Bio-inspired Learning Strategies for Resource-Constrained Cyber-Physical Systems

Thesis submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Suraj Kumar Pandey

Under the supervision of

Prof. Shivashankar B. Nair

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati - 781039 Assam India

July, 2024

Dedicated to

My family

Acknowledgements

I want to express my gratitude to my thesis supervisor, Prof. Shivashankar B. Nair. He allowed me the opportunity to venture into the exciting and intellectually fulfilling research landscape covered under this thesis. I would also like to thank him for the offshoot discussions and brainstorming sessions done during the initial phase of my research which I still cherish and value. I thank him for introducing me to the evolutionary side of Artificial Intelligence which is still untapped in many ways and intrigues me with its fascinating mechanisms.

I am thankful to Prof. Pradip K. Das for chairing my doctoral committee and guiding me with his valuable suggestions. I cherish our discussions and also the extra-curricular engagements that I could be a part of. I am also grateful to Prof. Pinaki Mitra and Prof. Chitralekha Mahanta for presiding over my doctoral committee and scrutinising my research trajectory with their valuable assessments.

I am and will always be grateful to my parents, Shri Ashok Kumar Pandey and Smt. Hemlata Pandey, for nurturing within me the interest and aptitude that led to this research endeavour and allowing me the freedom and due support to focus on this pursuit. Their foresight, planning, patience and teachings not only enabled me to land on this coveted platform but also helped me steer through the challenges faced along this journey. I shall forever be indebted to them for their contribution. I am also thankful to my elder sister, Pallavi, who is always there for me, right from childhood. I am thankful to my wife, Vibha, for being a constant support. Her presence made it easier to steer through the demanding pursuit of this research engagement. I would also like to thank my father-in-law, Shri Shashikant Dixit and my mother-in-law, Smt. Suman Dikshit for supporting me in this endeavour. Lastly, I would like to express my love for my son, Mrityunjay Pandey, who has brought more bliss and peace along with him. I envision an exciting future for him, as viewed through the lens of my research and the scientific advancements in general.

I am thankful to the Heads of the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Guwahati who presided over the course of my PhD, Prof. S V Rao, Prof. Jatindra Kumar Deka and Prof. T. Venkatesh, for allowing me the facilities of the department and for supporting my conference registration. I am also thankful to them and Prof. Pradip K. Das for helping me with the finances to arrange for the setup of one of my experiments which I hope will be useful for the future endeavours of the Robotics lab.

I am thankful to the Ministry of Education of India and IIT Guwahati for supporting my PhD with a scholarship. I am also thankful to the organisers of the Association for the Advancement of Artificial Intelligence for providing me with a generous travel grant and the opportunity to volunteer at their coveted conference. I am also thankful to the Genetic and Evolutionary Computation Conference for offering me the grant to attend their conference, which I unfortunately could not use.

I am thankful to Mr. Hemanta Kumar Nath, Mr. Pranjit Talukdar, Mr. Raktajit Pathak, Mr. Nanu Alan Kachari, Mr. Nava Kumar Boro, and Mr. Bhriguraj Borah for always being there in case of any technical issues or requirements. I would also like to thank Mrs. Gauri Khuttiya Deori, Mr. Gourish Mazumder and Mr. Prabin Bharali for helping with the administrative chores during my PhD. I enjoyed taking the coursework at IIT Guwahati and would thank the faculty

members and the associated teaching assistants for the conduct of the course. I am also thankful to the department's security staff, janitors and our college's canteen staff for their round-the-clock support.

I would also like to express my thanks to Dr. Chandan Karfa who presided as my hostel warden and ensured a peaceful environment for the inmates. A special thanks to the janitors and the mess staff of my hostel for their service.

I would like to thank Dr. Tushar Semwal and Dr. Sonia for helping me in the initial phase of my research and making the lab a lively, collaborative and positive place to work. I would thank Dr. Priya Nair for her warm company and engaging conversations. I would also like to thank my friends at IIT Guwahati, including Yashdeep, Hemraj, Nilotpal, Vanshali, Vinay, Debanjan, Maithili and Nidhi who made my stay at the college pleasant. I am also thankful to my seniors Dr. Swarup, Dr. Abhishek, Dr. Hema and Dr. Deepankar for their company and helpful conversations. I am also thankful to Mr. Karthik Mohan for collaborating as a mentee on an interesting research engagement.

I am thankful to Mr. Sunil Tiwari, my senior from my previous company who persuaded and supported me in joining this research profile. I am also thankful to my friends and seniors from my undergraduate days for staying in touch.

Lastly, I would like to thank the IIT Guwahati administration for maintaining a picturesque and pristine ambience throughout the campus along with the top-notch facilities including the gym, the swimming pool, the lush green grounds and the scenic lakes. Thanks to the student community of the campus as well for keeping the place happening and eventful.

31 July 2024 Suraj Kumar Pandey

Declaration

I certify that

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor(s).
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Apart from the text taken from the work published by me, elaborate sentences used verbatim from other published works have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete and sole responsibility if any complaint arises in future.
- I am fully aware that my thesis supervisor is not in a position to check for any possible instance of plagiarism within this thesis.

31 July 2024 Suraj Kumar Pandey

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

Dr. Shivashankar B. Nair
Professor
Email : sbnair@iitg.ac.in
Phone : +91-361-258-2356

Certificate

This is to certify that this thesis entitled "**Bio-inspired Learning Strategies for Resource-Constrained Cyber-Physical Systems**" submitted by **Suraj Kumar Pandey**, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Assam, India, is a record of the bonafide research work carried out by him under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date: 31 July 2024
Place: Guwahati

Prof. Shivashankar B. Nair
(Thesis Supervisor)

Abstract

Artificial Intelligence (AI) has spanned a wide variety of domains, including computer vision, natural language processing, speech, finance, healthcare, manufacturing, agriculture, etc. This widespread adoption of AI can be attributed majorly to the advancement in both the underlying intelligent algorithms, such as Deep Learning (DL), and also to the increased access to computationally rich processors such as Graphical Processing Units (GPUs) that support the hefty training pipelines of AI models. The combination of DL and GPUs has further catapulted the discovery of newer AI models such as Large Language Models, Diffusion models, etc. While GPUs have immensely fueled the adoption of AI, the dependence of AI models on GPUs proves to be restrictive under resource-constrained scenarios. With the advent of paradigms such as the Cyber-Physical Systems (CPS), the Internet of Things (IoT) and edge computing, the majority of the automation and decision-making processes are shifting from the *Cloud* to the edge devices. Edge devices have limited storage, power and computational resources which makes the deployment of AI operations on them a challenging task. Thus, it is the need of the hour to explore and expand the research landscape at the junction of AI and resource-constrained devices.

TinyML embodies some of the recent efforts made to cater to this challenge. However, most of the contemporary TinyML approaches only facilitate onboard inference, without supporting any onboard training. This involves using a pre-trained model and deploying it on the target edge device(s). Exposing such static pre-trained models to real-world scenarios, such as in the case of CPSs, opens up a Pandora's box of additional issues such as domain shift, concept drift, etc. This mandates the investigation of alternative optimisation routines and learning techniques that allow onboard adaption. Another challenge in this direction is the involvement of real-world data. Edge devices are often deployed in close conjunction with the real world which requires real-world data to be introduced in the training pipeline. However, capturing and annotating real-world data for training is an intricate and costly task. Simulated environments can suffice under such situations as an affordable ancillary source of data.

This thesis focuses on catering to challenges discussed so far across each of its contributions.

The first contribution acts as a kickstarter for the research endeavour of the thesis by realising a CPS for automated surveillance and response. It leverages an IoT-based solution against child trafficking. A majority of the solutions in this direction either rely heavily on the active response of the child or involve substantial cost and monitoring overheads. The existing solutions also involve equipment that is discernible to a trafficker, making the user vulnerable. This contribution proposes an embedded system-based wearable *Smart Patch*. The patch is affordable, concealable and can be attached to many apparel. The patch allows for monitoring a child and also facilitates voluntary and involuntary responses that are pushed towards the guardian. The solution also allows guardians to alter the monitoring parameters. Experimental analyses conducted using the *Smart Patch* highlight its practical feasibility as a wearable embedded system. The use of sensors, an actuator and a microcontroller, embedded within the patch, facilitates the recognition of covert gestures made by the child in distress, which in turn are processed and communicated as per requirement. This involves processing the sensor data and using a pre-determined filter to recognise the target gesture(s). This patch formed the basis for subsequent contributions, that deal with learning-based recognition strategies suited for low-data and low-resource scenarios.

The second contribution forges a synergic relation between connectionist and evolutionary approaches. It leverages the use of Siamese Neural Networks (SNN) that are known to perform well with less data and are capable of extracting valuable information regarding the similarities/dissimilarities within data. The features extracted by SNNs are also employed for multi-class classification. However, the potential of an SNN to transform the similarity space into a classification space needs further exploration to deliver higher accuracy and lower inference time. This contribution proposes a novel multi-class classification approach using SNNs by leveraging concepts from the Immune Network theory. This bio-inspired approach facilitates the extraction of class-specific information in conjunction with the similarity-specific features extracted by the SNN thereby culminating in the enhancement of the classification performance. The experimental analyses across three benchmark datasets highlight the consistent accuracy and inference time improvements achieved by the proposed approach over other SNN-based multi-class classification strategies. Having successfully explored the combination of evolutionary algorithms and Artificial Neural Networks (ANN), this contribution paves the way for leveraging this combination for resource-constrained scenarios in the next contribution.

The third contribution strives to realise embodied intelligence in resource-constrained edge devices. The requirement for enabling intelligence on edge devices has soared with the increase in automation across multiple sectors. However, catering to this requirement is a non-trivial task, owing to the limited computational and memory resources of edge devices. Most of the contemporary techniques deal with this issue by utilising pre-trained models or by outsourcing the training load. Pre-trained models suffer from issues such as concept drift, covariate shift, etc. Further, outsourcing the training load adds issues related to privacy, latency, scalability, etc. Thus, it is crucial to allow onboard training on real-world data. This contribution proposes one such method, the *ChaoticImmuneNet*, that leverages Transfer Learning, Chaos Theory, Artificial Immune Systems and the latent space of a Siamese Neural Network to facilitate onboard learning on resource-constrained edge devices, with limited and noisy data. *ChaoticImmuneNet* relies on the principle of embodied learning wherein the model continuously learns while interacting with its environment as an embodied entity. The proposed method yields improvements in terms of accuracy, time and storage when compared to existing techniques. The method has also been deployed on a real mobile robot inhabiting a prototype warehouse setting, testifying to its efficacy in real-world environments.

