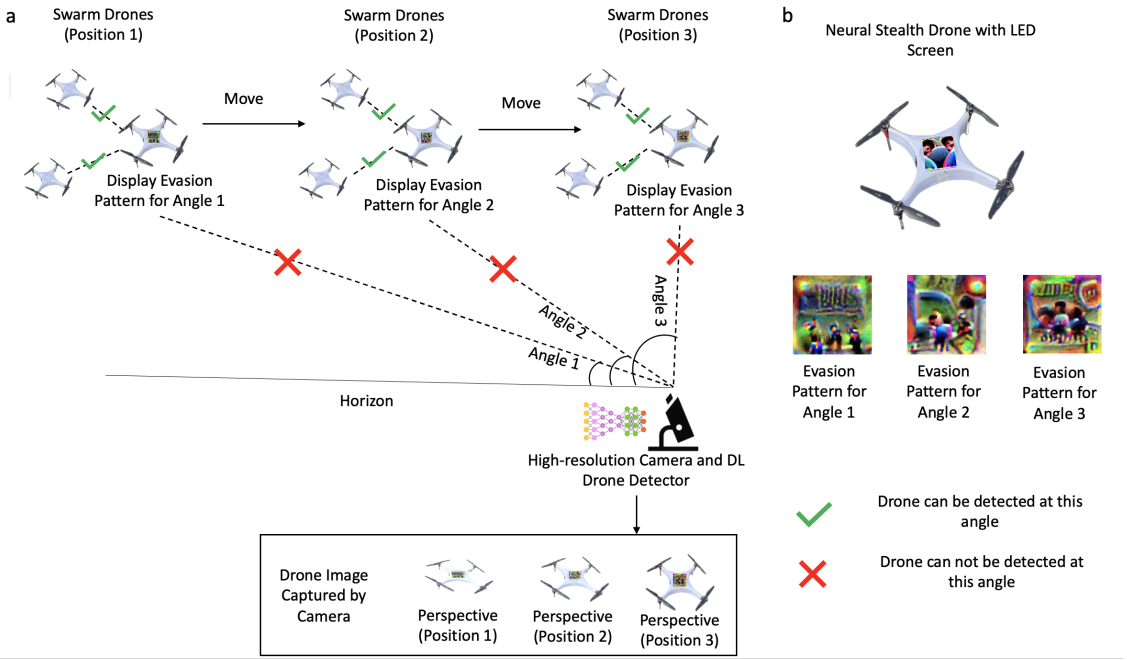


Figure 6.5: Neural Stealth Drone in Swarm Scenarios. a) I want to achieve: Assuming that the position of the drone detection system (camera and DL drone detector) is known, the evasion pattern only affects the DL inference in the drone detection system, without affecting detection among drones in swarm. b) I need: an LED screen to display different evasion patterns, and list of evasion patterns trained for different view angles. The evasion pattern to display is selected from the pattern list based on the relative position of the drone swarm and the drone detection system.

6.4.1 Evasion Patterns in Drone Swarm

As the position of the hostile drone detection system is known, the view angle of the high-resolution camera to swarm drones could be calculated. Given a camera view angle (see - Angle 1,2,3 in Fig. 6.5 a), it is possible to simulate the the image of drone swarm captured by the high-resolution camera of drone detection system (see - Drone Image Captured by Camera in Fig. 6.5 a). As shown in Fig. 6.5 b, my solution is to train a set of evasion patterns that work for different view angles. The drone will analyze the view angle of the camera in the drone detection system according to its relative position with the detection system. Then, choose the specific evasion pattern to display on the LED screen. Because the view angles of cameras on other drones are different from that of drone detection systems, the evasion pattern displayed will not work for them.

6.4.2 Result Analysis

Figure 6.6: Evasion Patterns Trained for Specific Perspectives: a-e) evasion pattern works for 65-90, 65-56, 56-50, 50-45, 42-45 degree view angle.

Table 6.2: Performance Evaluation of Evasion Patterns for Swarm Drones on Yolo V2 Generic Object Detector (threshold: 0.6).

View Angle	$65-90^\circ$	$65-56^\circ$	$56-50^\circ$	$50-45^\circ$	$42-45^\circ$	**Avg. 42-90°**
Avg. Detection Success Rate (Drone Detection System)	20.4%	28.7%	9.7%	18.5%	29.6%	**21.4%**
Avg. Detection Success Rate (Other Drones in Swarm)	80.1%	71.3%	62.2%	71.3%	71.5%	**71.3%**

The demonstration of evasion patterns trained for different view angles could be seen in Fig. 6.6. The performance evaluation of view-angle-specific evasion patterns in the virtual attack model could be seen in Table 6.2. From the result, I can see the evasion patterns trained for different view angle could reduce the detection success rate of drone detection system from 62.5% (post-canopy drone without any evasion pattern) to 21.4% (post-canopy drone with evasion patterns for specific view angles). The evasion patterns can also increase the detection success rate of other drones in the swarm to the target drone from 62.5% to 71.3%.

6.5 Methods

6.5.1 Problem Define

As shown in Fig. 6.1. I want to analyze which drone models are difficult for CNN to learn (low generalization ability) from a given drone model set. Design a new drone canopy with the common design features of "hard-to-learn" drones. Based on the new canopy, train and print an adversarial evasion pattern on the canopy to interfere with CNN inference (Yolo generic object detector and ResNet dedicated drone classifier). The whole process is demonstrated in Fig. 6.7, and method details are listed in the following sections.

6.5.2 Drone Models Image Dataset

The drone models image dataset was collected from 14 CAD models of commercial drones. During collection, I randomly apply 3D rotation to drones with a black background and capture images. 4200 images with 1800×1500 pixels resolution are collected evenly for 14 drone models (300 images per drone model). This is to avoid bias in DCGAN generalization ability by unbalanced data rate among drone models. Each image is labelled with 2 numbers, to identify the model name and image number for backtracking purposes. I call this dataset the 'raw dataset' in the following context.

6.5.3 Learn Data Distribution with Deep Convolutional Generative Adversarial Network

A deep convolutional generative adversarial network (DCGAN) is an unsupervised learning model that learns the data distribution in the training set and generates synthetic images following the same data distribution. There are two neural networks in each GAN model which are the generator G and discriminator D. In each training iteration, the generator generates a batch of random noise (vectors n) and applies a fractionally-strided convolutional layer set on them to generate synthetic images $G(n)$. Then, the discrimina-

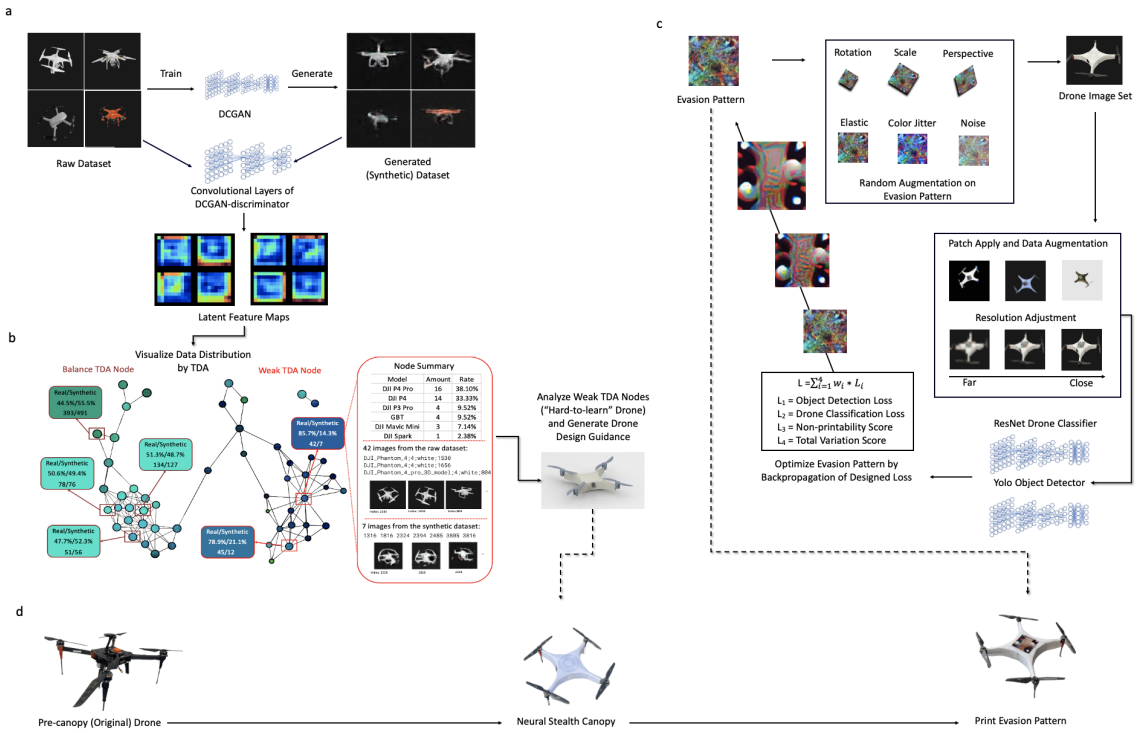


Figure 6.7: Method for Designing AI Stealth Drone using Adversarial Topology and Imagery. **(a)** Learn the data distribution of drone images from the Raw dataset with DCGAN, representing the learned data distribution by DCGAN-generated synthetic dataset. Extract latent feature maps for images from the raw dataset and the synthetic dataset separately by convolutional layers of the DCGAN-discriminator. **(b)** Visualize and express the data distribution in the raw dataset and data distribution learned by TDA. Analyse "hard-to-learn" drone models by information in weak TDA nodes, analyze their common design features and design new drone canopy. 3D print the canopy parts and assembly them onto the pre-canopy drone. **(c)** Train evasion pattern. Initialize the noise evasion pattern, and apply random evasion pattern augmentations. Stick the evasion pattern on drone images digitally, apply random data augmentation to the images, and adjust the image resolution to simulate different camera distances. Input the images into CNN models and calculate loss by a designed loss function. Optimize the evasion pattern through loss backpropagation. **(d)** The steps of producing an AI stealth drone. Prepare a post-canopy drone; assemble the 3D print drone canopy which is designed with "hard-to-learn" features; and print the trained evasion features on the canopy.

tor takes a batch of images B_i from the raw dataset $B = \{B_1, B_2, ..., B_i\}$ and the synthetic images $G(n)$ as inputs, and infer whether an image is synthetic or not ($D(G(n), B_i)$). I use Wasserstein distance as the loss function, which quantifies the divergence between ground-truth image tags T_t (True/False - From raw dataset/generated by DCGAN) and the discriminator inferences T_p. This is to guarantee the training stability of DCGAN and avoid the collapse mode [160]. The algorithm of DCGAN training with Wasserstein distance is demonstrated as follows.

Algorithm 5 Training of DCGAN with Wasserstein distance

Require: Training set $B = \{B_1, B_2 ... B_i\}$
Require: Generator G, Discriminator D, Gaussian Noise n, Wasserstein distance calculator W, Optimizer Opt
Require: Number of total epochs j
  Initialize G, D;
  for epoch $<= j$ **do**
    for $B_i \in B$ **do**
      Initialize Gaussian Noise n;
      Generate Synthetic Images $G(n)$;
      Prepare Data Tags (True or False):
      $T_t(G(n), B_i) = [\text{False} * G(n), \text{True} * B_i]$
      $T_t(G(n)) = [\text{False} * G(n)]$
      Discriminator Inference (True or False):
      $T_p(G(n), B_i) = [D(G(n)), D(B_i)]$
      $T_p(G(n)) = [D(G(n))]$
      Loss Calculation:
      $loss\_D = W(T_t(G(n), B_i), T_p(G(n), B_i))$
      $loss\_G = W(T_t(G(n)), T_p(G(n)))$
      Loss Backpropagation:
      $D \leftarrow Opt(loss\_D, D)$
      $G \leftarrow Opt(loss\_G, G)$
    end for
    epoch = epoch + 1
  end for
  return D, G

The DCGAN model in this chapter is trained on the raw drone images as the training set. Images are resized with a resolution of 64×64 pixels, and randomly divided into batches (size: 128 images). I apply the centre crop and random rotation augmentations, and also data normalization on the images. The learning rates for the discriminator and

generator are set to 4e-3 and 2e-3 respectively. The discriminator will be trained once after training the generator twice. This is to balance the ability of the generator and discriminator in generating synthetic images and discriminating images, guaranteeing the convergence of DCGAN. The training of DCGAN takes around 8000 epochs, I add a checkpoint after each 100 epochs training, to check the training of DCGAN and the quality of synthetic images. The network architectures of the generator and discriminator are listed as follows.