While the fourth contribution also focuses on onboard adaptation for edge devices, it addresses a major issue faced in real-world scenarios viz. the sparse availability of real-world data. The availability of a large amount of training data is often crucial for maintaining the performance of Machine Learning (ML) models. Nevertheless, edge devices often deal with real-world data that are tough and costly to collect. Many a time virtual data is thus, used to compensate for this scarcity of data. Domain Adaptation (DA) techniques are used to learn common features between the virtual and the real data, So as to arrive at a generic model. However, the resulting model is too hefty to be deployed on an edge device and often unable to adapt further. This contribution proposes *GADANN*, a TinyML-based Virtual-to-Real knowledge transfer method that allows onboard adaptation. *GADANN* allows onboard ML for edge devices by leveraging Deep Neural Networks, Domain Adaptation, TinyML and Genetic Algorithms. Experimental analyses with *GADANN* have shown that it can act as a lightweight alternative to the existing DA-based and generative AI-based methods for transforming data from the virtual domain to its real counterpart. The efficacy of *GADANN* for real-world settings has been assessed by deploying it on a real robot in a prototype warehouse. The adaptive nature of *GADANN* delivers a substantial performance boost in real-world scenarios when compared with existing DA-based methods that often involve static non-adaptive models.

In the course of experimenting with DA-based methods in the previous contribution, a dire

need was felt for a DA-based dataset for robot vision. Robot vision introduces way more challenges than generic computer vision tasks. The usage of simulated data for DA further exacerbates this issue due to the requirement of both simulated and real-world counterparts within the dataset. The fifth contribution caters to this challenge by introducing *RoboDA*, a multi-domain dataset, having data from both the real world and the simulated domain. A major advantage of the *RoboDA* dataset is the involvement of standard, affordable and open-source apparatus in the data collection process. This allows the users to easily extend the experiments using open-source apparatus. Experimental analysis using both DA-based and generative AI-based methods highlights the complex feature landscape provided by *RoboDA* to evaluate DA methods. *RoboDA* has also been compared with an existing standard computer vision dataset, justifying the role of its simulated counterpart.

The third and fourth contributions under this thesis explored onboard ML for resource-constrained devices such as the Raspberry PI microprocessor. The efficacy observed over the corresponding experiments in each of these contributions pushed this exploration further to find its viability in even smaller computational devices. The sixth and final contribution of this thesis employs the experience gained from the previous contributions to tackle a more challenging task. It deals with the time-series domain to realise a gesture-recognition task, while accommodating data from unseen classes, on an extremely resource-constrained edge device. Unlike the previous contributions, in addition to onboard learning, this contribution facilitates class incremental learning that allows the addition of the knowledge of new classes, after deploying the ML model onboard the target device. This allows the proposed method to deal with non-stationary data, alleviating the issues of catastrophic forgetting when augmenting the model with knowledge of the new class. The method utilises Latent Replays from a Deep Learning model, TinyML and Genetic Algorithms to create a multi-fitness landscape that can be traversed in a computationally efficient way to accommodate new class data, while retaining the previous knowledge. The proposed method has been tested on an extremely resource-constrained microcontroller (ESP32), which is by far the smallest among all the devices used in this thesis. This contribution also acts as a closure to the thesis by cycling back to the first contribution that dealt with gesture recognition, this time facilitating the task with a more robust ML-based recognition method onboard the target device.

Contents

List of Figures

List of Algorithms

List of Tables

List of Symbols

Symbols	Description
Ab_1, Ab_2, Ab_3	Antibodies
Ag	Antigen
$\mathcal{L}$	Contrastive loss
τ	Distance margin
D	Euclidean distance
ST_j^i	Stimulation for the sample j of class i
Aff_{jk}^i	Affinity of sample k for sample j in class i
SU_j^i	Suppression received by sample j of class i
S_{GS}	Set of gold standards
r	growth rate of logistic function
E_{GS}	Embedding of GS
D_{tr}	Training dataset
$E_{S_{GS}}$	Set of embeddings obtained from all the GSs in S_{GS}
$E_{D_{tr}}$	Set of embeddings for all training samples in the dataset D_{tr}
E_{tr}	Embedding of a training sample
F	Filter
Par_{GS}	*Paratope*
Epi_{tr}	*Epitope*
$\bar{F}$	Average vector of all the filters in F

Rw	Reward
F_{bottom}	Least performing filters
F_{top}	Best performing filters
»	Transfer of weights from a source network (left) to the target network (right)
	Using TL
$_{(D_T)}$	Training the target SNN onboard the robot
$_{(D_T)}$	Creating the GSs
D_{GS^i}	distance of GS^i with all other sample
SNN_S	Source Siamese Neural Network
D_S	Source dataset
SNN_T	Target Siamese Neural Network
D_T	Target dataset
ED_r	Real-world edge device
Env_r	Real-world environment
ED_v	Virtual edge device
Env_v	Virtual environment
D_r	Real-world data
D_v	Virtual data
M_{DA}	Domain adaptation model
B_v	M_{DA} branch for virtual input
B_r	M_{DA} branch for real-world input
E_v	Embedding of virtual data
E_v	Embedding of virtual data
E_r	Embedding of real-world data

B_c	Classification block
Ex	Extraction layer
M_*	Model obtained by truncating/appending to components of M_{DA}
M_{lite}	Compressed model for TinyML usage
O_{Ex}	Latent representation output of M_{lite}
O_c	Classification output of M_{lite}
F_{Ex}	Filtered representation obtained for data sample X by using O_{Ex}
$f_{parents}$	Population of parent filters
$f_{children}$	Population of child filters
f_{child}	A child filter
f_{best}	Best performing filter
P_i	Pallet
M_i	Microcontroller
U_i	Ultrasonic sensor
N_s	Source network
D_s	Source dataset
L_e	Extraction layer
L_c	Classification layer
N_t	Target network
N_{lite}	Compressed model for TinyML usage
old_i	A class in D_s
Exm_{old_i}	Exemplar for class i of old_i
Exm_{old}	Exemplars of all classes of D_s
D_{new}	Training data for new class to be learnt incrementally

Exm	Exemplars of all the classes existing in D_s and D_{new}
num_aug	Number of samples used for augmentation
φ	Noise vector used for augmenting Exm
Exm_{aug}	Augmented dataset based on Exm
Φ	Population of candidate solutions
S_{new}	A sample from the new dataset D_{new}
$E_{S_{new}}$	Embedding for S_{new}
E_{masked}	Masked embedding for $E_{S_{new}}$ using a filter from Φ
Exm_{masked_i}	Masked embedding for the exemplar of class i in Exm
$Disc_{Exm_i}$	Discrimination factor for exemplar of class i
F_{Φ_i}	Final fitness for filter Φ_i
Φ_{parent}	Population of parent filters
$\Phi_{interim}$	Merged population of parent and child, prior to the mutation operation
Φ_{best}	Filter having the best fitness

List of Abbreviations

Terms	Abbreviations
AI	Artificial Intelligence
DL	Deep Learning
GPU	Graphical Processing Unit
CPS	Cyber-Physical Systems
IoT	Internet of Things
SNN	Siamese Neural Network
ANN	Artificial Neural Network
ML	Machine Learning
DA	Domain Adaptation
GA	Genetic Algorithm
AIS	Artificial Immune System
ED	Edge Device
DNN	Deep Neural Network
EAI	Embodied Artificial Intelligence
CIL	Class Incremental Learning
BLE	Bluetooth Low Energ
WLAN	Wireless LAN
RSSI	Received Signal Strength Indicator
SSID	Service Set Identifier

BSSID	Basic Service Set Identifier
IER	Involuntary Emergency Response
VER	Voluntary Emergency Response
MQTT	Message Queuing Telemetry Transpo
NodeMCU	Node MicroController Unit
SoC	System-on-a-Chip
GS	Gold Standard
IN	Immune Network
MLP	Multi-Layer Perceptron
KNN	K-Nearest Neighbour
BIS	Biological Immune System
FMNIST	Fashion-MNIST
KMNIST	Kuzushiji-MNIST
TL	Transfer Learning
FT	Fine Tuning
CNN	Convolutional Neural Network
t-SNE	t-distributed Stochastic Neighbor Embedding
V2R	Virtual-to-Real
DR	Domain Randomisation
GAN	Generative Adversarial Networks
StoR	SyntheticToReal
ViPeD	Virtual Pedestrian Dataset
VisDA	Visual Domain Adaptation
IL	Incremental Learning

LR	Latent Replays
SVM	Support Vector Machines
IMU	Inertial Measurement Unit
ESP32	ESP32-Wroom-32

Glossary of Terms

Terms	Description
Cyber-Physical Systems	An engineered system comprising computational and physical parts working together to solve complex tasks
Internet of Things	A network of heterogeneous interconnected components able to share data with each other
Siamese Neural Network	A type of neural network comprising multiple branches that converge to create a similarity space
Domain Adaptation	A technique of learning a model from a source data distribution and applying that model to a different (but related) target data distribution
Genetic Algorithm	A population-based heuristic search algorithm inspired by the natural phenomenon of natural selection
Artificial Immune System	A computational system inspired by the working mechanism and principle of the biological immune system
Antibody	Large, Y-shaped proteins belonging to the immunoglobulin superfamily which are used by the immune system to identify and neutralize antigens such as bacteria and viruses
Antigen	A substance that causes the immune system to produce antibodies against it. It may be a substance from the environment, such as chemicals, bacteria, viruses, or pollen.
Immune Network	A dynamic biological network created by the interactions between antibodies for recognising antigens
Crossover	A method of generating new antibody population (offspring) from the existing population (parents)
Mutation	A method involving alteration of certain characteristics of the offspring
Affinity maturation	A process used by the immune system to produce antibodies with a greater affinity for an antigen during an immune response by leveraging somatic hypermutation and selection
Edge Device	An endpoint on the network, the interface between the data centre and the real world
Embodied Artificial Intelligence	An AI paradigm in which intelligent agents interact with the environment through a physical embodiment, such as a robot
Latent Replays	A representation of the input data space created by storing activation from an intermediate layer
Hypermutation	A biological process that diversifies antibody genes and helps the immune system adapt to new foreign elements
Gold Standard	Representative sample(s) of a class in a dataset

Chaos	A deterministic behaviour exhibited by certain dynamical systems having properties such as sensitivity to initial conditions, ergodicity, etc.

1

Introduction

Cyber-Physical Systems (CPS) are engineered systems that comprise a set of computational and physical counterparts, working together to achieve complex tasks [1]. Applications of CPSs span across multiple sectors including industries, smart grid, transportation, healthcare, agriculture, etc. [2]. CPSs entail a large volume of data and often have resource constraints [3].

While cloud resources have been utilised to support computation for crunching the data generated by CPSs, they add to the delay and thus hamper the real-time performance of CPSs [4, 5]. This has encouraged the usage of edge computing in conjunction with CPSs [4, 5], facilitating analysis and inference on components closer to the data source including Edge Devices (EDs). EDs are devices closest to the user, such as mobile phones, CPSs, wearables, etc. These devices are limited in terms of memory, computational and power resources [6]. EDs play a crucial role in enabling intelligent decision-making for automation tasks performed by CPSs. Intelligent EDs include vision-guided robots, industrial PCs, digital cockpit systems, patient monitoring devices, Internet of Things (IoT) devices, etc. [7]. Due to their real-world deployments, CPSs often involve real-world data. Dealing with real-world data and ensuring consistent performance, usable in the real world, requires the usage of robust Machine Learning (ML) algorithms. However, leveraging ML algorithms on resource-constrained EDs is a non-trivial task [8]. TinyML has generated immense impetus to facilitate the amalgamation of EDs and ML techniques [8]. Despite ongoing efforts, there exist several open issues that need to be tackled to efficiently facilitate ML techniques for EDs.

1.0.1 Motivation

The Artificial Intelligence (AI) market is projected to soar to around 740 billion U.S. dollars between the year 2023 and 2030, with an annual growth rate of 17.3% [9]. A parallel rise is being observed in the adoption of big data-related applications, backed by the growing expanse of IoTs, that leverage AI techniques to ingest the data and deliver valuable insights [10]. However, the usage of GPUs, generally used for hosting these applications, is following a declining trend. Application-specific integrated circuits are predicted to gain a substantial chunk in training AI applications by the year 2025, forcing the usage GPUs to plummet from 97% to 40% [11]. In fact, the global edge data center market size is predicted to double up in the year 2024 in comparison to the year 2020 with IoT being one of the underlying factors [11]. These trends highlight the role of AI as a driving force of market growth and the impending expanse of EDs and edge computing for driving AI applications in return. Leveraging both the trends, the introduction of ML for EDs improves user experience via latency reduction, near-real-time data analyses, privacy, and security. The global

edge ML market is expected to flourish with a compounded annual growth rate of 26% from 2022 to 2027 [12]. Edge ML provides the following advantages over traditional *Cloud* based techniques [13, 14]:

- Low latency: Edge ML involves the deployment of AI services close to the data source and the target users. This eliminates the need to send and receive data and inference to and from the *Cloud*, significantly reducing the operational latency. Low latency is vital for a wide variety of applications, including industrial robots and automated guided vehicles [14].
- Privacy Preservation: Due to less/no communication of data with external entities, the data used for AI training or inference stays closer to the ED, mostly within the premises of the user. This is crucial in privacy-sensitive applications such as healthcare, proprietary industrial applications, etc.
- Reliability: Involvement of multiple EDs as a replacement to a central *Cloud* service reduces the chances of overall failure by providing multiple functional nodes.
- Power-Efficiency: *Cloud* based operations require a substantial amount of power and also have a significant carbon footprint. The usage of hefty computing devices, such as GPUs, is the main reason for such high power loads. The usage of EDs on the other hand drastically reduces the power requirements.
- Cost Effectiveness: Edge ML-based applications require less bandwidth and computational resources and thus provide a more affordable alternative to *Cloud* based AI operations.

Despite the numerous advantages of edge ML systems, *Cloud* based systems can still be leveraged, in conjunction with EDs to realise a more robust AI ecosystem. Such an arrangement can involve frequent usage of EDs, within a CPS, for enabling real-time automation while leveraging the *Cloud* for occasional heavy-duty computational loads. A rudimentary way to achieve this is by quantising large models, trained on *Cloud* GPUs, into small-sized models suitable for running on low-resource devices under a CPS setup. In fact, some of the state-of-the-art AI architectures such as the large language models are also being optimised for operation on EDs [15].

However, training large models with a humongous amount of data is not the only solution that CPSs seek. CPSs involve noisy and dynamic data distributions, generated and consumed by EDs. This not only demands algorithms that act as good function approximators but also requires such algorithms to cover the optimisation space quickly and effectively. Bio-inspired algorithms are known to meet these demands by providing a wide range of optimisation strategies [16]. This establishes the need to delve deeper into the realm of bio-inspired algorithms catering to CPSs. However, efficient traversal of the feature space of the data distribution remains a challenge that needs to be tackled in the case of bio-inspired algorithms. With the resource constraint of the EDs used in the CPSs, the traversal needs to be quick while also covering the maximal feature space. Chaotic systems exhibit this property and are known to be able to evade issues such as local minima, early convergence and slow optimisation rate while traversing a feature space for optimisation tasks [17, 18, 19, 20]. Exploring the amalgamation of chaotic systems with bio-inspired algorithms is thus, an area that needs attention.

The challenges involved in realizing the aforementioned objectives and have been addressed in this thesis have enumerated below:

1. *Data Scarcity*: ML techniques often involve large amounts of data to train the underlying model. However, collecting and annotating real-world data from EDs is a costly affair. Delivering reliable performance with low availability of data is a challenge that needs to be tackled.

2. *Reality Gap*: Alternate sources of data, for instance, simulators, are often utilised to compensate for the scarcity or low utility of real-world data. However, the difference in the real and simulated data affects the real-world performance of the underlying model, often cited as the *reality gap* [21].

3. *Dataset Shift*: Dataset shift corresponds to the change observed in the training and test data sets upon deployment of a model in the test environment[22]. This difference causes the model performance to fall during the testing phase. This is a commonly observed issue in ML applications involving real-world data such as in CPSs.

4. *Onboard Training*: Off-the-shelf pre-trained models fail to perform well when deployed in the real world [23]. This necessitates the need for onboard training of the models with real-world data, obtained via the EDs. However, given the resource constraints of EDs, training ML models onboard the EDs is a challenging task due to the large memory and computational footprints of most of the existing ML models.

5. *Catastrophic Forgetting*: While onboard training is crucial for maintaining the model performance, acquiring new knowledge can negatively impact the pre-existing knowledge, a phenomenon often referred to as *catastrophic forgetting* [24]. For EDs with constrained storage resources, this issue is more prominent and needs to be tackled to retain the existing knowledge, while adapting to the new one.

6. *Resource Constraints*: EDs include endpoints used as wearables, components of the IoT and CPS, embedded and autonomous systems, robots, etc. These devices have limited storage and computational capabilities, and power resources. Performing compute-intensive ML techniques onboard the EDs can thus become a non-trivial task [6]. While most of the existing efforts of using ML with EDs involve onboard inference, *training* ML models onboard the EDs push this challenging task even further.

7. *Robot Vision*: Unlike in the case of conventional computer vision, processing robot vision holds numerous challenges. It requires the use of the vision model as an active and embodied [25] entity in a robot [25]. The performance of the underlying model is influenced by the actions of the robot as well as the changes in its environment [26]. With the increased involvement of robots in the CPS space, robot vision is an area that requires dedicated attention.

The aforementioned challenges create a multifarious landscape, requiring solutions from multiple domains. Consequently, the work conducted under this thesis caters to these challenges by exploring concepts from multiple domains, including connectionist algorithms such as Artificial Neural Networks (ANN), bio-inspired metaheuristic algorithms such as Genetic Algorithms (GA) and Artificial Immune Systems (AIS), Chaotic Dynamical Systems, TinyML, Generative Artificial Intelligence, Domain Adaptation (DA), Transfer Learning, etc. The initial focus of the thesis spans the implementation of a CPS for proactive surveillance and response along with the exploration of strategies for synergistic collaboration between connectionist and bio-inspired algorithms. The thesis then explores the synergistic relation formed to leverage ML techniques for resource-constrained EDs, with onboard learning to ensure real-world performance. This exploration primarily focuses on computer vision and the robotics domain. Finally, the thesis knits together the research conducted so far to facilitate onboard ML on extremely resource-constrained EDs for the time-series domain. A central theme of the thesis has been to ensure continual learning by exploring strategies for accommodating not only a shift in the data but altogether new data classes as well. The thesis thus, deals with the nascent research landscape lying at the junction of CPSs and ML and extends it further to allow robust and resource-efficient mechanisms that can be deployed in the real world.

1.1 Background

1.1.1 Cyber-Physical Systems, Internet of Things and Artificial Intelligence

CPSs have been leveraged across a rich variety of sectors including agriculture, aeronautics, design, infrastructure, energy, environmental quality, healthcare, manufacturing, transportation, etc. With the vast availability of data, CPSs support and offer more autonomy [1]. The pursuit of automation within a CPS has given rise to the usage of efficient techniques for data acquisition and ingestion, with IoT and Artificial Intelligence (AI) being at the forefront [27, 1]. An IoT involves a collection of components, often arranged across multiple layers of functionality, to allow data flow across the underlying system [27]. Considering the expanse of the CPSs across sectors, the integration of IoTs and AI with CPSs spawns multiple challenges.

A cornerstone of CPSs is the usage of EDs, comprising a variety of devices such as robots, microprocessors, microcontrollers, etc. In fact, the role of EDs is becoming more prominent due to the decline in the utility of the *Cloud* for real-time systems such as agriculture, livestock management, smart cities, healthcare, logistics, retail etc. [4, 8]. There is a growing demand for bringing contextual understanding and decision-making closer to the EDs [28]. Consequently, there is a soar in efforts towards realising ML techniques with EDs [29]. In fact, the large language models, one of the cutting-edge AI technologies, that are still constantly evolving, are being pushed into EDs [30]. Bringing AI to EDs is thus, not a matter of choice, but of time, as is evident from the survey conducted by *Gartner*. It suggests that more than 55 per cent of data analyses shall be conducted using Deep Neural Networks over EDs by the year 2025 [31].

However, enabling ML on resource-constrained EDs is a non-trivial task owing to its low memory, power and computational resources [8]. In recent years, TinyML, an umbrella term for techniques targeted towards enabling ML on resource-constrained devices, has attracted a substantial amount of research interest [8]. However, most of these techniques aim at the deployment of pre-trained ML models on EDs for inference. Given the dynamic and noisy nature of real-world deployments, devising adaptive algorithms that can accommodate dynamic changes in data, has become a crucial area of research. The complexity of the real-world data further demands the algorithms to be rich enough to provide smooth approximations for the data distributions. Bio-inspired algorithms are known to provide good approximations to complex problems with limited availability of data and computation [16] thereby providing the motivation to explore, blend and devise bio-inspired algorithms to address the aforementioned challenges.

1.1.2 Bio-Inspired Algorithms

The ongoing digital era has seen an overwhelming abundance of data and a growing demand for data-driven decision-making solutions. However, given the multi-dimensional vast hyperspace rendered by such large datasets, the resulting solutions become intractable, often falling into the NP-hard category. Chasing such solutions, for real-time applications, as in CPSs, proves to be expensive and inefficient. This calls for exploring optimal solutions that can cater to the complexity of the data and the associated applications.