- DCGAN Generator
  - Input: Gaussian Noise Vector with Size 100
  - Layer 1: 4*4 Fractionally-strided Convolutions; Batch Normalization; ReLU
  - Output: Tenser 512*4*4
  - Layer 2: 4*4 Fractionally-strided Convolutions; Batch Normalization; ReLU
  - Output: Tenser 256*8*8
  - Layer 3: 4*4 Fractionally-strided Convolutions; Batch Normalization; ReLU
  - Output: Tenser 128*16*16
  - Layer 4: 4*4 Fractionally-strided Convolutions; Batch Normalization; ReLU
  - Output: Tenser 64*32*32
  - Layer 5: 4*4 Fractionally-strided Convolutions; Tanh
  - Output: Tenser 3*64*64 (RGB image)
- DCGAN Discriminator
  - Input: Tenser 3*64*64
  - Layer 1: 4*4 Convolutions; Batch Normalization; Leaky ReLU
  - Output: Tenser 32*32*32
  - Layer 2: 4*4 Convolutions; Batch Normalization; Leaky ReLU

- Output: Tenser 64*16*16
- Layer 3: 4*4 Convolutions; Batch Normalization; Leaky ReLU
- Output: Tenser 128*8*8
- Layer 4: 4*4 Convolutions; Batch Normalization; Leaky ReLU
- Output: Tenser 256*4*4
- Layer 5: 4*4 Convolutions
- Output: Model Inference Tenser with Size 1

6.5.4 Extract Latent Feature Maps to Express Data Distribution

I use the convolutional layers (except the last one, which is for result inference) in the DCGAN discriminator as the latent feature extractor. The reason is that later layers extract higher-dimensional knowledge for an overall understanding of the image. The output of the latent feature extractor (tensor - size $256*4*4$) will be stretched into a one-dimension tensor (size: 4096) for TDA use. 4200 synthetic images will be generated $S = \{G(n_1), G(n_2)...G(n_i)\}$, which is the same amount of images as the raw dataset $B = \{B_1, B_2...B_i\}$. Here, each item in S and B means a batch of data. It is to guarantee the number of synthetic data is sufficient to express the learned data distribution. The extraction of latent feature maps for the raw dataset L_B and the synthetic dataset L_S could be seen as follows. Here, I further store the tag of each image from the raw drone dataset associated with its latent feature map for backtrack use.

6.5.5 Visualization of Data Distribution with Topological Data Analysis

To visualize the data distribution represented by latent feature maps, I apply TDA as a dimension reduction algorithm. TDA analyses the structural characteristics of drones in images by topological information across features from their latent feature maps, and

Algorithm 6 Extraction of latent feature maps

Require: Raw drone dataset $B = \{B_1, B_2 ... B_i\}$;
Require: Synthetic drone dataset $S = \{G(n_1), G(n_2) ... G(n_i)\}$;
Require: Convolutional layers of DCGAN's discriminator $C = \{c_1, c_2 ... c_i\}$

```
Define L_S, L_B = {}, {}
for i <= length(B) do
    l_B = B_i
    l_S = G(n_i)
    for j < length(C) do
        l_B = c_j(l_B)
        l_S = c_j(l_S)
        j = j + 1
    end for
    L_S.append(l_S)
    L_B.append(l_B)
    i = i + 1
end for
return {L_S, L_B}
```

further cluster data with similar topological features. I input the extracted latent feature maps of both data from the raw dataset and the synthetic dataset into a Kepler Mapper [194] to visualize and compare their data distribution in a uniform TDA space. Each TDA node in this space contains latent feature maps for images from the raw dataset and an indeterminate amount of latent feature maps for images from the synthetic dataset. I classify TDA nodes as follows, according to the ratio of the number of hidden feature maps for raw and synthetic data in each node.

- Weak TDA node: none or very rare latent feature maps are for data from the synthetic dataset
- Balance TDA node: contains a fair number of hidden feature maps for both raw and synthetic dataset
- Failed TDA node: contains only hidden feature maps for data from the synthetic dataset

A weak TDA node means the generalisation ability of DCGAN on data in this TDA node is weak; a balance TDA node indicates DCGAN has sufficient generalisation ability on

this kind of data; a failed TDA node means DCGAN generate synthetic data outside the data distribution in the raw dataset, which indicates a failure in DCGAN training. As data in each TDA follows similar topological features, I treat each TDA node as a sub-region of the data distribution. Thus, the generalization ability of DCGAN on data with different topological features could be seen from the classification of TDA nodes. I filter and extract all weak TDA nodes, analyze them individually, and describe their data features by the tags (drone model) of data (from the raw dataset) whose latent feature maps are in this node. By summarizing the information extracted from these weak TDA nodes, I can analyze the images of which drones do current DCGAN have low generalization ability on. According to the theory of this chapter, these drones are further defined to be ”hard-to-learn” drones for general CNNs.

6.5.6 Quantify the Curvature Metric of Drone Models

I first grayscale all the drone images, and uniform them into the same image size (64×64 pixels). Then, I apply 8 curve extractors (convolution kernels, each with 5×5 kernel size, 1-pixel stride) designed for curve extraction on all the images. In the output of each curve extractor, a higher score indicates a higher similarity between the current image receptive field and the curve features of the current convolution kernel. Then, the average score of the 8 curve extractors over all images of a certain drone is used as the curvature metric for the current drone model.

6.5.7 Canopy Design, 3D printing and Assembly

The research team manufactured a physical drone canopy through 3D printing. PTC CREO 3D modeling software was used for canopy design. During the design process, the number of curved surfaces and arcs was maximized. After the design is completed, the 3D model is sliced using CURA slicing software, and the .gcode code is generated and input into the 3D printer for printing and manufacturing. In the manufacturing process of canopy, the research team used PLA material, cut the 3D shell into eight parts, and

printed them on the Anycubic Chiron 3D printer.3D printer parameters are set to nozzle temperature 236°C, hot bed temperature 60°C. After 168 hours of printing, the eight parts were completely printed.

6.5.8 Image Collection for Post-canopy Drone

I capture images (square shape images) for the post-canopy drone with an iPhone native camera, without any filter effects. The images are captured indoors and outdoors, in different weather and lighting conditions to eliminate unexpected bias. Then, a segmentation-based image background removal method is applied - keeping the same background as images in the raw dataset for further TDA analyses and other purposes. I name this dataset the 'post-canopy dataset' in the following context.

6.5.9 Performance Evaluation of Neural Stealth Canopy

Following the same process of previous latent feature maps extraction, I use the same feature extractor on the post-canopy dataset. Then, I put the latent feature maps of both images from the post-canopy dataset and raw dataset together into a TDA mapper, to analyse the closest drone model to the post-canopy drone in the latent feature space. In this TDA space, each TDA node contains latent feature maps of images from the post-canopy dataset and/or the raw dataset. I classify TDA nodes into:

- Class A: contains latent feature maps of only the post-canopy drone or raw dataset drones.
- Class B: contains both latent feature maps of the post-canopy drone and more than 7 of the raw dataset drones
- Class C: contains both latent feature maps of the post-canopy drone and less than 7 of the raw dataset drones

I use information in Class C nodes to analyze the similarity in latent feature space of the post-canopy drone and the other 14 drone models, the reason as follows. Class A will not give any information on the similarity between the post-canopy drone and other drone models (in the raw dataset), because the images of the post-canopy dataset do not overlap with any images from the raw dataset on the topological features in the current TDA node. Although Class B contains both latent feature maps of the post-canopy drone and other drones, more than half of the drone models have images placed in this TDA node means the topological feature in this node is generic ones that are met by the majority of drone designs. Class C contains non-generic topological features that are only met by the post-canopy drone and several other drone models, which means the post-canopy drone has higher similarity to these drones in latent features compared with other drones in Class B nodes.

I rank the similarity of the post-canopy drone to other drone models based on the number of images of other drone models in all class C nodes, since the number of images of different drone models in the raw dataset is the same (300 images per drone model). I filter all Class C TDA nodes, gather all images, and trackback their tags (name of drone model). I computed and ranked the proportion of different drone models in this dataset, which I believe is positively correlated with their similarity to the post-canopy drone.

6.5.10 Dedicated Drone Classifier and Generic Object Detector

The dedicated drone classifier used in this chapter is based on ResNet-18 architecture, trained to classify 14 drones. The output of the ResNet-18 drone classifier is 14 float numbers, each number represents the classification score of the correlated drone model, and the drone model with the highest score will be the classification result. During the training, I use pre-trained weights at model initialization to avoid a cold start, and fine-tune all layers during training. **Dataset**: The training set is the raw dataset (4200 images, 14 drone models). The validation set contains 294 images (21 images per drone model). To avoid over-fitting and strengthen classifier robustness and generalisation ability to out-

of-distribution data, I apply random crop, random colour jitter, and random rotation as data augmentations. **Training settings**: The input image size for the classifier is set to 224×224 pixels, and the training rate is set to 1e-4. Cross-entropy loss and Adam optimiser are used during classifier training. During the training, the classifier converges after 8 epochs of training.

I use a YOLO V2 model pre-trained on the COCO dataset (330k images and 80 categories) as the generic object detector in this project. According to [195], well-trained object detection models on multi-category data are with generalization ability for out-of-distribution object detection tasks. The output of the YOLO V2 object detector on one image is several 85-dimension tensors. Each tensor represents a detection box, whose possibility of having an object is higher than the pre-defined threshold. The order of the items in the output tensor is: (1-2) x,y coordinate of the top-left vertex of the box; (3-4) length and height of the box; (5) object confidence score; (6-85) possibility of this object belong to each category. Since YOLO V2 is used as a generic object detector, I are only interested in object confidence scores (5).

6.5.11 Train Evasion Patterns

The primary method is from [185]. I use the post-canopy dataset for evasion pattern training. Firstly, the initial evasion pattern is generated with random noise (3 channels for RGB, width and length depending on resolution). In order to ensure that the evasion pattern is correctly pasted on the drone in the image, the position of the evasion pattern on the image is calculated according to the coordinates of the detection box in Yolo image labels (5 numbers: (1) the drone class, (2-5) the detection box in the upper left corner point coordinates and length and width). According to [196], The RGB colour of the evasion pattern on the computer is different from the physical colour printed b printers and captured by the camera, which influences the practical usability of the evasion pattern. The solution is to apply a regression model on RGB colour to simulate the physical colour of the evasion pattern. Secondly, the evasion pattern will be randomly augmented in

the following order to prevent over-fitting and increase evasion pattern robustness and invariance:

1. Median pooling with random kernel size
2. Random scaling
3. Random colour jitter in brightness, contrast and saturation
4. Add random noise
5. Random rotation
6. Random perspective transformation
7. (Optional) Random elastic transformation [197]

Thirdly, stick evasion patterns on the images, randomly adjust the resolution of images, and input the images to the Yolo V2 object detector and the ResNet-18 drone classifier. The evasion pattern will be directly optimized by the backpropagation of the designed loss. In the design of loss, I consider the following aspects:

- Loss 1 (L_1): Object detection loss. This loss is computed on the output (5th dimension) of the Yolo V2 object detector, which is the maximum confidence score of detection boxes for an image. In each detection box, if the confidence score is below a pre-defined confidence threshold in the Yolo V2 model, the model does not report objects detected in that box. As the goal is to let the drone escape from CNN detection, the L_1 needs to be reduced during the optimization of the evasion pattern.
- Loss 2 (L_2): Drone classification loss. This loss is the cross-entropy loss between the output of the ResNet-18 dedicated drone classifier and ground-truth data labels. A low cross-entropy loss means the predictions from the CNN model are close to the ground-truth labels. Although the design of the new drone canopy already mislead the drone classifier in inference, most are classified into drone model 4 and

model 1 (64% and 26%). The goal of this chapter is to further confuse the CNN inference to other wrong drone models (according to the result, images are misclassified into drone models 0, 3, 4 with rate: 31%, 25%, 34%), which means the L_2 needs to be raised during the optimization of the evasion pattern.

- Loss 3 (L_3): Non-printability score. L_3 indicates to what extent the RGB colours in evasion pattern pixels P could be represented correctly by a printable set of colours C in common printers [198], as printers can only print a limited number of colours compared with RGB colours. This is calculated by:

$$L_3 = \sum_{p \in P} \min_{c \in C} abs(p - c) \tag{6.1}$$

  Here, lower L_3 means the colour in evasion pattern pixels could be printed more accurately. The L_3 needs to be reduced during the optimization of the evasion pattern.

- Loss 4 (L_4): Total variation score. L_4 indicates how smooth the colour transitions among close pixels [199]. This loss could be used to prevent the generated evasion pattern from noisy patterns, and also help to eliminate meticulous colour features. This is calculated by:

$$L_4 = \sum_{i,j} \sqrt{(p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2} \tag{6.2}$$

  Here, lower L_4 means the colour transitions in the evasion pattern is more smooth. The L_4 needs to be reduced during the optimization of the evasion pattern.

The overall designed loss function could be seen as follows:

$$L = \sum_{i=1}^{4} w_i \times L_i \tag{6.3}$$

Each loss is granted with a weight to balance different evasion pattern capabilities. During the optimization of evasion patterns, the direction of gradient descent will lead to a

reduction of L. In this chapter, the weights are set to 0.6, -0.4, 0.1, and 1.5 respectively.

6.5.12 Performance Evaluation of Evasion Pattern

The evaluation of evasion pattern is based on two attack models, which are:

- Virtual Attack Model: the evasion pattern is post-hoc stuck on the canopy digitally on captured drone images.
- Real-world Attack Model: the evasion pattern is physically printed and stuck on the canopy before the capture of drone images.

To compare the performance of evasion features trained with different augmentation methods (with or without perspective), I trained two evasion patterns for evaluation:

- Evasion pattern trained with perspective: the training of evasion pattern applies the perspective augmentation and also other additional augmentation methods mentioned in Section 4.11
- Evasion Pattern trained without perspective: no perspective nor other additional augmentation methods during training of evasion pattern

I use the trained two evasion patterns to generate both virtual and real-world attack models for performance evaluation. The virtual attack models are evaluated using the post-canopy dataset, while the evaluation of the real-world attack models uses new images collected from drones printed with physical evasion patterns.

6.5.13 Train Evasion Pattern Sets for Swarm Drones Scenarios

The evasion patterns for different perspectives will be trained separately based on a revised version of the method detailed in Chapter 6.5.12. The changes are listed as follows:

- For each perspective, I adjust the aspect ratio of the original evasion pattern by affine transformation to simulate the imagery distortion caused by this perspective.