Owing to their rich meta-heuristic optimisation capabilities, bio-inspired algorithms, are considered to be one of the best choices of algorithms to devise intelligent methods for dealing with complex problems involving dynamic and imperfect data with limited computation capabilities [16]. Bio-inspired algorithms form an umbrella term for computational methods orchestrated by drawing inspiration from nature. Their robust nature often facilitates their use across a wide variety of machine learning tasks [32]. A rich variety of bio-inspired algorithms have been proposed [16]. These

include the Artificial Neural Network (ANN), the Genetic Algorithm (GA), the Ant Colony Optimisation (ACO) algorithm, the Particle Swarm Optimisation (PSO) algorithm, the Artificial Immune Systems (AIS), etc. [16]. Each of these algorithms is suited for a range of specific applications. For instance, ANNs are very good function approximators and given a large amount of training data, can mimic a data distribution well. On the other hand, GAs are more suited for applications where the definition of the optimisation space is complex. They require implicit parameters and *fitness* functions to evaluate and drive the model performance and training, respectively. Given the time-tested mechanisms from which the bio-inspired algorithms draw their conceptual basis, the resulting computational models deliver robust and efficient mechanisms to deal with real-world noisy data. Bio-inspired algorithms were chosen as the basis for devising the solutions proposed in this thesis, mainly because of the complex and noisy data associated with CPSs.

1.2 Problem Statement

The increasing reliance on GPUs and computationally heavy model configurations for realising modern AI solutions, poses significant deployment and maintenance challenges, particularly for resource-constrained environments. The edge devices, integral to Cyber-Physical Systems (CPS) often lack the computational power, storage capacity, and energy resources necessary to support contemporary AI operations. Consequently, there is a pressing need to explore solutions that enable effective AI deployment under resource-constrained settings. Real-world deployment of AI solutions on resource-constrained devices poses the following key challenges

1. Limited Onboard Learning Capabilities
2. Non-stationary real-world data
3. Data Scarcity and Annotation Difficulties
4. Resource Constraints

This research aims to address these challenges by proposing novel methodologies that facilitate the deployment of AI models onboard resource-constrained devices along with the provision to maintain the onboard performance of models deployed in the real world. Through a series of contributions focused on these areas, this thesis seeks to push the boundaries of AI deployment in edge computing scenarios, ultimately contributing to more robust and adaptable AI systems for real-world applications.

1.3 Contribution of the Thesis

The research works portrayed in this thesis constitute a flow that ensures that subsequent contributions leverage the ones described earlier. These contributions are discussed as below.

1.3.1 Devising a Proactive Surveillance and Response solution using a Cyber-Physical System

CPSs play a crucial role in supporting the upcoming concept of Smart Cities that use intelligent computing strategies to collect and process data from multiple sources and facilitate resource optimisation, monitoring, surveillance, etc. [33]. Most of the existing CCTV-based systems act as

passive surveillance systems that rely heavily on manual intervention and response. With the soaring amount of smart devices such as cameras and wearables, the corresponding data is bound to increase exponentially, making manual analysis intractable. This necessitates the role of automated monitoring systems that can sense and respond to emergencies. This contribution provides an IoT based CPS as a solution for automated monitoring and response against child trafficking. Child trafficking is a grave and rising issue that demands a specialised solution due to the clandestine nature of the problem and the lack of the required responsiveness and awareness of the victim (child). A major challenge in achieving this solution is to provide a non-intrusive setup that can, in real-time, sense an exigent event and provide an automated mitigatory response. The solution also needs to have a low cost and a low form factor to increase its usability. The proposed solution, which constitutes a CPS, comprises a *wearable patch* that can be worn by a child to monitor his/her location and activities. Based on any exigent event, the patch generates responses either voluntarily or involuntarily to proactively send notifications to the guardian. The CPS setup leverages IoT techniques to facilitate two-way communication where information can flow to and fro the child and the guardian. The usage of a low-cost microcontroller, sensors and an actuator, allows real-time sensing of the child's location and a pre-determined gesture to instigate a voluntary response in case of an emergency. The proposed solution stands apart from its contemporaries due to its low cost, low form factor, covert setup, ease of usage and enhanced control over the response being generated. The experiments conducted in the thesis testify to the practical usability of the proposed wearable and its efficacy in sensing and responding to the desired incidents. This contribution also acted as a motivation to steer the thesis towards the exploration of intelligent mechanisms that can enhance the performance of EDs for real-time and real-world applications.

1.3.2 Forging a Synergy between Connectionist and Evolutionary Machine Learning algorithms

CPSs need to deal with complex real-world data, rife with noise. Modelling such a data distribution requires effective algorithms that can approximate the underlying distribution function. The expressive power of Deep Neural Networks (DNN) has been investigated and exposed in great detail [34] making them an ideal candidate for tackling noisy data. Among the several types of DNNs, Siamese Neural Network (SNN) is a type of neural network that quantifies the similarities between its inputs [35]. SNNs have low data requirements due to which special attention has been given to them in this thesis owing to the scarce availability of real-world data. This contribution specifically focuses on a multi-class classification problem. While SNNs have been used across multiple applications, their usage for classification has not been explored exhaustively and is often realised at the cost of losing on the inter-class information, scalability, inference time and classification performance. Taking inspiration from the coordinated response of the Biological Immune System, (BIS), this contribution leverages concepts from the AIS [36] to enhance the multi-class classification performance of an SNN. The ablative analysis conducted in this contribution validates the architecture of the proposed method. Its comparison with the existing methods across multiple standard datasets, including the MNIST [37], the Fashion-MNIST (FMNIST) [38] and the Kuzushiji-MNIST (KMNIST) datasets [39], highlights the accuracy and inference time gain achieved. The experimental results of this contribution testify to the synergistic relationship created between neural networks and bio-inspired approaches that have been explored in subsequent contributions.

1.3.3 Embodied Learning in Resource-Constrained Devices

While noisy and sparse real-world data is one challenging aspect of CPSs, the constraint on computational, power and storage resources is another. Devising algorithms to deal with noisy data solves only a part of the challenges faced in realizing resource-constrained CPSs. The increasing expanse of EDs, which are an integral part of the CPSs, has drawn the attention of the research community to embed intelligent mechanisms in them. While pre-trained DNN models have been used onboard EDs, issues like concept drift and domain shift, deter the performance of such static models when faced with real-world noisy data. This necessitates the usage of Embodied Artificial Intelligence (EAI) wherein the intelligence model adapts to the environment that it is placed in, via continual interactions with the environment, thereby adapting to the environmental variations. Due to the resource constraint of EDs, realising DNN-based EAI, onboard, is a tough task. This contribution tackles the aforementioned challenges by using an augmentation of Chaos and the Immune Network [40] termed as *ChaoticImmuneNet*. It leverages an SNN combined with transfer learning and concepts from AIS and chaos theory. While the SNN operates even under low data availability, the AIS provides for onboard EAI to the*ChaoticImmuneNet*. The chaotic mechanism allows for efficient traversal of the EAI optimisation space. Experiments covering the ablative analyses of *ChaoticImmuneNet* validate the role of each of its components. Comparison of multi-class classification performance of *ChaoticImmuneNet* with existing approaches for EDs are also presented. The image classification performance of *ChaoticImmuneNet* has been tested with the MNIST [37], the *Local* and the KMNIST datasets [39].*ChaoticImmuneNet* has also been tested on real-world scenarios by deploying it on a real mobile robot, equipped with an onboard camera, operating in a prototype warehouse. *ChaoticImmuneNet* was found to be able to successfully act as an EAI method for resource-constrained EDs by minimising onboard compute-intensive gradient-based training. It was also able to limit its memory footprint by relying on the latent space of the SNN. Experimental analyses conducted on high-dimensional and noisy computer vision data revealed the robustness of this proposed method. This contribution successfully forges a synergistic link developed between connectionist and evolutionary AI techniques. However, a major chunk of the computational load, due to gradient-based training, is still performed onboard the EDs. The subsequent contribution alleviates this issue by completely removing gradient-based learning performed at the EDs.

1.3.4 Domain Adaption for Edge Devices

This contribution also deals with the usage of ML techniques onboard the EDs. However, it emerges from the idea of relieving the computational overload from the ED. This is achieved by leveraging simulated data to augment the learning of the pre-trained model. Capturing and labelling real-world data proves to be costly and time-consuming. Most often simulators are used for this purpose. Generating simulated data using *Cloud*-hosted or local simulators is far easier and involves limited infrastructural setup. However, using a model that has been trained using simulated data in the real world often suffers from the *domain shift* issue [41]. The model trained on simulated data is unable to deliver satisfactory performance when posed with noisy real-world data. While there exist DA methods for transferring knowledge from virtual to real domainS (V2R), most of them assume a pre-determined mix of simulated and real training data to represent the real-world data observable at run time [42]. However, such an assumption fails when dealing with EDs. For instance, a mobile robot equipped with an onboard camera and using a mix of simulated and real-world data could encounter visual noise due to vibrations, object warping, occlusion, scaling, variation in lighting, etc., leading to a fiasco. This can be circumvented only if, in addition to V2R knowledge

transfer, the pre-trained model also undergoes onboard learning, post-deployment on the robot (ED). Gradient-based DNN training techniques can prove to be costly when performed onboard an ED. GAs are known to perform better than gradient-based methods for optimising neural network parameters [43, 44]. However, for high-dimensional data such as in the case of computer vision, optimising the vast search space using GAs is non-trivial. Achieving such a V2R-based onboard learning technique onboard on an ED is challenging. This contribution proposes *GADANN*, that merges concepts from domain adaptation **(DA)**, genetic algorithms **(GA)** and TinyML (D**NN**) to meet the aforementioned demands. Experiments comparing *GADANN* with existing V2R methods, including generative AI based methods, highlight the utility of *GADANN* for real-world scenarios where the whole of the data is not available at once and the underlying model is required to adapt to the incoming data. Ablative experiments justify the role of each component of the GA on the performance of *GADANN*. The real-world and the virtual dataset used in the experiments were created in the lab., both in the real and the virtual world environments, respectively. The proposed method was also deployed on a mobile robot, both in the virtual (simulated) and the real worlds for a pallet identification task in a prototype warehouse. The real-world experiments highlighted the impact of the *domain shift* issue on the performance of a pre-trained model in the real world. The capability of *GADANN* to adapt, in real-time, to the noisy and high-dimensional real-world robot vision data was also testified. Despite the onboard learning capabilities, *GADANN* offers a lightweight solution, by replacing gradient-based onboard training of DNNs with a GA-based optimisation strategy. The cornerstone of this contribution is the reliance on simulated data and its corresponding real-world data that facilitates the DA task. In the domain of robot vision, the availability of such a dataset that comprises data from both domains is scarce. This instigated the requirement for creating an easy-to-use domain adaptation dataset for robot vision which has been explored in the next contribution.

1.3.5 A Dataset for Domain Adaptation tasks for Robot Vision

There exist plenty and a variety of datasets for computer vision based ML tasks. While these datasets have corresponding utilities across different niches, they are unable to cater fully to the robot vision domain. Robot vision often involves a mobile robot, equipped with a vision-based model, as an active entity that interacts with its environment. Under such a setting, several data variations occur due to environmental changes such as environmental noise or transformation and also due to the action of the robot itself, such as scaling, motion, occlusion, etc. While DNN-based models can be used to tackle such a variation in the data distribution, DNN requires a large amount of training data. Capturing and processing real-world data, using a robot, is a costly and time-consuming affair. This necessitates the usage of simulators that provide close to real simulated environments for experimentation. However, a DNN model trained solely on simulated data will fail to perform well in the real world due to the *reality gap*. This has given rise to a variety of DA methods for transferring knowledge from the simulated world to the real world. Corresponding datasets have also emerged to facilitate DA-based tasks. However, the existing datasets in this domain are not curated specifically for mobile robots, leaving multiple open issues faced under robot vision tasks. An immediate requirement is thus to facilitate a multi-domain dataset that is easy to work with and uses standardised apparatus for easy reproducibility of robot vision based experiments. This contribution caters to this requirement by providing RoboDA. RoboDA is a robot vision based dataset, curated specifically for DA-based tasks. The dataset has been created by using a mobile robot, both in a simulator and the real world. The apparatus used to create RoboDA is standardised and open source, thus allowing anyone to extend the dataset and its related experiments. This contribution conducts a detailed survey of the existing V2R-based

datasets and identifies the challenges and open issues for each of them. It also elaborates in detail the acquisition process of the RoboDA dataset. Experiments conducted using the RoboDA dataset with multiple state-of-the-art DA-based methods, including generative AI based methods, highlight the challenging feature landscape of RoboDA, useful in evaluating DA-based methods. RoboDA has also been compared with a standard image dataset to highlight the efficacy of the former for robot vision tasks. This contribution was made with a vision to share with the robotics research community a well-curated and intensively evaluated data useful for quickly realising and evaluating robot vision based experiments.

1.3.6 Onboard class incremental learning for extremely resource-constrained scenarios

Across the previous contributions, the thesis has dealt with multiple issues, including proactive monitoring and response using a CPS, learning under resource constraints, tackling domain shifts and real-world noise, etc. However, none of the contributions discussed so far deal with the accommodation of data from new classes at run-time. Observing data from new classes, at run-time, is a common phenomenon in the real world, such as the addition of a new employee to the attendance records, the addition of a new gesture to a wearable monitoring system, the addition of a new audio source to a speech recogniser, etc. Class Incremental Learning (CIL) is an umbrella term that comprises methods to incrementally absorb the knowledge of data from new classes to pre-trained models while retaining the older knowledge. Another aspect left open, in the first contribution, is to tackle ML for time-series data on resource-constrained devices. While gesture recognition was an aspect of the first contribution, the recognition was based on a rudimentary pattern-matching technique that does not accommodate noisy and a variety of patterns for recognition. This contribution acts as a culmination of all the research efforts made across the previous contributions under this thesis. A major accomplishment under this work is the extension of TinyML approaches for onboard CIL on extremely resource-constrained devices. It also brings to test the experience gained so far in tackling a real-world problem of gesture recognition on time-series based sensor data. The use of GA to extend TinyML beyond just onboard inference capabilities is another novel aspect of this contribution. Experimental analyses comparing the performance of the proposed method with existing CIL strategies testify to the utility of the former. Portions of the WISDM [45] dataset and a locally generated dataset created by using the MPU6050 sensor [46] were used in the experiments. Deployment of the proposed method on an extremely resource-constrained device proves its efficacy for real-world applications. The method is generic and lightweight which opens up opportunities of extending this contribution to a rich set of applications. Having dealt with time-series gesture data, this contribution connects back to the first contribution, delivering a closure to the research endeavour under this thesis.

Figure 1.1 shares a snapshot of the contributions made under the thesis and their relationship with each other.

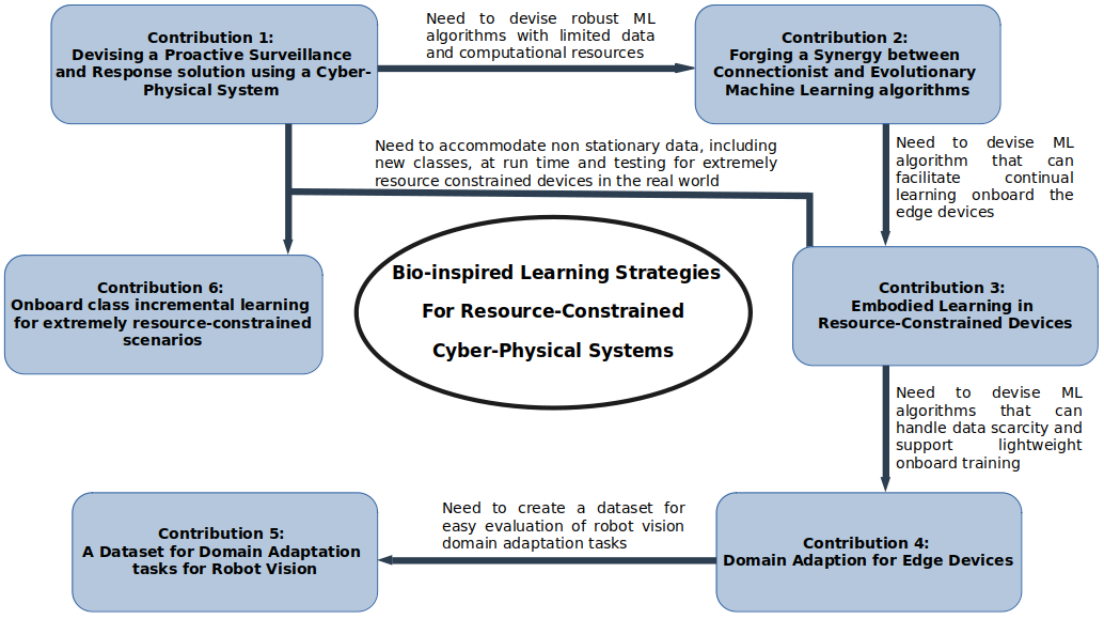


Figure 1.1: Contributions made under the thesis and their inter-relations

1.4 Outline of the Thesis

The thesis comprises eight chapters, described briefly as follows:

Chapter 1: This chapter briefly summarises the motivation and the research contributions made across the thesis.

Chapter 2: This chapter discusses the first contribution that realises a CPS using the IoT to tackle a real-world problem of child trafficking. The chapter is based on the published work reported in [47]

Chapter 3: This chapter revolves around the proposed method of creating a synergistic relation between a connectionist and an evolutionary technique. The resulting performance increment achieved under this contribution motivates further exploration as undertaken in the subsequent contributions. This chapter is based on the published work reported in [48] also covered in [49]

Chapter 4: This chapter discusses a method for onboard learning for resource-constrained devices. The proposed method was tested on a real-world prototype warehouse developed in the Robotics lab. of the Department of Computer Science and Engineering at the Indian Institute of Technology Guwahati, India. This work ushers the aspect of real-world usage of ML approaches for resource-constrained devices which is observed in most of the subsequent contributions.

Chapter 5: This chapter enhances the scope of the thesis to include simulated data for buttressing the usage of onboard learning for resource-constrained EDs. It also highlights the role of evolutionary techniques by providing it as an alternative to gradient-based techniques for onboard learning. The proposed method was tested on a real-world prototype warehouse developed in the Robotics lab. of the Department of Computer Science and Engineering at the Indian Institute of Technology Guwahati, India.

Chapter 6: This chapter presents a DA related dataset, specifically curated for robot vision-based tasks.

Chapter 7: Having explored multiple techniques of onboard learning for resource-constrained devices, this chapter extends the research landscape of this thesis further by dealing with class incremental learning. It also penetrates into the new modality of the time-series domain data. The research endeavours conducted so far are put to the ultimate challenge of realising onboard learning on an extremely resource-constrained ED (4 MB memory and the size of a thumb). From an application point of view, this contribution caters back to the gesture recognition task that was undertaken in the first contribution, this time by using efficient ML approaches. An excerpt from this work has been published as [50].

Chapter 8: This chapter summarises and concludes the research activities undertaken within this thesis. It also discusses the potential future research exercises that the thesis encourages to undertake.

2

Smart Patch: Leveraging an IoT based CPS for Proactive Surveillance and Response

IoT based CPSs involve sensors and actuators connected over the Internet, communicating with each other to perform complex tasks [51]. Such an arrangement suits applications demanding real-time solutions, in conjunction with the real world. However, many a time, the target applications require affordable and resource-constrained components that can ensure ease-of-use. This chapter discusses one such scenario wherein an IoT-based CPS is leveraged to cater to the issue of child-trafficking. Since the issue requires a non-intrusive and real time solution, the contribution utilises components having a low form factor and low resource-requirements. The components are used to realise an ecosystem that allows proactive surveillance and response against child trafficking. However, ensuring a reliable, affordable and easy-to-use solution, targeted for children, is a non-trivial task and requires integration of multiple aspects with careful analyses of each, as explained in the rest of this chapter.

Human trafficking, as per the United Nations Trafficking in Persons Protocol, refers to the recruitment, transport, transfer, harboring or receipt of a person by such means as threat or use of force or other forms of coercion [52]. More than 40 million people across the globe were affected by human trafficking in the year 2016 [53]. It is a complex criminal activity and is extremely difficult to detect and track, owing to its clandestine nature. Victims generally remain reluctant to speak out or take action because of the lack of resources and awareness [54], absence of accurate information and even social stigma. The United Nations has reported active trafficking across 106 countries [55], making it despicably the third largest profitable criminal activity in the world [56]. In the 2016 Global Slavery Index, 18.3 million people were reported to be in modern slavery in India [57].

Unlike in the case of adults, children get trafficked unknowingly and without consent; with virtually no chances of escape. A crucial step in such scenarios is to somehow at least detect the occurrence of the trafficking incident at the earliest. A child must thus, be constantly monitored and tracked, by using proactive mechanisms and real-time responses, so that any possible bid at trafficking is detected and relayed to the guardian as soon as possible; to initiate actions to thwart the attempt.

The devices used to implement such mechanisms should preferably be indiscernible and hassle-free in terms of their usage. Wearable and smart tracking devices that are either camouflaged or hidden beneath an apparel [36] can prove to be more effective under such scenarios. Existing wearable solutions suffer from limitations such as high communication bandwidth requirements, high power consumption, manual operation, etc. Low-cost and low-power embedded systems can be used as alternatives to such devices. The advent of such systems, along with a wide range of

sensors, has facilitated the IoT to be used in a plethora of services. In this contribution, we realise a CPS by leveraging the IoT to target the issue of child trafficking. We embed and conceal both sensing and communication capabilities within a patch attached to a shoe, and use it as an IoT node that can connect to a smartphone remotely. The shoe continuously monitors the location of the wearer and provides options to generate automated or manual emergency stealth signals allowing a swift response in the case of an emergency.

The proposed solution strives to reduce the latency in detecting a child trafficking incident by continuously monitoring and spontaneously generating responses as and when the child wanders or is forced into a forbidden area where he/she ought not to be. It also empowers the child to willingly generate and send a covert signal to a remote guardian at times of distress. Though the motives behind child trafficking and kidnapping differ, in this contribution we assume the former to be synonymous with the latter since the solution portrayed herein, could be used to foil both.