Then, stick the transformed evasion patterns on all drone images in the training set (generate 'system dataset'). This is to simulate the images captured by the camera of the drone detection system.

- The original evasion pattern with random perspective transformations will be applied to all drone images in the same training set (generate 'other drone dataset'). This is to simulate the images captured by the cameras on other drones in the drone swarm.

- The object detection loss L_1 in the designed loss function will be split into two parts, which are the detection loss of DL drone detector on 'system dataset' L_{1s} and detection loss on 'other drone dataset' L_{1o}. During the patch training, L_{1s} and L_{1o} will be assigned with a positive and negative weight respectively (1 and -0.5 in the experiment in this chapter). L_{1o} is designed to force the evasion patterns to only works for a specific perspective, as I found that the evasion patterns trained with only L_{1s} will slightly reduce the detection confidence score of images captured with other perspectives (detection by other drones in the same swarm).

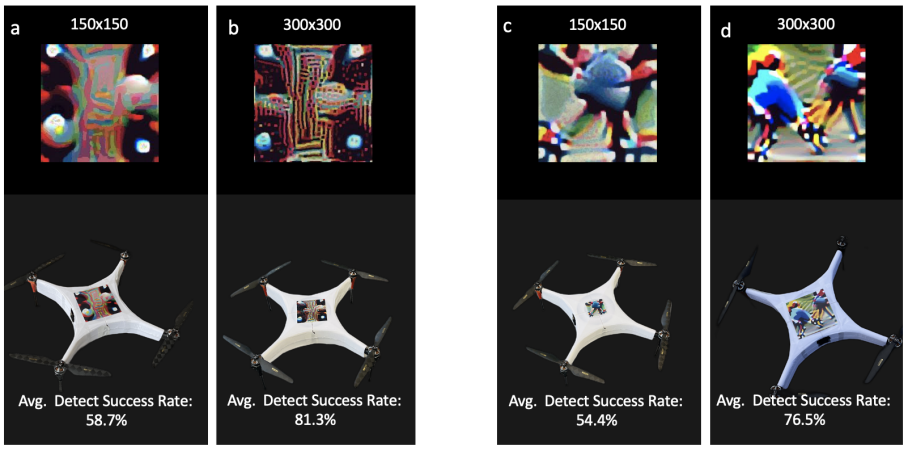


Figure 6.8: Compare High-resolution and Low-resolution Evasion Patterns. **(a, b)** Two evasion patterns trained with the same loss function but different resolutions, I can see the higher resolution pattern works not as robust as the lower one in real-world attacks. **(c, d)** I slightly change the parameters in the loss function, and I can see the higher resolution pattern still works not as robust as the lower one in real-world attacks.

6.6 Supplementary Information

6.6.1 High-resolution Evasion Pattern vs Low-resolution Pattern

In the experiments, I found that high-resolution evasion patterns always converged to a lower loss compared with low-resolution patterns during training. This means, in virtual attack models, under training with the same loss function, high-resolution evasion patterns are more effective than low-resolution evasion patterns. However, as demonstrated in Fig. 6.8, high-resolution evasion patterns can not maintain robustness in real-world attack models. This leads to low-resolution patterns outperforming high-resolution patterns in real-world attacks. I suppose this is because the meticulous features in high-resolution patterns are easily corrupted by distortions caused by ambient lighting and camera quality during image capture.

6.6.2 RGB Color and Physical Color

As shown in Fig. 6.9, there is a difference between the RGB color of the evasion pattern and its physical color (print the physical evasion pattern by a printer, then capture it by

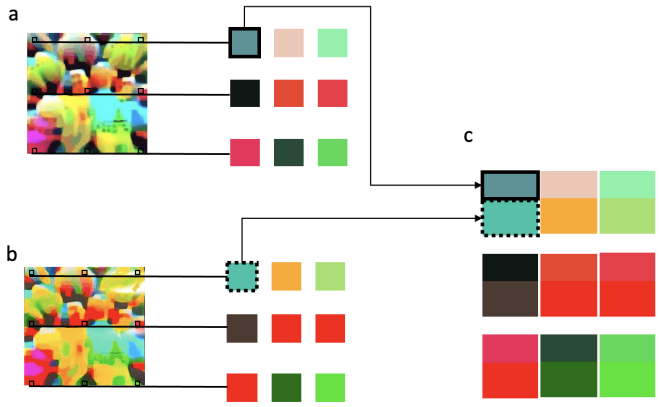


Figure 6.9: RGB Color and Physical Color of Evasion Patterns. **a** Evasion pattern in RGB color, and color sampling of evasion pattern. **b** Physical color of the same evasion pattern, and color sampling with the same sample points. **c** Comparison of sampled colour from RGB color evasion pattern and physical evasion pattern.

a camera). This will lead to a low transferability of evasion patterns from virtual attack models to real-world attack models. [196] proposed a method to use a regression model on the mapping relationship between RGB color and physical color. This method could simulate the color performance of a physical evasion pattern based on its RGB information, which eases the difference between them.

6.6.3 Evasion Patterns and Ambient Lighting

In the experiments, I found that different ambient lighting conditions could change the color performance of evasion patterns and lead to the failure of evasion patterns, as demonstrated in Fig .6.10. I provide a color performance (hue, saturation, value) analysis on both the RGB evasion pattern and the real-world evasion pattern (under a strong ambient lighting condition). I can see although the brightness (value) of the real-world evasion pattern is higher than the RGB evasion pattern, its has a huge drop in color saturation and color hue. And, this effect from ambient lighting is hard to simulate by conventional augmentation methods. I suppose this could be solved by GAN-based augmentation methods, and leave it as further work.

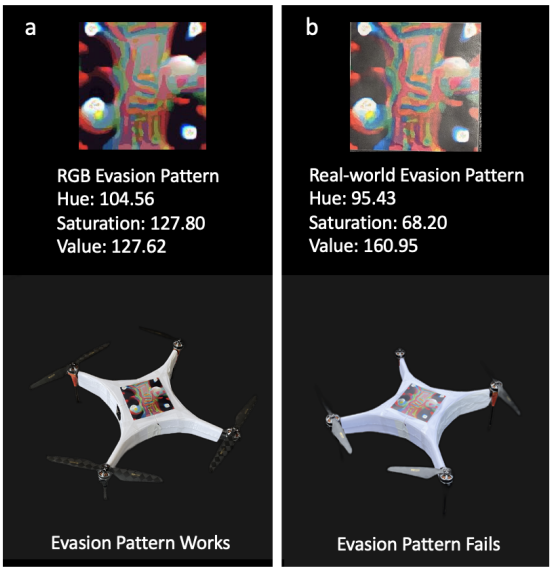


Figure 6.10: Impact of Ambient Lighting on Physical Evasion Patterns. **a** the RGB evasion pattern and its color performance (hue, saturation, value). **b** the physical evasion pattern captured under a strong ambient lighting condition, and its color performance (hue, saturation, value). I found that the color performance influenced by ambient lighting could lead to the failure of evasion patterns.

6.6.4 Flexible Shape Boundary of Evasion Pattern

Actually, the evasion patterns could be trained with different shapes, to achieve a higher canopy surface coverage and higher deploy flexibility. In Fig .6.11 a and c, I demonstrate two evasion patterns trained with square-shape and star-shape respectively.

6.6.5 Elastic Transform Augmentation

I demonstrate the effect of elastic transformation augmentation of evasion patterns in Fig .6.11 b and d. Elastic transformation augmentation could grant the evasion pattern with more robustness to some tiny surface changes of drone canopies (e.g. soft-canopy drones) or some image shape distortion during image capturing. The difficulty is maintaining the gradient information of evasion patterns during the transformation [197].

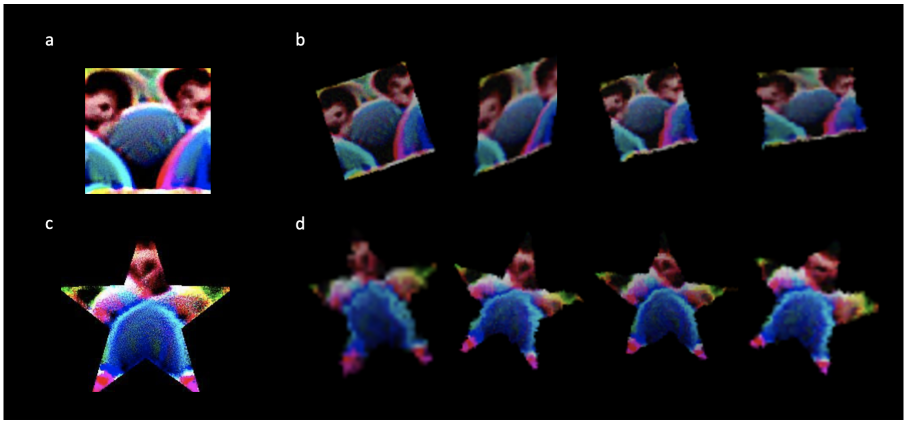


Figure 6.11: Demonstration of Evasion Patterns with Different Shape, and Elastic Transformation Augmentation. **a** Square-shape evasion pattern in RGB. **b** Samples of random elastic transformations for the square-shape evasion pattern. **c** Star-shape evasion pattern in RGB. **d** Samples of random elastic transformations for the star-shape evasion pattern.

Chapter 7

Overall Discussion

According to the related submitted works detailed in Chapter 2-6, the overall discussion of the thesis is organized into: summary of key findings, academic and real-world impacts, potential implementations and research limitations.

7.1 Summary of Key Findings

This thesis achieves efficient and high-trustworthiness deep learning for drones using efficient drone informatics. There are several key findings in this thesis: A theoretical QoT metric and an AI trustworthiness supervision protocol are proposed based on a study of AI trustworthiness and XAI. DL energy consumption scaling law to different model architectures and settings is analyzed by theoretical TOs metric. A GAN-TDA method is proposed to analyze the learning efficiency of drone-related DL models to different drone images; Efficient drone informatics is explained and guides the additional drone data collection to improve DL model performance more effectively. Sub-key findings are detailed according to research areas as follows.

In AI Trustworthiness, a novel theoretical metric QoT is proposed with consideration of both AI physical trust and emotional trust from human users with different backgrounds (pure AI users and developers). In this research, the explainability of AI is investigated based on the review of different XAI methods. AI explainability is theoretically quantified

as a metric that contributes to AI physical trust. Also, AI users are classified according to their backgrounds. Human satisfaction with AI explanations and performance is quantified and counts into emotional trust. A lifelong AI trustworthiness protocol is proposed later.

In DL Explainability, the feasibility of using shape-learnable AFs to increase the NN transparency is discussed. A GP neural network with KST-based topology is proposed to achieve partial explainability. It is found that the explainability of NN is highly related to the network complexity (here, the number of units), despite the Gaussian Process activation function is fairly transparent. This also validates the relationship between network complexity and explainability proposed in Chapter 2.

In DL Energy Efficiency, the mapping relationship between operations (linear and non-linear) led by DL architecture and settings to the basic operations in computational hardware is studied. A theoretical transistor operations metric is proposed to quantify the hardware workload of running different DL models, which further be used to estimate the practical energy consumption of DL models.

In DL Learning Efficiency, the learning behaviour of general convolutional layers on drone images is analyzed with GAN. TDA explain the generalization ability of GAN by visualizing the data distribution of both the training dataset of GAN and the GAN-generated dataset. This forms a GAN-TDA method to analyze which kind of data is poorly learned by DL models and is critical for improving the performance of DL models. These critical data are further explained by data tags, resulting in the discovery of hard-to-learn drones.

Efficient Drone Informatics is dissected from common design features of hard-to-learn drones. Based on the extracted efficient drone informatics, a neural stealth drone canopy using adversarial topology and imagery is designed to fool CNN-based drone detectors and classifiers, which is further extended for swarm drone scenarios.

7.2 Academic Impacts

This thesis shows a new way to analyze and explain the learning behaviour and energy consumption of DL models. This increases the transparency of DL models and contributes to the understanding and research of DL trustworthiness. This study is also a step forward in designing objects using interpreted human-understandable DL latent features.

Physical trust and emotional trust provide researchers with a holistic understanding of how different factors contributes to AI trustworthiness. The analysis of AI QoT allow researchers to focus more on trustworthiness when developing new AI algorithms. QoT also provides them with a more holistic AI evaluation metric compared with conventional performance metrics. The partial explainable GP network informs researchers of a new way to design high-transparency NNs. This GP NN could be used as a surrogate model to provide structural reasoning for other low-transparency DL models. At the same time, it shows the potential to help researchers to discover complex mapping relationships among high-dimensional inputs and outputs.

TOs explains how DL architecture and setting resulting the energy consumption of hardware. This can help researchers in designing high-efficiency learning structures and hardware. At the same time, TOs could be used as an energy metric, to be considered in loss function design for an energy-aware DL model pruning.

The GAN-TDA provide researchers with a novel method to understand the learning behaviour of general CNN-based models by visualizing the evolution of generalization ability of convolutional layers during model training. It helps to explore and explain the data that DL models have low generalization ability on.

The design of the neural stealth drone by efficient drone informatics informs researchers with a understanding of how the weak generalization ability of DL models could be used to generate attacks. This could inspire them in designing new attack methods, or new defence methods to influence DL trustworthiness. At the same time, the combination of the above findings may generate new attacks on DL explainability, which further lead to a reduction in DL trustworthiness. At the same time, evasion patterns can be

reverse-trained to improve AI performance in detecting and classifying objects.