The work conducted in this chapter is focused on realising an IoT-based solution against child trafficking. The availability of affordable and lightweight embedded circuitry along with the widely available IoT communication channels has motivated the efforts conducted under this chapter to realise a real-time surveillance and response system.

2.1 Related Work

Several attempts at anti trafficking/personal safety have been reportedly made in the form of ingenious mechanisms. Existing solutions [58, 59, 60] either require external devices/tags to be carried by the user or consume high communication bandwidth and power while working continuously and autonomously. Most of these devices are GPS based [61] that sense and communicate coordinates of the wearer using an underlying mechanism. However, such GPS based devices tend to perform poorly in indoor locations or around buildings. They are also affected by weather conditions [62, 63]. Patel et al. [64] have used a concealed apparatus that responds in case of an assault. It responds to stress/assault but may not be necessarily proactive to be used in case of child trafficking. Since children can often be cajoled into such situations or are often unaware of such insidious threats, anti trafficking mechanisms need to be proactive. Moodbidri et al. [37] have specifically targeted child safety through their solution but the mechanism demands active participation on part of the parent. The parent has to manually ping the child's wearable device to get his/her status. Such systems do not have real-time responses and are not easy to use. Raflesia et al. [65] have proposed an application to be used in a smartphone that alerts the parent based on the situation of the child. This incurs additional cost as it inherently implies that the child needs to possess and carry a smartphone. Children are seldom allowed to carry smartphones and even if so, these are the first things to be seized by an offender. Proactiveness, autonomy and secrecy thus are mandatory requirements in every anti child-trafficking mechanism.

2.2 Background

Detection of a potential child trafficking incident involves sensing his/her state (whether in distress or not) and tracking the current location. A brief discussion of these two activities follows.

2.2.1 Sensing

To tackle the problem at hand proactively, the intended system needs to have the capability to sense (monitor) the child's current state and respond in case of an emergency. By the current state,

we mean whether the child is exhibiting a normal behaviour or is in distress. Several methods of sensing the current state or location of a subject have already been implemented in the form of interactions through additional external devices such as phone [60], tags [59], etc. However, in such cases the interacting devices used are either not concealed [60] or are too small for the user to remember where it was kept [59]. An adequate system would thus need to be non-intrusive, elicit minimal manual interactions and tether to the child in a concealed manner. The action of monitoring the child also needs to be autonomous, prompting responses only when required, thereby saving power consumed and preventing potential errors associated with manual monitoring.

The system should also have the facility to respond on-demand, based on the subject's intent. Hence, in addition to an automated emergency response, the system should also facilitate the ability to generate a manual emergency response. For instance, a child could trigger a voluntary response using a gesture.

2.2.2 Tracking

Tracking the child's current location is crucial since his/her position could change throughout the day (e.g. home, school, playground, etc.), spanning a substantial geographical area. A few prominent location tracking technologies viz. RFID, Infra-Red (IR), GPS and signal strength based systems, are discussed below.

In RFID based systems, the position estimation is based on electromagnetic communication between RFID readers and associated tags. RFID tags can be passive or active. The cost of compatible readers [66] and the limited range of the passive tags (approximately 1-2 m) form the major drawbacks of RFID technology. Infrared based systems provide wireless data transmission via infrared radiation. They require unobstructed line-of-sight paths between the transmitter and the receiver. Any interference from strong sources of light can cause issues. Infrared indoor positioning systems are thus limited to transmission within a single room [67].

GPS based systems are beneficial outdoors but provide limited coverage indoors [68]. They are not reliable for indoor environments because of the barriers posed in the line-of-sight transmission between the receivers and the satellites [69].

Signal strength based systems detect the strength of an ambient signal and ascertain the distance from the source. Such systems use Bluetooth Low Energy (BLE), Wi-Fi, ZigBee and other wireless systems as either sources or detectors. BLE based systems provide for indoor localisation [70]. The BLE devices are generally installed in an indoor environment so that their locations are known a priori. Location estimation is fast and straightforward. BLE beacons are also easy to manipulate and configure, but the positioning results are not always good enough [71].

FM radio-based systems have broader coverage and can work both indoors and outdoors. However, their indoor performances are fair only for confined spaces [72].

Wi-Fi networks have also been used for localisation [73]. The accuracy of typical Wireless LAN (WLAN) positioning systems using the received signal strength is approximately 3 to 30 meters [74]. Within this range, the Received Signal Strength Indicator (RSSI) values of the Wi-Fi signals can be mapped to the distance and hence the location of the intended subject being tracked.

Other techniques include the use of acoustics, cameras, ZigBee technology, etc. but they either demand dedicated infrastructure or have higher processing overheads. Though some implementations of localisation involve tracking a device worn by a child [75], they do not work for long ranges.

A Wi-Fi-based localisation technique can provide substantial coverage. The Service Set Identifier (SSID) of a Wi-Fi service can be leveraged to distinguish among a group of Wi-Fi networks to estimate the current location of the subject. Though contemporary technologies for localisa-

tion provide better accuracy, in certain cases, using Wi-Fi is a better choice because Wi-Fi based localisation is less demanding on the infrastructural requirements. Wi-Fi routers are becoming commonplace and are comparatively inexpensive and easy to maintain. Further, knowing the exact location of a child being tracked may not always be required. It may be sufficient to know whether the child is within an area where he/she ought to be. All this makes the use of Wi-Fi viable from the point of view of deployment, cost and scalability. Also, the advent of smart cities will soon make Wi-Fi stations ubiquitous and publicly available, further reducing infrastructural demands.

The Wi-Fi-based localisation fits well with the requirements of the proposed solution. This is so because the problem at hand involves dense environments (for instance school premises) that are not suited for line-of-sight based tracking systems. Additionally, the solution needs to be tolerant to disturbances (radiation, weather, light, etc.) and have a moderate range of coverage. Moreover, the solution lays more stress on the timely notification and proactive monitoring and tracking rather than on post-incident (after the child has been trafficked/abducted) tracking of the child. Thus Wi-Fi-based localisation technique tends to strike the balance for the solution.

2.3 *Smart Patch* based system: The *Smart Shoe*

Under this contribution, we propose a distributed system comprising a *Smart Patch* that can be concealed within an apparel worn by a child and a smartphone in possession of his/her remote guardian. These two components, along with a cluster of Wi-Fi networks, an Internet router and the intermediary communication applications, create a surveillance and response ecosystem. In this contribution, we describe how such a patch, housed within a shoe, can aid in thwarting child trafficking. Compared to other articles of clothing, the shoe being a sturdy and well-tethered component of the wearer, forms an ideal component to conceal the patch. The proposed solution also facilitates a certain atypical gesture that can initiate a covert distress call to the child's remote guardian, in times of need. Under such situations, the foot and hence the shoe tend to attract less attention while making such gestures, as compared to those made by a hand with the patch incorporated as a bracelet or wristband. The *Smart Shoe* populated with an embedded controller and associated sensors, is capable of sensing whether the child is in an *allowed* or a *forbidden* zone. An *allowed* zone is a Wi-Fi network coverage area that has been permitted a priori by the guardian for the child to be in for a particular duration. Areas other than these are designated as forbidden. The guardian, if required, has the facility to change such areas into either *allowed* or *forbidden* zones, remotely. The system continuously senses the zone in which the child is currently located and generates automated emergency responses to inform the guardian as and when the child moves into a forbidden zone or has gone too far from an allowed Wi-Fi network coverage area. Similar responses are generated if the shoe has been taken off by the child. Such responses are involuntary in nature and are categorised as Involuntary Emergency Responses (IER). Figure 2.1 depicts a typical *Smart Shoe* usage scenario. As can be seen, if a child C2 is under a forbidden zone/evading the Wi-Fi network coverage then his/her guardian G2 receives an IER through a smartphone indicating so. In case of an emergency, the *Smart Shoe* also allows the child to voluntarily raise a Voluntary Emergency Response (VER) by making a covert gesture using the foot.

This also has been depicted in Figure 2.1, wherein the child C3 is in an unattended *allowed* zone. Upon being forced by the trafficker T, the child voluntarily makes the covert gesture using the foot with the *Smart Shoe* to generate a VER. This in turn, is received by the smartphone in possession of his/her guardian G3. The overall *Smart Shoe* system may be divided into two sub-systems:

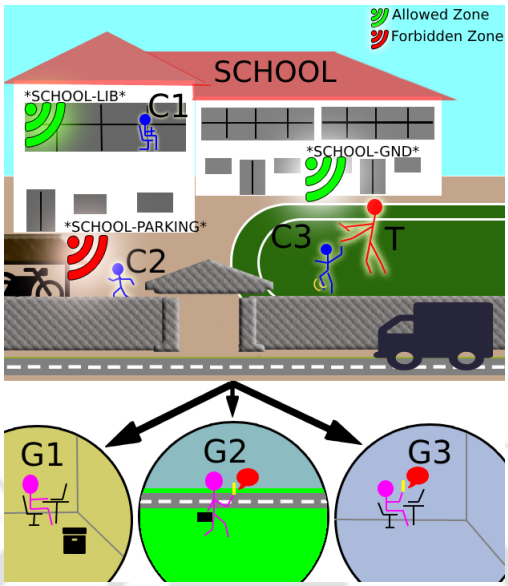


Figure 2.1: The *Smart Shoe* system

2.3.1 The Sensory Sub-system

This sub-system caters to sensing whether the shoe is worn or removed, sensing location of the *Smart Shoe*, and IER and VER triggering.

Shoe Occupancy Sensing and IER generation

The occupancy of the shoe is continuously sensed by a sensor embedded within the shoe. It ensures that the shoe is always tethered to the child so as to respond in case of an emergency. The moment the shoe is taken off, the sensor triggers an IER.

Location Sensing and IER generation

A guardian initially feeds a list of SSIDs of the Wi-Fi networks of the *allowed* zones to the *Smart Shoe*. For this, the guardian may need to coordinate with the authorities of the expected locations of the child, to obtain their respective Wi-Fi network SSIDs. Using this list and the ambient Wi-Fi networks, the *Smart Shoe* ascertains whether the child is currently in an *allowed* zone or not. This decision is made by sensing the RSSI values for all the ambient Wi-Fi networks and then assuming the current zone to be the one having the highest RSSI value. If the child is in a forbidden zone, an IER is promptly generated to inform the guardian that the child has crossed over to a forbidden area, via the Internet router. This response is also generated when the RSSI values of the Wi-Fi networks, within the *allowed* zones, fall below a pre-determined threshold. This is done to indicate that the child has moved too far from the allowed Wi-Fi network coverage area into a weak or no network area. Such a response signals the increase in chances of an imminent loss of location tracking.

State Sensing and VER generation

There could be cases when a child becomes aware that his/her safety has been compromised. In such an event, the *Smart Shoe* allows the child to inform the guardian of his/her state of distress

by using a covert foot gesture. Such a VER can be generated irrespective of the zone in which the child is located. A sensor implanted within the shoe captures the angular and acceleration profile of the movement of the foot so as to determine the gesture. An actuator within the shoe gives feedback to the child when the gesture is successfully sensed. A VER is generated only on confirmation by the child to avoid false or unwanted triggering.

2.3.2 The Communication Sub-system

This sub-system ensures a remote and two-way communication between the child and the guardian. The guardian can send the list of *allowed* zones and also get the child's current Wi-Fi based location using the Message Queuing Telemetry Transport (MQTT) protocol. The *Smart Shoe* and the guardian's smartphone exchange messages over a cloud server using a mobile application, thus allowing communication over a longer range. Additionally, the *Smart Shoe* communicates with the guardian involuntarily or voluntarily using custom TCP/IP protocol through another application. The guardian can tweak the response mechanism by updating the *allowed* zone list over a cloud server using a mobile application running on his/her smartphone.

2.4 Implementation

The *Smart Shoe* system is an integration of a Shoe-*Smart Patch* assembly, *Wi-Fi networks*, an *Internet Router*, a *Cloud Server* and a *Smartphone*.

The implementation of the proposed system is described herein from the perspective of the child and the guardian.

2.4.1 Child Side

This side pertains to the *Smart Shoe*, its composition and working. A Node MicroController Unit (NodeMCU), embedded within the shoe, forms the main controller. This unit provides an open-source software and hardware development platform which is based on an inexpensive System-on-a-Chip (SoC) that mainly houses a CPU, RAM and a Wi-Fi module. With a reasonably low footprint, this cheap controller can be considered as a near-ideal node in an IoT. A force-sensitive resistor to detect shoe occupancy, an accelerometer cum gyroscope sensor (MPU6050) used to capture the covert gesture, a buzzer for gesture confirmation and a power source (battery), all of which are interfaced to the NodeMCU; form the remaining components of the *Smart Shoe* circuitry. Figure 2.2 shows the placement of the circuitry within the shoe. It can be observed that the circuitry is strategically spread across the shoe to keep it concealed. The power source for the circuitry constitutes a 3.3V LiFePO4 battery.

In order to track the child's current zone, a cluster of Wi-Fi networks was used to localise the child among the *allowed* and *forbidden* zones. In addition, an Internet router was employed for Internet connectivity across these zones to communicate the responses to the guardian. Since this router has to cover all the zones to provide the Internet connectivity, it is not used to localise the *Smart Shoe*. The NodeMCU microcontroller in the shoe has a Wi-Fi module embedded within which handles the Wi-Fi network sensing and the Internet communication tasks.

Shoe occupancy is detected using a force sensitive resistor. This type of sensor changes its conductance when it is either bent or stressed. The sensor was embedded beneath the tongue of the shoe over which the lace was tied. The extra pressure exerted by the shoe tongue and the foot on the sensor due to the tying of the shoelace, changes its conductance. This change aids in

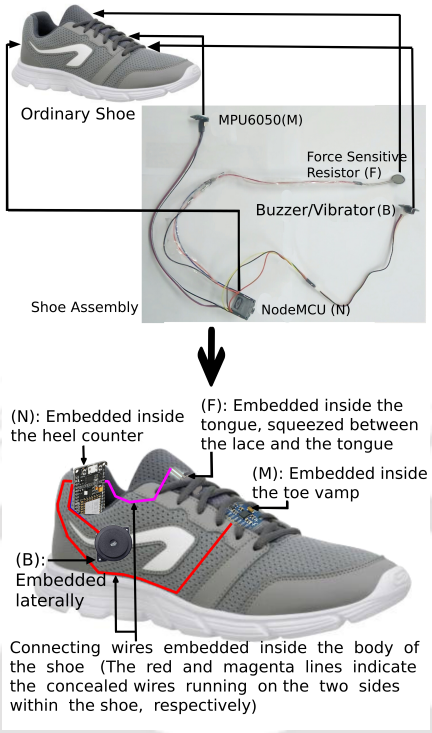


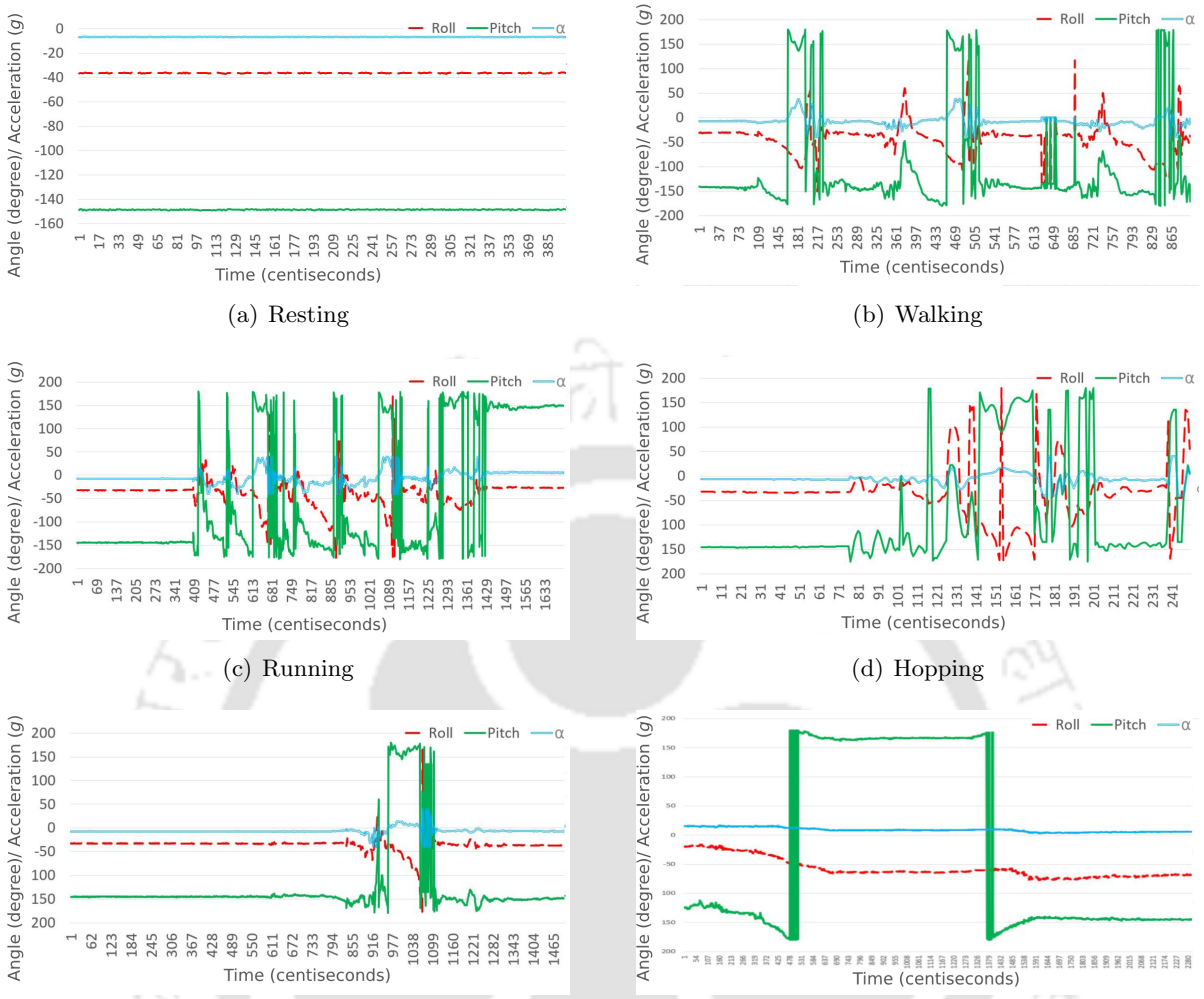
Figure 2.2: Placement of the circuitry within the *Smart Shoe*

detecting whether the shoe is worn or removed. When the shoelace knot is undone, the pressure on the sensor eases off, which consequently triggers an IER.

On sensing a threat, the child can generate a VER by making a covert gesture with the foot, which in turn is detected using the MPU6050 sensor. A child may end up enacting this specific gesture unintentionally, triggering an undesired VER. To avoid/minimise such false alarms, this gesture needs to be unique. Thus, a contained analysis of the gait patterns was performed to find a unique foot gesture which did not overlap with any of the routine movements of the foot (Figure 2.3).

The graphs in Figure 2.3(a) through 2.3(e) depict the angular and acceleration profiles of the foot for routine actions viz. resting, walking, running, hopping and leaping; respectively. It can be observed here that for routine movements (except while resting), the green coloured pitch curve cycles above and below both the red coloured roll and the blue coloured α curves. α is the acceleration of the foot along the z-axis (measured relative to the gravitational acceleration g) scaled up 20 times.

After several experiments, a unique combination of the roll, pitch and α values of the sensor for the foot was found which made the covert gesture reliably distinct from other routine foot movements. As seen in Figure 2.3(f), the associated covert gesture is such that it ensures the pitch to move beyond both the roll and α values and then maintain the same, long enough to be distinctly detected. Physically, this gesture needs to be enacted in two-stages. The first stage requires the foot to be tilted forward leftward for a minimum of 4 seconds, thereby making the pitch to shoot above the roll and α values. In the second stage, the foot needs to be tilted forward rightward for at least 4 seconds, resulting in the pitch to move to a portion well below the other two values. Any deviation from this sequence nullifies the gesture in which case the child needs to redo both these



(a) Resting

(b) Walking

(c) Running

(d) Hopping

(e) Leaping

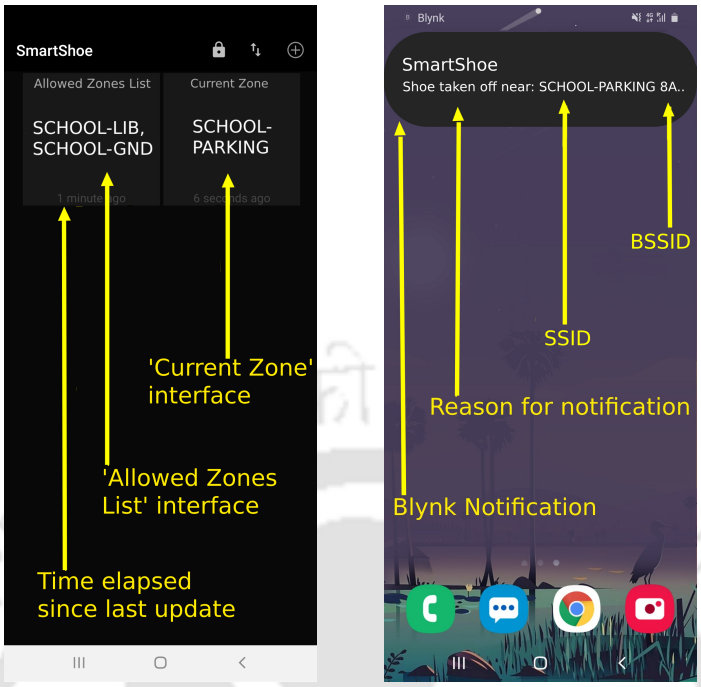
(f) Covert Gesturing

Figure 2.3: Angular (Roll, Pitch) and Acceleration (α) profiles of the foot wearing the *Smart Shoe* over different routine activities and the covert gesture

stages of action. The buzzer notifies the completion of the first and second stages separately and thus makes the child aware of the generation of the VER. If the child has unknowingly enacted the first stage, the buzzer informs him/her of the same. In such a case, the child needs to ensure that he/she does not perform the second stage, thereby cancelling the VER generation procedure.