7.3 Real-world Impacts

QoT helps developers analyze user-aware trust requirements for real-world AI products, and provides them with a holistic evaluation metric for AI product development and testing. The theoretical lifelong AI trustworthiness supervision protocol lets the supervision of AI models and products become more systematic. This also helps developers in developing trustworthy AI products, which improve the willingness of human users to trust their AI products. GP NN can be used to develop products for explaining AI decisions, e.g., tell people how the AI credit evaluation systems make decisions, and which factor they should improve to have a higher score.

TOs let developers estimate the DL energy consumption without deploying them into practical hardware, this could avoid unnecessary experiments to save energy. The process of calculating TOs also emphasize that each additional TOs operation will bring additional energy cost. This reminds developers to consider energy consumption when making the trade-off between space and time of calculation hardware designs, resulting in fast and energy-efficient calculation hardware.

The GAN-TDA provide developers with a method to analyze the weakness of CNN-based AI products and which data is significant to address this weakness. This could significantly reduce the time and money cost of developing and improving AI products.

The design process of the neural stealth drone by efficient drone informatics could be transferred to develop other hard-to-learn or easy-to-learn products. For example, with Autopilot gaining traction recently, objects like street signs could be assembled with some easy-to-learn features either in printing or shaping to increase the accuracy of AI in detecting them. Vests that enhance AI recognition could be invented to improve the safety of workers.

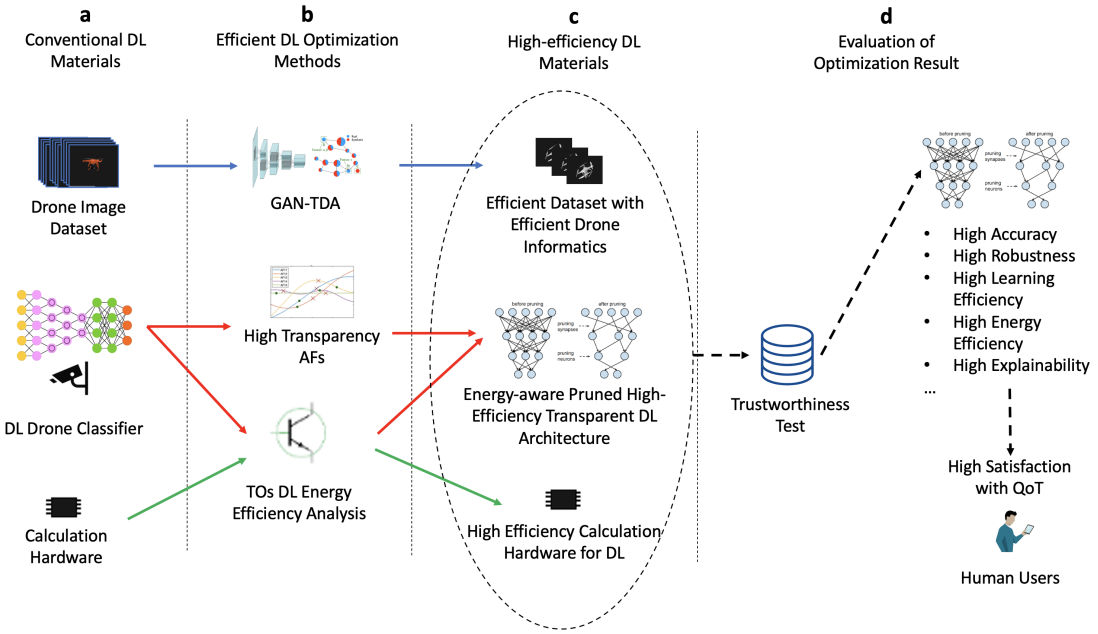


Figure 7.1: Potential Implementation 1: Efficient Trustworthy DL Drone Classifier

7.4 Potential Implementation

The research outcome of this thesis could be applied to real-world use to improve the performance of practical DL products and human lives. Here, two potential implementations are briefly introduced: 1) an efficient trustworthy DL drone classifier and 2) an AI-safe vest to protect road workers

7.4.1 Implementation 1: Efficient Trustworthy DL Drone Classifier

The research outcomes in this thesis could be applied to the training process of conventional DL drone classifiers to improve the training efficiency and trustworthiness of the final DL product.

As shown in Fig. 7.1 a, the material for training a DL drone classifier include a well-labelled drone image dataset, a CNN-based model, cameras to collect visual input, and hardware to process all the DL calculation tasks. However, the conventional training process is always time and cost-consuming. As shown in Fig. 7.1 b and c, findings in this thesis could provide the GAN-TDA method to analyze efficient drone informatics, and

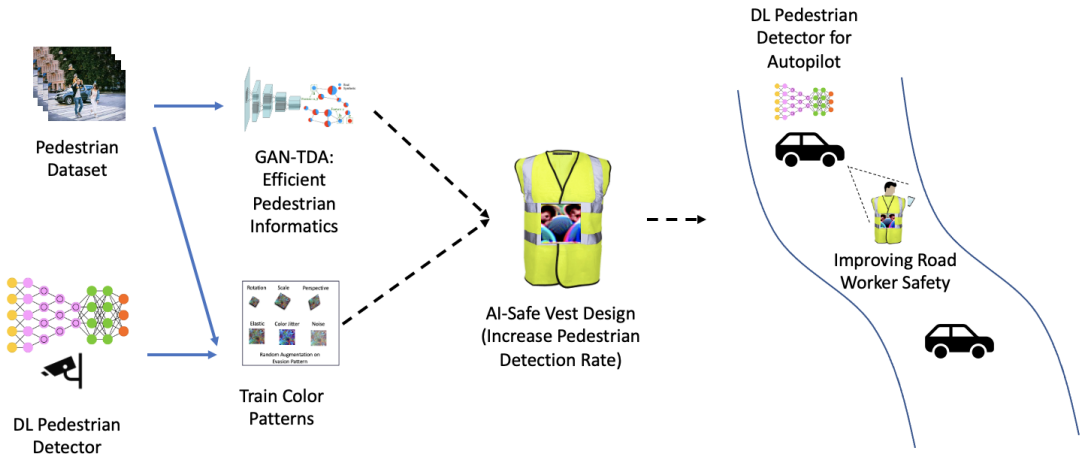


Figure 7.2: Potential Implementation 2: AI-safe Vest to Protect Road Worker

guide the collection of high-efficiency training datasets. TOs and High transparency AFs could be used to choose high-efficiency and transparency DL architecture. At the same time, TOs could also guide the optimization of hardware operation logic (e.g. both data types like FP-32/64 and GELU-specific logic) to meet the requirements of the current DL model with high processing efficiency in time and energy cost. After all the high-efficiency DL materials are ready (see - Fig. 7.1 c), the optimized DL model and associate materials will be uploaded to the trustworthiness test. As shown in Fig. 7.1 d, the DL drone classifier trained on high-efficiency materials is supposed to with high trust factors (e.g. accuracy, robustness, explainability), which increase human satisfaction with QoT.

7.4.2 Implementation 2: AI-safe Vest to Protect Road Workers

As CNN could be used as part of autopilot to detect obstacles and humans, their detection accuracy is critical to road workers' safety. The research outcomes in this thesis could be applied to design an AI-safe vest to protect road workers.

As shown in Fig. 7.2, the efficient pedestrian informatics could be analyzed by the GAN-TDA method on pedestrian images, which reveals which clothes design features are easy to be learned by CNNs. Also, the dataset and a DL pedestrian detector could be

used to train color patterns to increase the detection of humans. An AI-safe vest could be designed on both efficient pedestrian informatics and color pattern, to increase the detection success rate and robustness of CNN-based pedestrian detectors. This means that a road worker in an AI-safe vest has a lower chance of being miss detected by a CNN model, compared with workers in conventional work clothes.

7.5 Limitations

Comparing the trustworthiness and quantifying the trust factors of AI models with different algorithms under fair criteria is still an open challenge (e.g. complexity of tree-based and DNN models is hard to be uniform). This means, analyse the scaling of trustworthiness and QoT among models with the same model type is achievable, while cross-model-type analysis is still theoretical work. TOs model is currently only applicable for single-core scenarios, and only for computational energy consumption. As the weak point and slow-learned features detected by GAN-TDA are only for general convolutional layers, without any consideration of inference steps happen in later full-connection layers. This means, the analysed hard-to-learn latent features are generic for the feature extraction process, but not guaranteed to be critical for DL inference layers fine-turned on different targets. This also means, the GAN-TDA method is not applicable currently for DL models targeting other input data types (e.g. DNN for pure statistical data, transformers). Efficient drone informatics only considers imagery information currently. The explanation of hard-to-learn drone design features by the human expert is verified to be valid, but it is still not comprehensive. This means other drone design features may also contribute to efficient drone informatics. The training method of evasion patterns is only for flat surfaces, currently not invariant when partially blocked, and not applicable for shape changes in foldable drones.

Chapter 8

Conclusion & Future Work

In this thesis, efficient drone informatics in DL is discussed and discovered, which achieves an efficient training of high-trustworthiness DL models. The finding of this thesis is contributed by research in AI trustworthiness, DL explainability, DL learning efficiency and DL energy efficient. Two practical experiments are presented to verify the correctness of efficient drone informatics. First, an efficient additional drone dataset is collected under the guide of efficient drone informatics. This dataset achieves a faster performance improvement in a DL drone classifier compare with the randomly collected drone dataset. Second, a neural stealth drone canopy is designed with the reverse use of efficient drone informatics and is effective in interfering with DL inferences. It is shown that 1) AI trustworthiness, DL explainability, and DL efficiency are correlated with each other, and all highly connect to data. 2) AI trustworthiness can theoretically be quantified through physical trust and emotional trust, thus providing a more comprehensive performance metric for AI. 3) Flexible AFs and appropriate NN topology help improve DL transparency. 4) As an energy metric directly analyzed from model architecture and settings considering both linear and nonlinear operations, TOs works better than conventional energy metrics in estimating DL energy consumption. 5) Efficient drone imagery informatics analyzed by the GAN-TDA method is proven to be effective. 6) Efficient drone informatics could be used to improve or reduce the trustworthiness of drone-related DL models efficiently.

It is believed that the methodology and research results of this thesis can be extended to research fields other than DL and swarm drones.

In further work, the theoretical QoT metrics and trustworthiness supervision protocol will be improved for practical use. GP NN should be improved in energy efficiency with the research output of TOs. Other energy-efficient flexible AFs will be discussed. The energy consumption of DL models in multi-core scenarios will be studied. The generalization ability analysis process by GAN-TDA could be transferred into feedforward NN usage. Evasion patterns should be trained and expanded with more functionalities, and increase the usability of evasion patterns in the real world. As the explainability of evasion patterns start to acquire research attention in [200], more method to explain the latent information in evasion patterns should be developed. This may also be helped by the GAN-TDA method to analyze the impact of evasion patterns on feature distributions of images. The distance in TDA space could be used in loss functions to develop a new attack model. The new attack is supposed to interfere with DL inference by using evasion patterns to change the imagery structural topology information of objects. As evasion patterns are trained with latent information in DL models, the feasibility of using it as a new XAI method will be discussed. Efficient informatics in DL will be applied to more application areas (e.g. intelligent transportation systems).

.1 Extra Data

.1.1 Performance of TOs Model in DNN Energy Consumption Estimation Under a Multi-core Running Environment

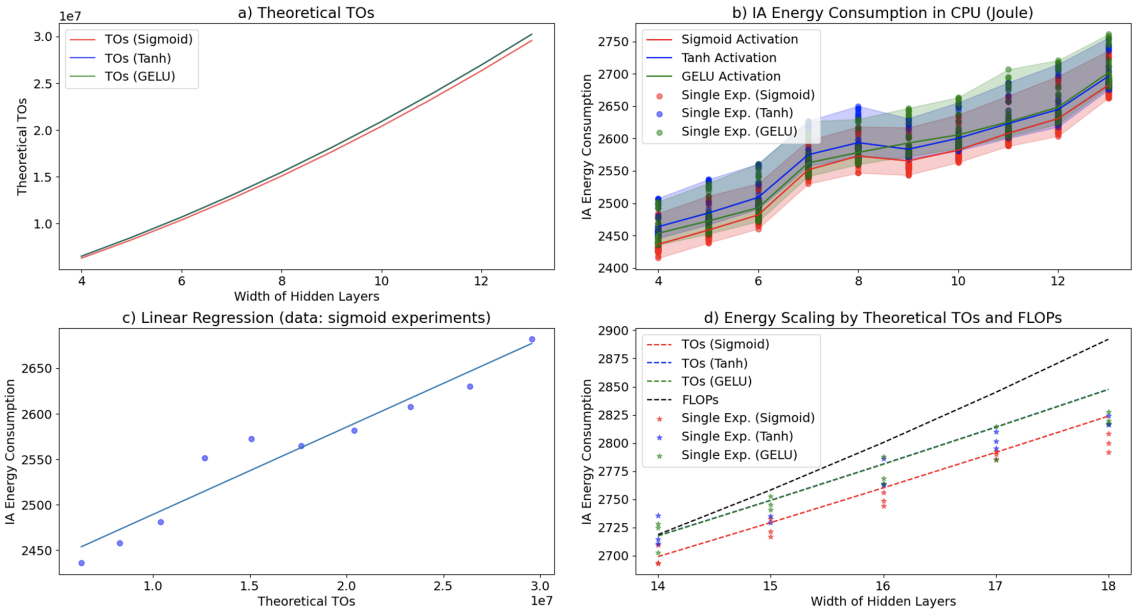


Figure 1: Performance of TOs Model in DNN Energy Consumption Estimation Under a Multi-core Running Environment (8 core Intel CPU). a) Theoretical TOs to the width of hidden layers in a DNN model. b) The calculation energy consumption in IA core to the width of hidden layers in a DNN model, the energy consumption of DNN model with width 7 and 8 is higher than with width 9. c) The linear regression model trained to fit the relationship between TOs and IA energy consumption. d) The prediction of the energy consumption of DNN model with 14-18 width by the TOs model.

.1.2 Extra Evasion Patterns Trained with Different Loss Functions

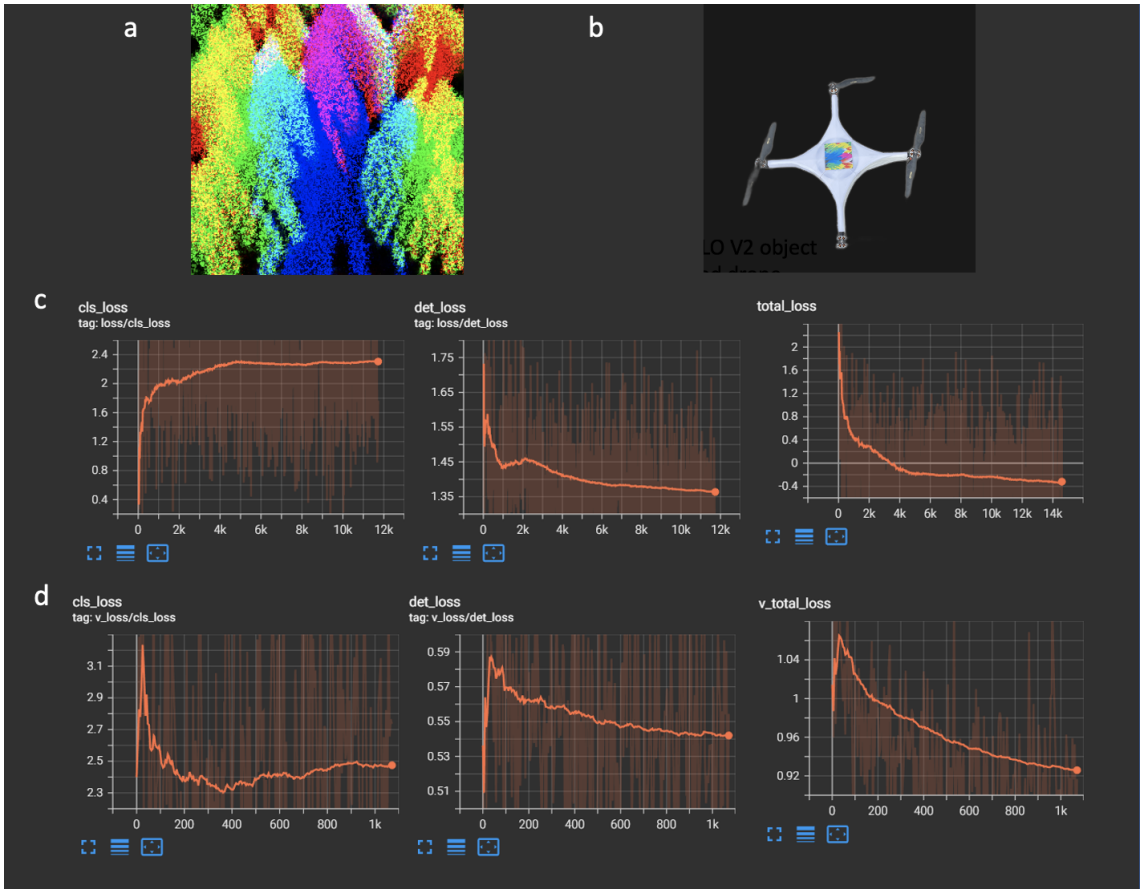


Figure 2: Extra Evasion Patterns Trained with Different Loss Functions - Pattern 1. a) The evasion pattern. b) The evasion pattern in real-world attack model. c) Performance metrics during training on training set. d) Performance metrics during training on validation set. The pattern is trained with DL classification loss and detection loss (with weight 1 and 1 respectively) and a high learning rate (0.5) on images for all drone models. During the training, a random rotation augmentation (-20 to 20 degree) and random affine augmentation (to simulate perspective) is applied. This evasion pattern fails in real-world attack models (avg. detection success rate of the drone with this evasion pattern from a Yolo V2 object detector is over 90%).

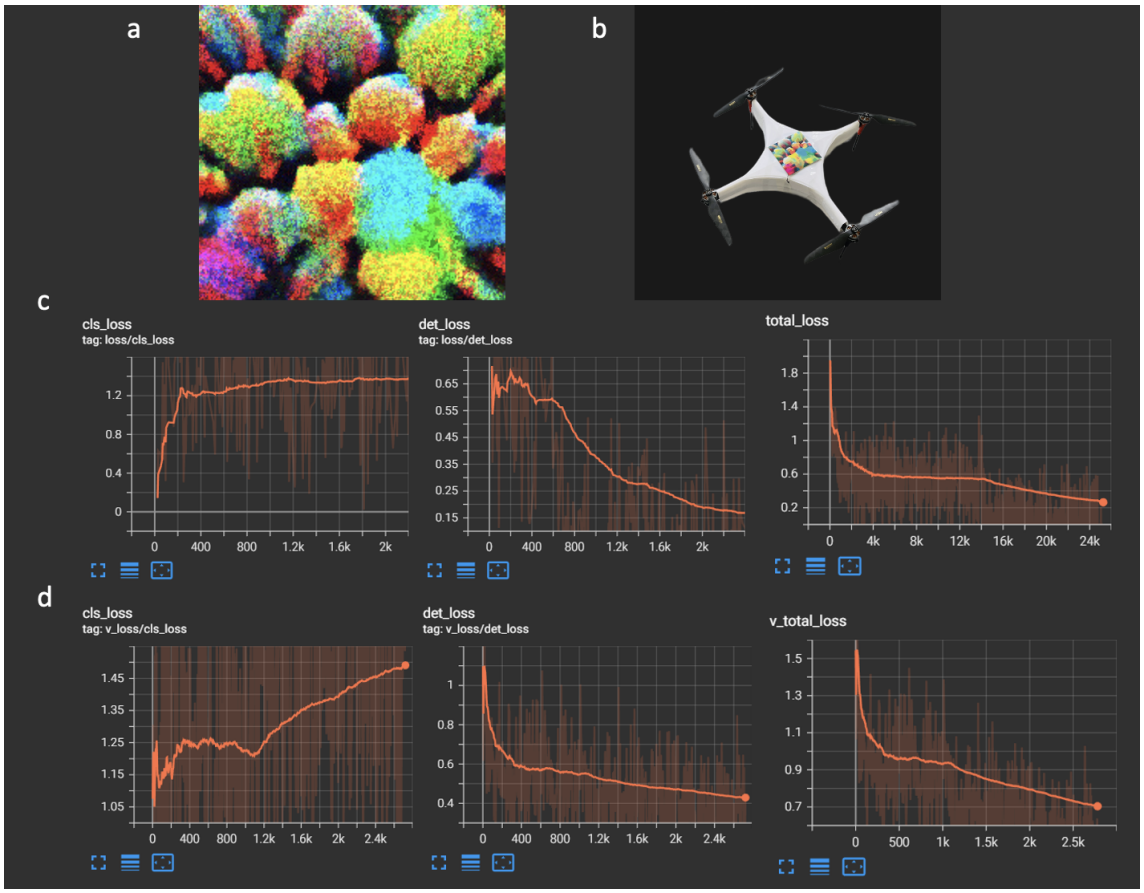


Figure 3: Extra Evasion Patterns Trained with Different Loss Functions - Pattern 2. a) The evasion pattern. b) The evasion pattern in real-world attack model. c) Performance metrics during training on training set. d) Performance metrics during training on validation set. The pattern is trained with DL classification loss and detection loss (with weight 1 and 1 respectively) and a reduced learning rate (0.3) on images for all drone models. During the training, a random rotation augmentation (-45 to 45 degree) and random affine augmentation (to simulate perspective) is applied. This evasion pattern fails in real-world attack models (the detection success rate of the drone with this evasion pattern is higher than that without).

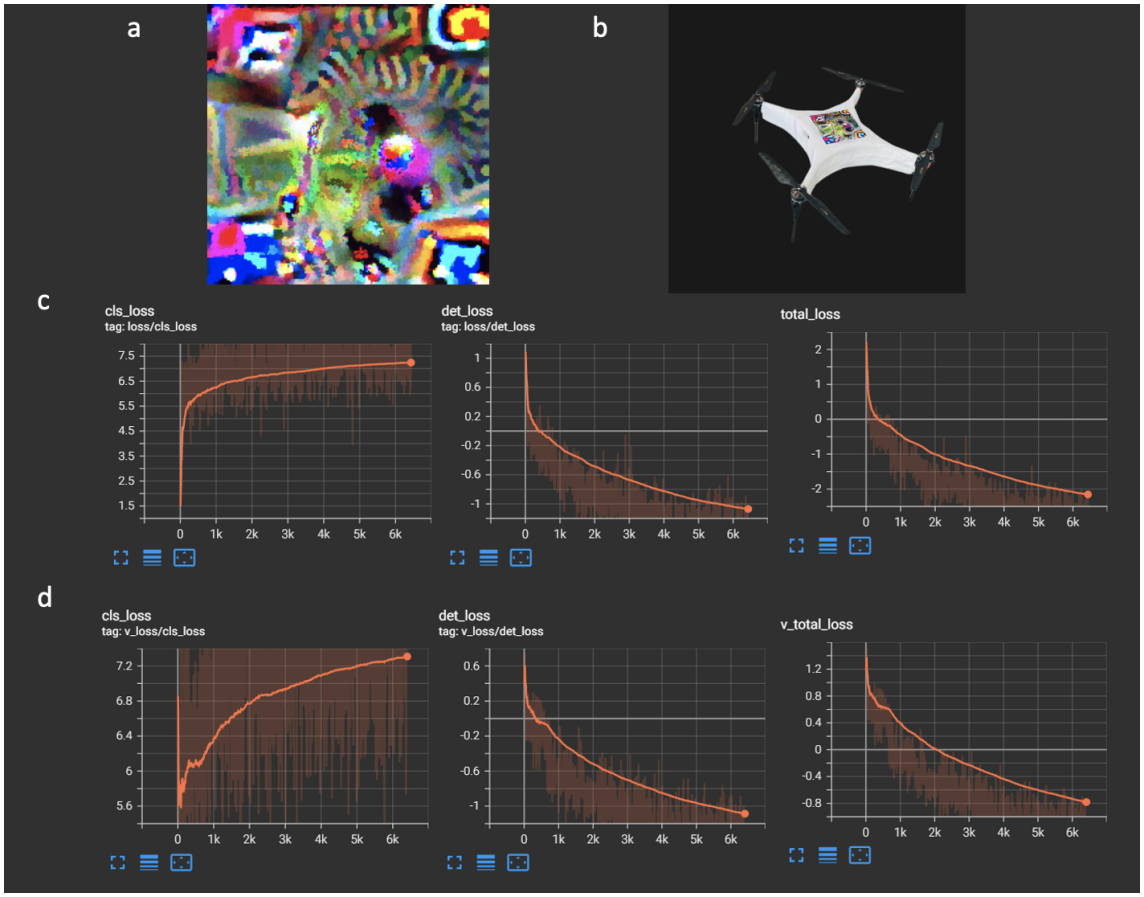


Figure 4: Extra Evasion Patterns Trained with Different Loss Functions - Pattern 3. a) The evasion pattern. b) The evasion pattern in real-world attack model. c) Performance metrics during training on training set. d) Performance metrics during training on validation set. The pattern is trained with DL classification loss and detection loss (with weight 0.8 and 1 respectively) and a reduced learning rate (0.3) on images for only post-canopy drone (neural stealth canopy). During the training, only random rotation augmentation (-45 to 45 degree) is applied, without any perspective augmentation. This evasion pattern fails in real-world attack models (the detection success rate of the drone with this evasion pattern is higher than that without).

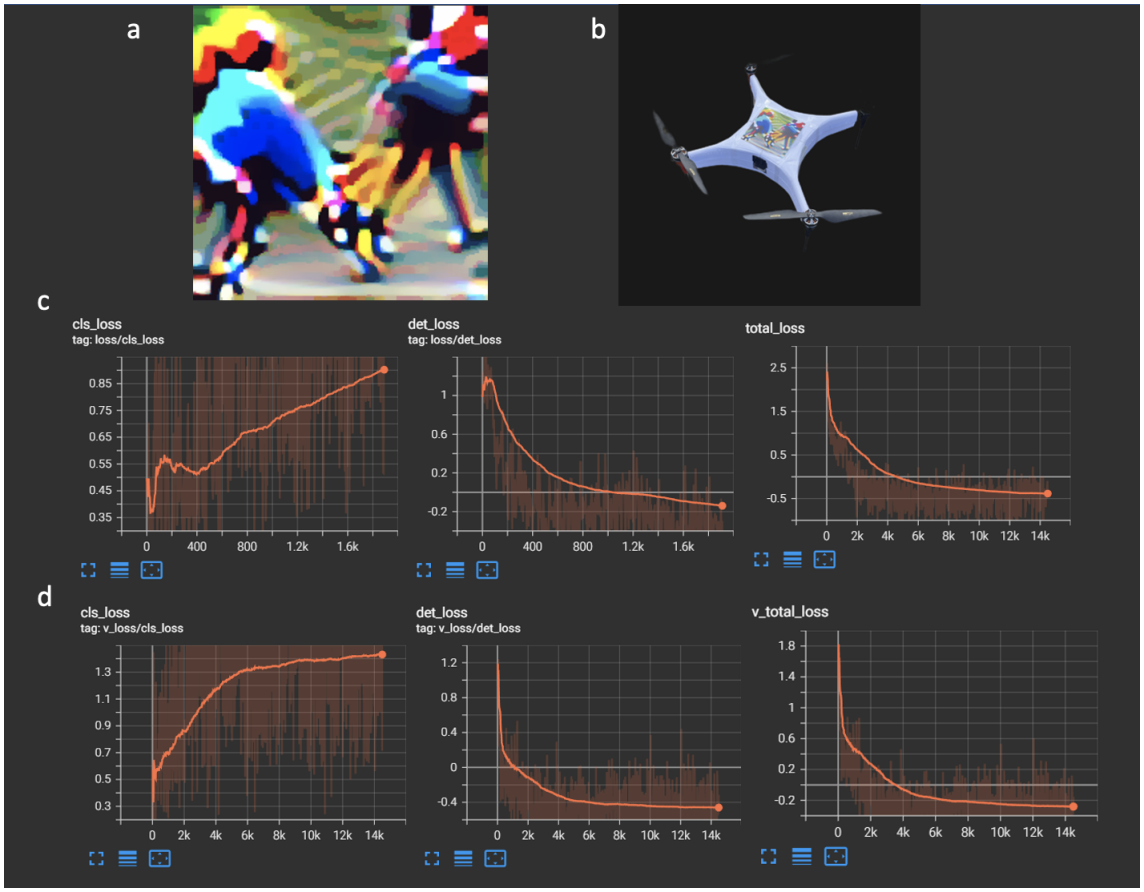


Figure 5: Extra Evasion Patterns Trained with Different Loss Functions - Pattern 4. a) The evasion pattern. b) The evasion pattern in real-world attack model. c) Performance metrics during training on training set. d) Performance metrics during training on validation set. The pattern is trained with only detection loss and a reduced learning rate (0.3) and high-resolution (300,300) on images for only post-canopy drone (neural stealth canopy). During the training, random rotation augmentation (-45 to 45 degree) is applied, with perspective augmentation. This evasion pattern is with slightly effectiveness in real-world attack models (around 6% drop in detection success rate of Yolo V2 object detector).

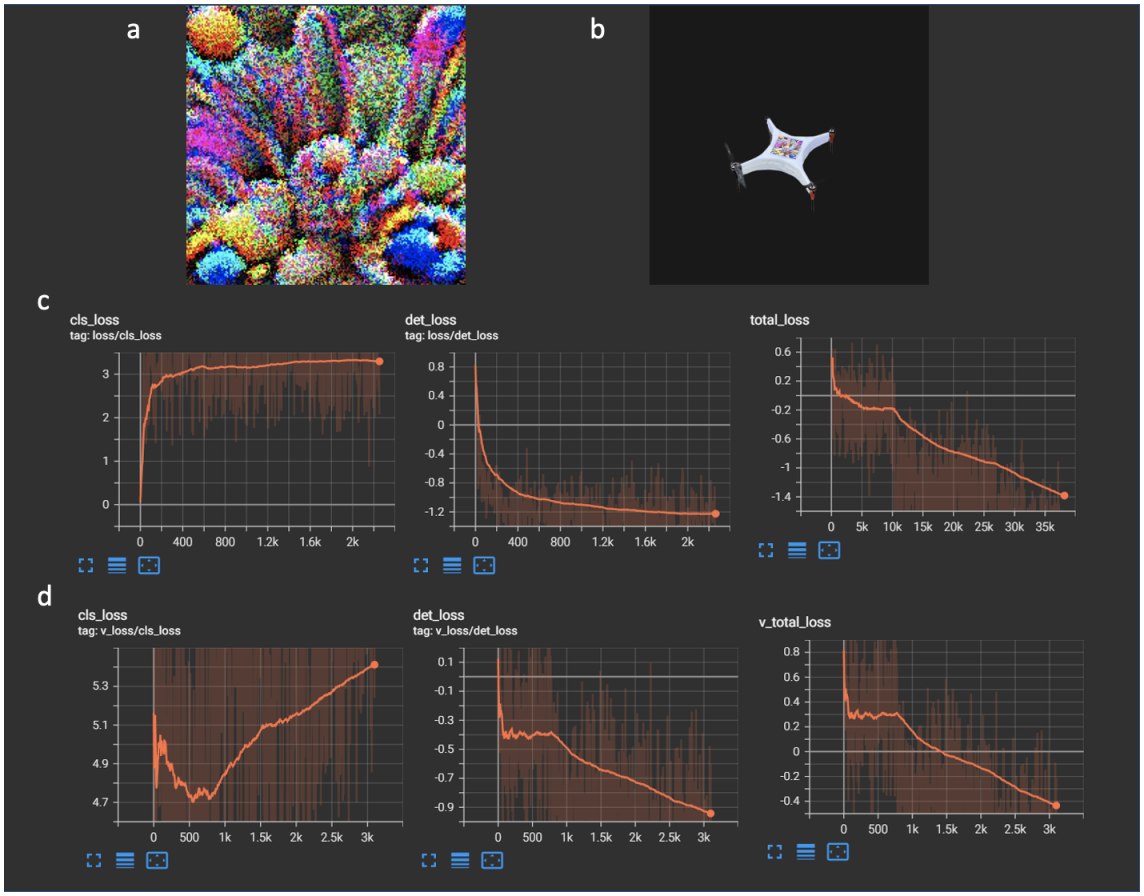


Figure 6: Extra Evasion Patterns Trained with Different Loss Functions - Pattern 5. a) The evasion pattern. b) The evasion pattern in real-world attack model. c) Performance metrics during training on training set. d) Performance metrics during training on validation set. The pattern is trained with DL classification loss and detection loss (with weight 0.8 and 1 respectively) and learning rate (0.5) on images for all drone models. During the training, random rotation augmentation (-45 to 45 degree) is applied, without any perspective augmentation. This evasion pattern fails in real-world attack models (the detection success rate of the drone with this evasion pattern is higher than that without).

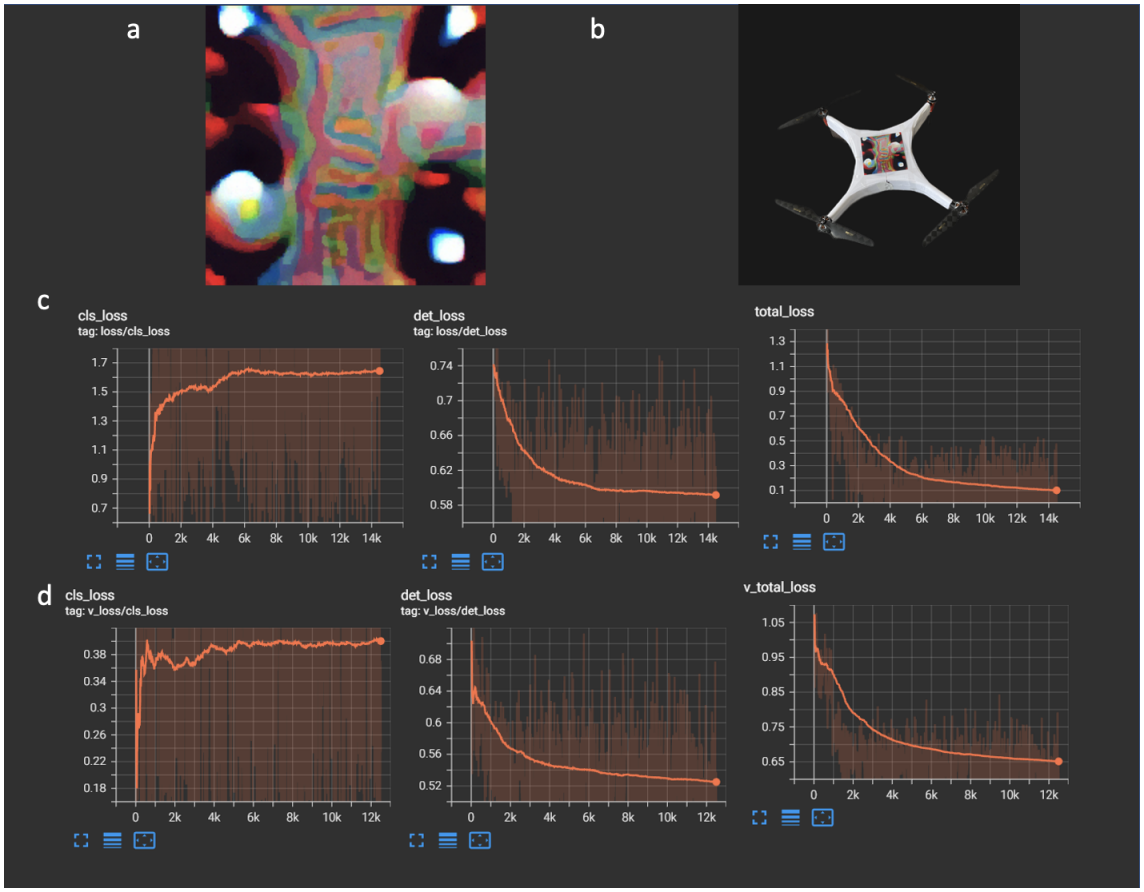


Figure 7: Extra Evasion Patterns Trained with Different Loss Functions - Pattern 6. a) The evasion pattern. b) The evasion pattern in real-world attack model. c) Performance metrics during training on training set. d) Performance metrics during training on validation set. The pattern is trained with both DL classification loss and detection loss (with weight 0.8 and 1 respectively) and a low learning rate (0.03) on images for only the post-canopy drone. During the training, random rotation augmentation (-45 to 45 degree) and perspective augmentation are applied. This evasion pattern is effective in real-world attack models (around 25% drop in detection success rate of Yolo V2 object detector).

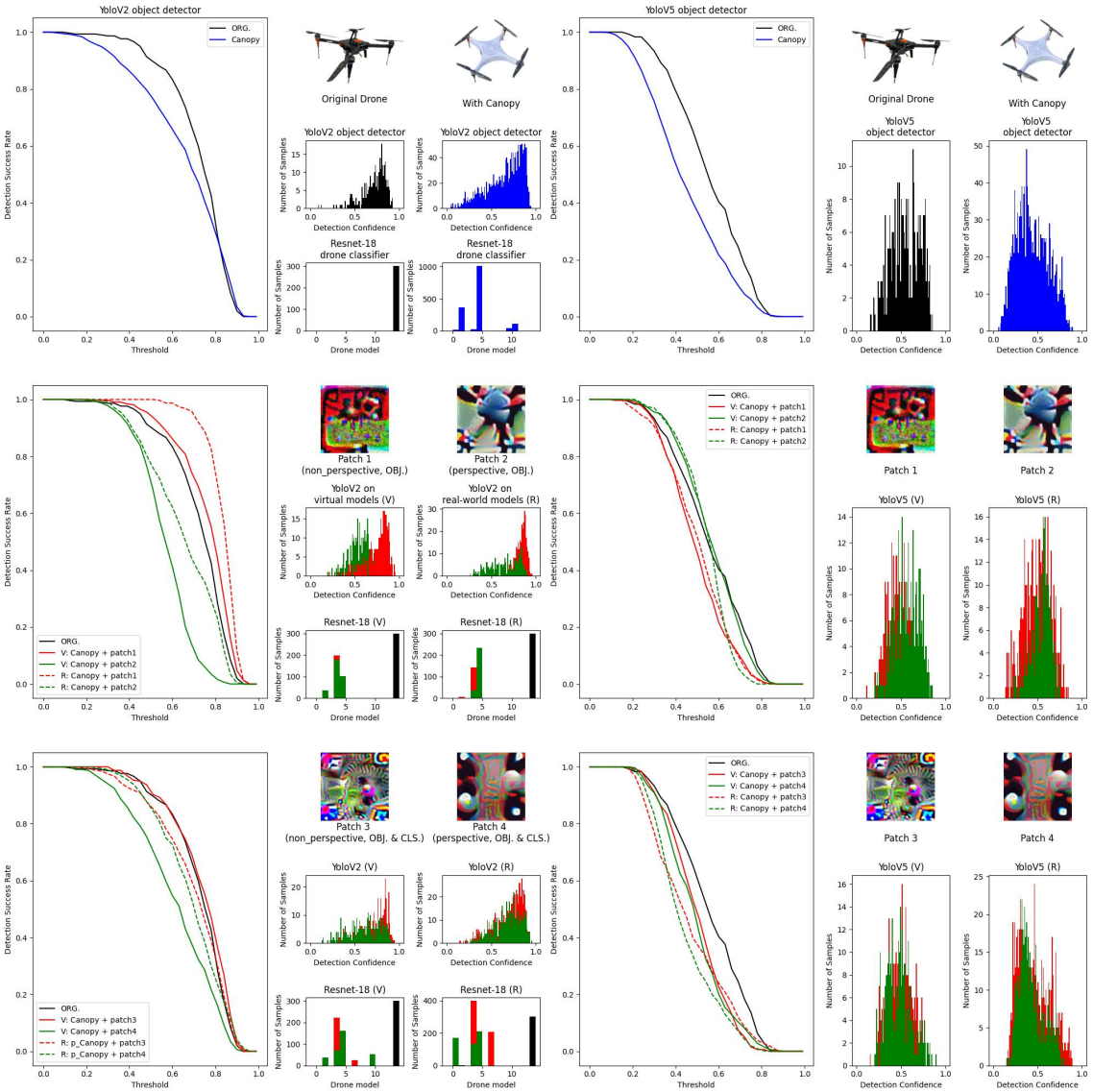


Figure 8: Evaluation Result of Some Extra Evasion Patterns. It is shown the random perspective augmentation significantly increase the invariant of evasion patterns in real-world attack models. And the evasion patterns are with transferability from Yolo V2 to Yolo V5 object detectors.

References

[1] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2014.

[2] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *IEEE conference on computer vision and pattern recognition*, 2017.

[3] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[4] K. V. Andersen, M. H. Frederiksen, M. P. Knudsen, and A. D. Krabbe, "The strategic responses of start-ups to regulatory constraints in the nascent drone market," *Research policy*, vol. 49, no. 10, p. 104055, 2020.

[5] L. Kapustina, N. Izakova, E. Makovkina, and M. Khmelkov, "The global drone market: Main development trends," in *SHS Web of Conferences*, vol. 129. EDP Sciences, 2021, p. 11004.

[6] C. D. M. Size, "Share & trends analysis report by application (filming & photography, inspection & maintenance), by product (fixed-wing, rotary blade hybrid), by end use, and segment forecasts, 2019–2025," *Market Analysis Report. Grand View Research*, 2019.

[7] J. Dominicus, "New generation of counter uas systems to defeat of low slow and small (lss) air threats," in *NATO Science and Technology Organization-MP-MSG-SET-183 Specialists' meeting on drone detectability, pp. KN-2-1-KN-2-20*, 2021.

[8] J. Wang, Y. Liu, and H. Song, "Counter-unmanned aircraft system (s)(c-uas): State of the art, challenges, and future trends," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 3, pp. 4–29, 2021.

[9] M. V. K. Myjak and P. Ranganathan, "Unmanned aerial system (uas) swarm design, flight patterns, communication type, applications, and recommendations," in *2022 IEEE International Conference on Electro Information Technology (eIT)*. IEEE, 2022, pp. 586–594.

[10] P. Schmidt, F. Biessmann, and T. Teubner, "Transparency and trust in artificial intelligence systems," *Journal of Decision Systems*, vol. 29, no. 4, pp. 260–278, 2020.

[11] M. Labbe, "Energy consumption of ai poses environmental problems," *TechTarget*, 2021.

[12] C. Li, S. C. Sun, Z. Wei, A. Tsourdos, and W. Guo, "Scarce data driven deep learning of drones via generalized data distribution space," *Neural Computing and Applications*, 2023. [Online]. Available: https://doi.org/10.1007/s00521-023-08522-z

[13] C. Li, W. Guo, S. C. Sun, S. Al-Rubaye, and A. Tsourdos, "Trustworthy deep learning in 6g-enabled mass autonomy: From concept to quality-of-trust key performance indicators," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 112–121, 2020.

[14] M. Elsayed and M. Erol-Kantarci, "Ai-enabled future wireless networks: Challenges, opportunities, and open issues," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 70–77, Sep. 2019.

[15] K. Burse, R. N. Yadav, and S. Shrivastava, "Channel equalization using neural networks: A review," *IEEE transactions on systems, man, and cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 3, pp. 352–357, 2010.

[16] Z. Du, Y. Deng, W. Guo, A. Nallanathan, and Q. Wu, "Green deep reinforcement learning for radio resource management: Architecture, algorithm compression and challenge," *arXiv preprint arXiv:1910.05054*, 2019.

[17] M. Ylianttila, R. Kantola, A. Gurtov, L. Mucchi, I. Oppermann, Z. Yan, T. H. Nguyen, F. Liu, T. Hewa, M. Liyanage *et al.*, "6g white paper: Research challenges for trust, security and privacy," *arXiv preprint arXiv:2004.11665*, 2020.

[18] C. Li, C. Sun, S. Al-Rubaye, A. Tsourdous, and W. Guo, "Uncertainty propagation in neural network enabled multi-channel optimisation," in *IEEE Vehicular Technology Conference 2020-Spring Recent Results and Workshops*, 2020.

[19] K. Siau and W. Wang, "Building trust in artificial intelligence, machine learning, and robotics," *Cutter Business Technology Journal*, vol. 31, no. 2, pp. 47–53, 2018.

[20] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[21] W. Guo, "Explainable artificial intelligence (xai) for 6g: Improving trust between human and machine," *arXiv preprint arXiv:1911.04542*, 2019.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

[23] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3145–3153.

[24] E. M. Kenny and M. T. Keane, "Twin-systems to explain artificial neural networks using case-based reasoning: comparative tests of feature-weighting methods in ann-cbr twins for xai," in *Twenty-Eighth International Joint Conferences on Artifical Intelligence (IJCAI), Macao, 10-16 August 2019*, 2019, pp. 2708–2715.

[25] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–19.

[26] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant mobile networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 662–675, 2019.

[27] A. Glass, D. L. McGuinness, and M. Wolverton, "Toward establishing trust in adaptive agents," in *Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 2008, pp. 227–236.

[28] S. C. Sun, C. Li, Z. Wei, A. Tsourdos, and W. Guo, "Scalable partial explainability in neural networks via flexible activation functions (student abstract)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 18, 2021, pp. 15 899–15 900.

[29] E. C. White-Paper, "On artificial intelligence - a european approach to excellence and trust," 2020.

[30] 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.

[31] A. M. Alaa and M. van der Schaar, "Demystifying black-box models with symbolic metamodels," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 301–11 311.

[32] W. Guo, "Explainable Artificial Intelligence (XAI) for 6G: Improving Trust between Human and Machine," *IEEE Communications Magazine*, 2020.

[33] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional net-

works: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[34] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3145–3153.

[35] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[36] R. M. Neal, "Priors for infinite networks," in *Bayesian Learning for Neural Networks*. Springer, 1996, pp. 29–53.

[37] C. K. Williams, "Computing with infinite networks," in *Advances in neural information processing systems*, 1997, pp. 295–301.

[38] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Advances in neural information processing systems*, 2017, pp. 6231–6239.

[39] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014.

[40] F. Piazza, A. Uncini, and M. Zenobi, "Artificial neural networks with adaptive polynomial activation function," 1992.

[41] L. Vecci, F. Piazza, and A. Uncini, "Learning and approximation capabilities of adaptive spline activation function neural networks," *Neural Networks*, vol. 11, no. 2, pp. 259–270, 1998.

[42] S. Scardapane, M. Scarpiniti, D. Comminiello, and A. Uncini, "Learning activation functions from data using cubic spline interpolation," in *Italian Workshop on Neural Nets*. Springer, 2017, pp. 73–83.

[43] X. Zhang, Y. Zhao, K. Guo, G. Li, and N. Deng, "An adaptive b-spline neural network and its application in terminal sliding mode control for a mobile satcom antenna inertially stabilized platform," *Sensors*, vol. 17, no. 5, p. 978, 2017.

[44] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini, "Kafnets: Kernel-based non-parametric activation functions for neural networks," *Neural Networks*, vol. 110, pp. 19–32, 2019.

[45] Y. Shen and B. Wang, "A fast learning algorithm of neural network with tunable activation function," *Science in China Series F: Information Sciences*, vol. 47, no. 1, pp. 126–136, 2004.

[46] Y. Shen, B. Wang, F. Chen, and L. Cheng, "A new multi-output neural model with tunable activation function and its applications," *Neural processing letters*, vol. 20, no. 2, pp. 85–104, 2004.

[47] S. Qian, H. Liu, C. Liu, S. Wu, and H. San Wong, "Adaptive activation functions in convolutional neural networks," *Neurocomputing*, vol. 272, pp. 204–212, 2018.

[48] A. S. Douzette, "B-splines in machine learning," Master's thesis, 2017.

[49] S. Urban, M. Basalla, and P. van der Smagt, "Gaussian process neurons learn stochastic activation functions," *arXiv preprint arXiv:1711.11059*, 2017.

[50] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of resnet and googlenet models for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 29–37, 2019.

[51] P. Ballester and R. M. Araujo, "On the performance of googlenet and alexnet applied to sketches," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[52] W. Yu, K. Yang, Y. Bai, T. Xiao, H. Yao, and Y. Rui, "Visualizing and comparing alexnet and vgg using deconvolutional layers," in *Proceedings of the 33 rd International Conference on Machine Learning*, 2016.

[53] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[54] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, no. 1, pp. 93–105, 2003.

[55] F. Gruau, D. Whitley, and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in *Proceedings of the 1st annual conference on genetic programming*, 1996, pp. 81–89.

[56] X. Yao and Y. Liu, "Towards designing artificial neural networks by evolution," *Applied Mathematics and Computation*, vol. 91, no. 1, pp. 83–90, 1998.

[57] B.-T. Zhang and H. Muhlenbein, "Evolving optimal neural networks using genetic algorithms with occam's razor," *Complex systems*, vol. 7, no. 3, pp. 199–220, 1993.

[58] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," in *Doklady Akademii Nauk*, vol. 114, no. 5. Russian Academy of Sciences, 1957, pp. 953–956.

[59] J. Braun and M. Griebel, "On a constructive proof of kolmogorov's superposition theorem," *Constructive approximation*, vol. 30, no. 3, p. 653, 2009.

[60] B. Igelnik and N. Parikh, "Kolmogorov's spline network," *IEEE transactions on neural networks*, vol. 14, no. 4, pp. 725–733, 2003.

[61] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 183–192, 1989.

[62] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[63] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[64] J.-D. Shao, G. Rong, and J. M. Lee, "Learning a data-dependent kernel function for kpca-based nonlinear process monitoring," *Chemical Engineering Research and Design*, vol. 87, no. 11, pp. 1471–1480, 2009.

[65] C. Sun and W. Guo, "Forecasting wireless demand with extreme values using feature embedding in gaussian processes," *arXiv preprint arXiv:1905.06744*, 2019.

[66] A. N. Srivastava, J. Schumann, and B. Fischer, "An ensemble approach to building mercer kernels with prior information," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3. IEEE, 2005, pp. 2352–2359.

[67] Y. Xu, W. Xu, F. Yin, J. Lin, and S. Cui, "High-accuracy wireless traffic prediction: A gp-based machine learning approach," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[68] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[69] A. G. Wilson, "Covariance kernels for fast automatic pattern discovery and extrapolation with gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[70] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[71] H. Jiang, W.-K. Ching, and W. Hou, "On orthogonal feature extraction model with applications in medical prognosis," *Applied Mathematical Modelling*, vol. 40, no. 19-20, pp. 8766–8776, 2016.

[72] F. Nie, S. Xiang, Y. Liu, C. Hou, and C. Zhang, "Orthogonal vs. uncorrelated least squares discriminant analysis for feature extraction," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 485–491, 2012.

[73] C. Li, A. Tsourdos, and W. Guo, "A transistor operations model for deep learning energy consumption scaling law," *IEEE Transactions on Artificial Intelligence*, pp. 1–13, 2022.

[74] D. Amodei, D. Hernandez, G. Sastry, J. Clark, G. Brockman, and I. Sutskever, "Ai and compute," *Heruntergeladen von https://blog. openai. com/aiand-compute*, 2018.

[75] N. Jones, "The information factories," *Nature*, vol. 561, 2018.

[76] W. Guo, S. Zhou, Y. Chen, S. Wang, X. Chu, and Z. Niu, "Simultaneous information and energy flow for iot relay systems with crowd harvesting," *IEEE Communications Magazine*, vol. 54, 2016.

[77] J. Wu, S. Guo, J. Li, and D. Zeng, "Big data meet green challenges: Greening big data," *IEEE Systems Journal*, vol. 10, 2016.

[78] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *Annual Meeting of the ACL*, 2019.

[79] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," *arXiv preprint arXiv:2109.05472*, 2021.

[80] Z. Du, Y. Deng, W. Guo, A. Nallanathan, and Q. Wu, "Green deep reinforcement

learning for radio resource management: Architecture, algorithm compression, and challenges," *IEEE Vehicular Technology Magazine*, vol. 16, no. 1, pp. 29–39, 2021.

[81] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," *arXiv preprint arXiv:2105.04206*, 2021.

[82] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[83] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[84] E. Strickland, "Andrew Ng, AI Minimalist: The Machine-Learning Pioneer Says Small is the New Big," *IEEE Spectrum*, vol. 59, 2022.

[85] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, 2020.

[86] B. Li, P. Chen, H. Liu, W. Guo, X. Cao, J. Du, C. Zhao, and J. Zhang, "Random sketch learning for deep neural networks in edge computing," *Nature Computational Science*, vol. 1, no. 3, pp. 221–228, 2021.

[87] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[88] V. Camus, C. Enz, and M. Verhelst, "Survey of precision-scalable multiply-accumulate units for neural-network processing," in *IEEE Int. Conf. on Artificial Intelligence Circuits and Systems*, 2019.

[89] J. Yu, J. Park, S. Park, M. Kim, S. Lee, D. H. Lee, and J. Choi, "Nn-lut: neural approximation of non-linear operations for efficient transformer inference," in *ACM/IEEE Design Automation Conference*, 2022.

[90] S. Kuninobu, T. Nishiyama, H. Edamatsu, T. Taniguchi, and N. Takagi, "Design of high speed mos multiplier and divider using redundant binary representation," in *IEEE Symposium on Computer Arithmetic*, 1987.

[91] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107281, 2020.

[92] Q. H. Vo, N. L. Le, F. Asim, L.-W. Kim, and C. S. Hong, "A deep learning accelerator based on a streaming architecture for binary neural networks," *IEEE Access*, vol. 10, pp. 21 141–21 159, 2022.

[93] M. Christ, F. de Dinechin, and F. Pétrot, "Low-precision logarithmic arithmetic for neural network accelerators," in *IEEE Int. Conf. on Application-specific Systems, Architectures and Processors*, 2022.

[94] A. Sabbagh Molahosseini, L. Sousa, A. A. Emrani Zarandi, and H. Vandierendonck, "Low-precision floating-point formats: From general-purpose to application-specific," *Approximate Computing*, 2022.

[95] N. Wang, J. Nie, J. Li, K. Wang, and S. Ling, "A compression strategy to accelerate lstm meta-learning on fpga," *ICT Express*, 2022.

[96] S. M. Mishra, A. Tiwari, H. S. Shekhawat, P. Guha, G. Trivedi, P. Jan, and Z. Nemec, "Comparison of floating-point representations for the efficient implementation of machine learning algorithms," in *IEEE International Conference Radioelektronika*, 2022.

[97] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz *et al.*, "Discovering faster

matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[98] P. Pennestrì, Y. Huang, and N. Alachiotis, "A novel approximation scheme for floating-point square root and inverse square root for fpgas," in *International Conference on Modern Circuits and Systems Technologies*, 2022.

[99] N. Campos, E. Edirisinghe, S. Fatima, S. Chesnokov, and A. Lluis, "Fpga implementation of a custom floating-point library," in *SAI Intelligent Systems Conference*, 2023.

[100] Y. LeCun, "1.1 deep learning hardware: Past, present, and future," in *IEEE International Solid-State Circuits Conference*, 2019.

[101] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[102] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: principles, techniques, & tools*. Pearson Education India, 2007.

[103] D. G. A. Godse, *Digital Logic Design*. Technical Publications, 2009.

[104] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[105] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *Asilomar conference on signals, systems, and computers*, 2017.

[106] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of solid-state circuits*, vol. 52, 2016.

[107] A. Shah, C.-Y. Wu, J. Mohan, V. Chidambaram, and P. Krähenbühl, "Memory optimization for deep networks," *arXiv preprint arXiv:2010.14501*, 2020.

[108] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *IEEE conference on computer vision and pattern recognition*, 2017.

[109] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[110] C. F. Rodrigues, G. Riley, and M. Luján, "Synergy: An energy measurement and prediction framework for convolutional neural networks on jetson tx1," in *Int. Conference on Parallel and Distributed Processing Techniques and Applications*, 2018.

[111] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," in *Asian Conference on Machine Learning*. PMLR, 2017, pp. 622–637.

[112] E. García-Martín, N. Lavesson, H. Grahn, E. Casalicchio, and V. Boeva, "How to measure energy consumption in machine learning algorithms," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 243–255.

[113] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[114] R. W. Ahmad, A. Naveed, J. J. Rodrigues, A. Gani, S. A. Madani, J. Shuja, T. Maqsood, and S. Saeed, "Enhancement and assessment of a code-analysis-based energy estimation framework," *IEEE Systems Journal*, vol. 13, no. 1, pp. 1052–1059, 2018.

[115] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," *arXiv preprint arXiv:2109.05472*, 2021.

[116] J. J. Dai, Y. Wang, X. Qiu, D. Ding, Y. Zhang, Y. Wang, X. Jia, C. L. Zhang, Y. Wan, Z. Li *et al.*, "Bigdl: A distributed deep learning framework for big data," in *ACM Symposium on Cloud Computing*, 2019.

[117] M. J. Quinn, "Parallel programming," *TMH CSE*, vol. 526, p. 105, 2003.

[118] A. Lastovetsky and R. R. Manumachu, "New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1119–1133, 2016.

[119] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997.

[120] Z. Abid, H. El-Razouk, and D. A. El-Dib, "Low power multipliers based on new hybrid full adders," *Microelectronics Journal*, vol. 39, 2008.

[121] J. Saini, S. Agarwal, and A. Kansal, "Performance, analysis and comparison of digital adders," in *IEEE Int. Conference on Advances in Computer Engineering and Applications*, 2015.

[122] S. Asif and Y. Kong, "Performance analysis of wallace and radix-4 booth-wallace multipliers," in *IEEE Electronic System Level Synthesis Conference*, 2015.

[123] M. A. Cornea-Hasegan, R. A. Golliver, and P. Markstein, "Correctness proofs outline for newton-raphson based floating-point divide and square root algorithms," in *IEEE Symposium on Computer Arithmetic*, 1999.

[124] H. T. Bui, Y. Wang, and Y. Jiang, "Design and analysis of low-power 10-transistor full adders using novel xor-xnor gates," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 1, pp. 25–30, 2002.

[125] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[126] S.-W. Kim, J. J.-S. Lee, V. Dugar, and J. De Vega, "Intel® power gadget," *Intel Corporation*, vol. 7, 2014.

[127] S. Gochman, A. Mendelson, A. Naveh, and E. Rotem, "Introduction to intel core duo processor architecture." *Intel Technology Journal*, 2006.

[128] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Energies*, vol. 12, no. 11, p. 2204, 2019.

[129] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Thunderx2 performance and energy-efficiency for hpc workloads," *Computation*, vol. 8, no. 1, p. 20, 2020.

[130] N. Coporation, "Nvml api pages-for gpu utilization," 2017.

[131] S. Desrochers, C. Paradis, and V. M. Weaver, "A validation of dram rapl power measurements," in *Proceedings of the Second International Symposium on Memory Systems*, 2016, pp. 455–470.

[132] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *USENIX Annual Technical Conf*, 2011.

[133] J. Miles, "R-squared, adjusted r-squared," *Encyclopedia of statistics in behavioral science*, 2005.

[134] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, 2014.

[135] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[136] D. Page, *A practical introduction to computer architecture*. Springer Science & Business Media, 2009.

[137] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, 2018.

[138] S. Hosseininoorbin, S. Layeghy, B. Kusy, R. Jurdak, and M. Portmann, "Exploring deep neural networks on edge tpu," *arXiv preprint arXiv:2110.08826*, 2021.

[139] B. Vinod, "The covid-19 pandemic and airline cash flow," *Journal of Revenue and Pricing Management*, vol. 19, no. 4, pp. 228–229, 2020.

[140] M. Ball, C. Barnhart, M. Dresner, M. Hansen, K. Neels, A. Odoni, E. Peterson, L. Sherry, A. Trani, and B. Zou, "Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the united states," 2010. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/6234

[141] I. Hatıpoğlu, Ö. Tosun, and N. Tosun, "Flight delay prediction based with machine learning," *LogForum*, vol. 18, no. 1, 2022.

[142] S. Silalahi, T. Ahmad, and H. Studiawan, "Named entity recognition for drone forensic using bert and distilbert," in *2022 International Conference on Data Science and Its Applications (ICoDSA)*. IEEE, 2022, pp. 53–58.

[143] P. Thai, S. Alam, N. Lilith, and B. T. Nguyen, "A computer vision framework using convolutional neural networks for airport-airside surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 137, p. 103590, 2022.

[144] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[145] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2019.

[146] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[147] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[148] P. Jin, L. Lu, Y. Tang, and G. E. Karniadakis, "Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness," *Neural Networks*, vol. 130, pp. 85–99, 2020.

[149] Z. Liu, Y. Xu, C. Qiu, and J. Tan, "A novel support vector regression algorithm incorporated with prior knowledge and error compensation for small datasets," *Neural Computing and Applications*, vol. 31, no. 9, pp. 4849–4864, 2019.

[150] W. Zai El Amri, F. Reinhart, and W. Schenck, "Open set task augmentation facilitates generalization of deep neural networks trained on small data sets," *Neural Computing and Applications*, vol. 34, no. 8, pp. 6067–6083, 2022.

[151] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[152] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[153] M. Huisman, J. N. van Rijn, and A. Plaat, "A survey of deep meta-learning," *Artificial Intelligence Review*, pp. 1–59, 2021.

[154] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[155] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[156] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, 2021.

[157] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *stat*, vol. 1050, p. 1, 2014.

[158] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," *arXiv preprint arXiv:2007.03898*, 2020.

[159] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[160] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[161] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[162] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[163] H. Wang, X. Wu, Z. Huang, and E. P. Xing, "High-frequency component helps explain the generalization of convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8684–8694.

[164] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. E. Hopcroft, "Convergent learning: Do different neural networks learn the same representations?" in *FE@ NIPS*, 2015, pp. 196–212.

[165] I. K. Fodor, "A survey of dimension reduction techniques," Citeseer, Tech. Rep., 2002.

[166] D. Engel, L. Hüttenberger, and B. Hamann, "A survey of dimension reduction methods for high-dimensional data analysis and visualization," in *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering-Proceedings of IRTG 1131 Workshop 2011*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[167] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson, "Extracting insights from the shape of complex data using topology," *Scientific reports*, vol. 3, p. 1236, 2013.

[168] M. G. Bergomi, P. Frosini, D. Giorgi, and N. Quercioli, "Towards a topological–geometrical theory of group equivariant non-expansive operators for data analysis and machine learning," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 423–433, 2019.

[169] H. J. Van Veen, N. Saul, D. Eargle, and S. W. Mangham, "Kepler mapper: A flexible python implementation of the mapper algorithm." *Journal of Open Source Software*, vol. 4, no. 42, p. 1315, 2019.

[170] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network train-

ing by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[171] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[172] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "How to train a gan? tips and tricks to make gans work," *Github. com*, 2016.

[173] G. Singh, F. Mémoli, G. E. Carlsson *et al.*, "Topological methods for the analysis of high dimensional data sets and 3d object recognition." *PBG@ Eurographics*, vol. 2, pp. 091–100, 2007.

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

[175] P. Jin, L. Lu, Y. Tang, and G. E. Karniadakis, "Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness," *Neural Networks*, vol. 130, pp. 85–99, 2020.

[176] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*. PMLR, 2014, pp. 647–655.

[177] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[178] H. Wang, X. Wu, Z. Huang, and E. P. Xing, "High-frequency component helps explain the generalization of convolutional neural networks," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8684–8694.

[179] L. Wasserman, "Topological data analysis," *Annual Review of Statistics and Its Application*, vol. 5, pp. 501–532, 2018.

[180] F. Hensel, M. Moor, and B. Rieck, "A survey of topological machine learning methods," *Frontiers in Artificial Intelligence*, vol. 4, p. 681108, 2021.

[181] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[182] 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.

[183] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020.

[184] A. Sharma, Y. Bian, P. Munz, and A. Narayan, "Adversarial patch attacks and defences in vision-based tasks: A survey," *arXiv preprint arXiv:2206.08304*, 2022.

[185] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.

[186] R. den Hollander, A. Adhikari, I. Tolios, M. van Bekkum, A. Bal, S. Hendriks, M. Kruithof, D. Gross, N. Jansen, G. Perez *et al.*, "Adversarial patch camouflage against aerial detection," in *Artificial Intelligence and Machine Learning in Defense Applications II*, vol. 11543. SPIE, 2020, pp. 77–86.

[187] A. Liu, X. Liu, J. Fan, Y. Ma, A. Zhang, H. Xie, and D. Tao, "Perceptual-sensitive gan for generating adversarial patches," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1028–1035.

[188] H. Yakura, Y. Akimoto, and J. Sakuma, "Generate (non-software) bugs to fool classifiers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1070–1078.

[189] X. Zhou, Z. Pan, Y. Duan, J. Zhang, and S. Wang, "A data independent approach to generate adversarial patches," *Machine Vision and Applications*, vol. 32, no. 3, pp. 1–9, 2021.

[190] A. Zolfi and S. Avidan, "Adversarial mask: Real-world universal adversarial attack on face recognition models."

[191] Z. Chen, B. Li, S. Wu, J. Xu, S. Ding, and W. Zhang, "Shape matters: deformable patch attack," in *European Conference on Computer Vision*. Springer, 2022, pp. 529–548.

[192] M. Lennon, N. Drenkow, and P. Burlina, "Patch attack invariance: How sensitive are patch attacks to 3d pose?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 112–121.

[193] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert, "Gan augmentation: Augmenting training data using generative adversarial networks," *arXiv preprint arXiv:1810.10863*, 2018.

[194] H. J. van Veen, N. Saul, D. Eargle, and S. W. Mangham, "Kepler mapper: A flexible python implementation of the mapper algorithm." *Journal of Open Source Software*, vol. 4, no. 42, p. 1315, 2019. [Online]. Available: https://doi.org/10.21105/joss.01315

[195] P. Zhu, H. Wang, and V. Saligrama, "Zero shot detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 998–1010, 2019.

[196] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin, "Adversarial t-shirt! evading person detectors in a physical world," in *European conference on computer vision*. Springer, 2020, pp. 665–681.

[197] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: an open source differentiable computer vision library for pytorch," in *Winter Conference on Applications of Computer Vision*, 2020. [Online]. Available: https://arxiv.org/pdf/1910.02190.pdf

[198] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 1528–1540.

[199] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[200] A. Shamir, O. Melamed, and O. BenShmuel, "The dimpled manifold model of adversarial examples in machine learning," *arXiv preprint arXiv:2106.10151*, 2021.