2.4.2 Guardian Side

In the current implementation, the guardian needs to install two third-party applications viz. *MQTT Dash* [76] and *Blynk* [77] on his/her smartphone to facilitate a remote connection with the *Smart Shoe* through the *CloudMQTT* [78] server. The *MQTT Dash* application is configured to have two interfaces: 'Allowed Zones List' and 'Current Zone', as shown in Figure 2.4(a). The 'Allowed Zones List' is a text field that can be used to send the list of *allowed* zones to the *Smart Shoe*. This list replaces any earlier present list, thereby updating the *allowed* zones. This interface, when not being used to send the list, displays the latest *allowed* zones list. The 'Current Zone' interface displays the SSID of the Wi-Fi network that the child is currently under. In Figure



(a) MQTT Dash (b) Blynk

Figure 2.4: Snapshots of the two apps running on the Guardian's smartphone

2.4(a), 'SCHOOL-LIB' and 'SCHOOL-GND' are the SSIDs of the Wi-Fi networks corresponding to *allowed* zones while 'SCHOOL-PARKING' is the SSID of the Wi-Fi network at the zone where the child wearing the *Smart Shoe* is presently located. The *Blynk* application is used to receive the emergency notifications (viz. IER and VER) sent by the *Smart Shoe*. As shown in Figure 2.4(b), the responses from the *Smart Shoe* are received in the form of a notification that conveys the cause of response, the SSID and the Basic Service Set Identifier (BSSID) of the Wi-Fi network that the child currently is under.

The *CloudMQTT* server and the *Blynk* application require respective login accounts in response to which unique IDs and passwords for communication are received via the user's registered email id. These credentials provide the necessary security for the communication between the guardian and the *Smart Shoe*.

2.5 Experiments & Results

To test the efficacy of the proposed approach, several experiments were carried out in different environments (rooms, courtyards, open spaces, etc.) each covered under a cluster of WiFi networks along with a dedicated Internet router. Subjects were made to wear the *Smart Shoe* and move around. The specifications of the power supply and ambient temperature for the *Smart Shoe* used in the experiments are detailed in Table 2.1.

The analysis of the experiments has been detailed in subsequent sections.

Table 2.1: Experimental specifications of the *Smart Shoe*

Power source (Battery)	Capacity	10000mAh, 37Wh
	DC input	5V, 2.5A
	DC output	5V, 2.1A
Consistent Continuous usage	Tested	8 hours
	Suggested	8-12 hours
Operational temperature	Ambient temperature	25-26 °C
	Microcontroller temperature	35-36 °C

2.5.1 Occupancy Sensing

The effectiveness of shoe occupancy detection was analysed by positioning the force- sensitive resistor sensor at three different portions of the shoe, as depicted in Figure 2.5.

Portion-1: The sensor was placed beneath the lace knot area within the tongue of the shoe. Portion-2: The sensor was placed on the lateral side within the shoe. Portion-3: The sensor was placed upright behind the heel inside the shoe.

Occupancy sensing detection was conducted for these three different sensor placements by making the subject sit, stand, walk, run and hop. Figure 2.5 shows the results obtained for 20 trials for these sensor placements. As can be seen in this graph, placing the sensor in Portion 2 and 3 exhibited substantial inconsistency (50%) while the subject walked, ran and hopped. However, when the sensor was placed in Portion-1, the consistency rose to 95%. This higher accuracy in Portion-1 may be attributed to the higher forces exerted on the sensor by the tied shoelace and the tarsal and metatarsal regions of the foot. Even in case of any occasional retraction of the foot muscles within the shoe, this placement ensures that the minimum amount of pressure is maintained for the sensor to indicate occupancy.

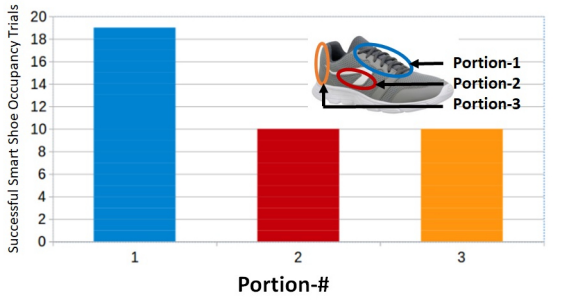


Figure 2.5: Analysis of *Smart Shoe* Occupancy Sensing

2.5.2 Gesture Recognition

The gesture recognition based response (VER) generation was found to be fairly reliable. Upon making the covert foot gesture, the response was always generated. During the normal actions of resting, walking, running, hopping and leaping, the system behaved in accordance, by not generating a false VER. However, in case of unintentional execution of the first stage of the gesture, the buzzer

alerted the user of the same. This warning ensured avoidance of any false VER being generated. The buzzer could be replaced by a vibration motor, to keep the VER generation concealed and non-intrusive.

2.6 Chapter Summary

The work reported herein describes a *Smart Patch*, as a part of a CPS, constituting a microcontroller and associated sensors embedded within, that endeavours to continuously ensure that the wearer, viz. a child, is always within an allowed area, thereby curtailing the possible attempts at trafficking.

The embedded circuitry facilitates autonomous monitoring and generation of need-based emergency responses. The low-footprint circuitry is well concealed so that it does not draw the attention of the trafficker or kidnapper. Tracking is performed based on Wi-Fi SSIDs which in future will become fairly ubiquitous, especially with the advent of smart cities. While the IER is generated without the intervention of the child, the VER ensures that the child can, at will, communicate to the guardian in case of an emergency even in *allowed* zones. The current net cost of the additional patch circuitry could be just around USD 15 and can be greatly reduced if all components are integrated together as an SoC and manufactured on a large scale. This could make it affordable for the underprivileged section as well. The current implementation contributes towards the prevention of child trafficking by restraining the child's movement to predetermined Wi-Fi network coverage areas. Any movement beyond these areas is immediately intimated to the remote guardian so as to facilitate quick detection of a possible danger. The proposed *Smart Patch* Is not fully fool-proof as it assumes seamless Wi-Fi connectivity. A GSM module, if integrated into the *Smart Patch* could remove this dependency and allow for tracking in areas where the mobile signals are prevalent. This augmentation can also greatly ameliorate post-incident tracking of the child. In addition to trafficking, the patch can be modified to cater to a variety of other scenarios including surveillance in restricted areas, non-intrusive monitoring of the elderly and demented, etc.

This contribution described in this chapter lays the foundation for the rest of the research work reported in this thesis by realising a CPS that aided in familiarising and exposing inherent issues while working with resource constrained devices.

From an application point of view, this contribution presents an affordable, proactive and easy-to-use solution that can be used for allowing surveillance and response against child trafficking. However, from a research perspective, it opens up opportunities to contribute to the inherent deficiencies in such a system. The pattern matching-based gesture recognition employed in the proposed solution is one such component that needs to be improved further by utilising ML-based recognition. The main challenge however, comprises the hosting of such ML-based solutions within the associated resource-constrained devices. The rest of the contributions reported in this thesis therefore strive to achieve this goal by opting to follow a staggered approach of exploring and devising resource-efficient algorithms realise onboard ML.

3

Immuno-inspired Augmentation of Siamese Neural Networks for Multi-Class Classification

CPSs most often have to deal with real-world data that is complex and noisy. While robust ML techniques, such as deep learning, are able to handle such complex data distributions to quite an extent, they require a computationally rich infrastructure complemented by a vast amount of training data. Since, CPSs often involve resource-constrained devices and need to deal with sparsely available data, the demand for computationally efficient approaches that are suited to such environments has become the need of the day. A Siamese Neural Network (SNN) is one such type of DL model that works well with low amounts of data. This consequently, decreases the need for high computational load for training. However, SNNs deal with the quantification of the similarities between data points. They are not well suited for other types of inferencing, for instance classification. The work described in this chapter explores a synergistic amalgamation of SNNs with the concepts derived from the domain of AIS, which in turn strives to convert the similarity space created by SNNs into a classification space, while also boosting the classification performance.

SNNs have been used in a plethora of applications, ranging from signature verification [35] to several instances of multi-class classification [58, 59, 60, 61]. SNNs achieve a similarity metric in the form of a learnt model that allows the discrimination between similar and dissimilar data, resulting in a similarity space. Owing to their useful features like low data requirement, SNNs have also been leveraged into the multi-class classification domain by transforming this similarity space into a classification space [58, 59, 60, 61]. However, the usage of SNNs for multi-class classification is often done at the cost of losing inter-class relations [59], by using random class representations [60] or by involving further complex training procedures such as plugging additional neural network layers on top of the SNN [58, 61], which also tends to limit their scalability. Some other approaches take a heavy toll in terms of high inference times [62, 63, 64, 79]. Finding an approach that can enhance both accuracy and the speed of inference, thus, remains a desideratum.

This contribution proposes a novel immuno-inspired method that enhances an SNN-based multi-class classification approach for images. The method involves distillation of the underlying multi-class dataset to find a better generic representation, for the samples of each class. The representations, referred to as *Gold Standards* (GS), in turn, interact with each other to form a network that brings out the similarities and dissimilarities between various classes, in the form of *stimulations* and *suppressions*, much like an Immune Network (IN) [57], resulting in better prediction performance. The main contributions of the work presented herein are enumerated

below:

- Rather than making an SNN to merely learn the characteristics of all the classes, this work emphasises its usage towards data distillation by generating a GS per class, based on the information collated from the dataset. These GSs interact with one another, resulting in an IN which inherently contains the inter-class similarities/dissimilarities, thereby contributing towards the improvement of classification accuracy, viz. the ratio of the number of correct predictions to the total number of predictions.
- The proposed approach augments the SNN with the IN, during run time, thereby enhancing the accuracy and decreasing the inference time.

3.1 Background and Related Work

This section briefly discusses the two main entities - the SNN and the IN, used in the proposed method along with the related work.

3.1.1 Siamese Neural Network

First introduced by Bromley et al. [35] for signature verification, SNNs predominantly aid in finding the similarity between inputs. A typical SNN consists of two Artificial Neural Networks (ANN) branches (Figure 3.1), fed by an input each, and whose outputs are combined as a single output. Each of the two neural networks transforms the high-dimensional input samples into what are

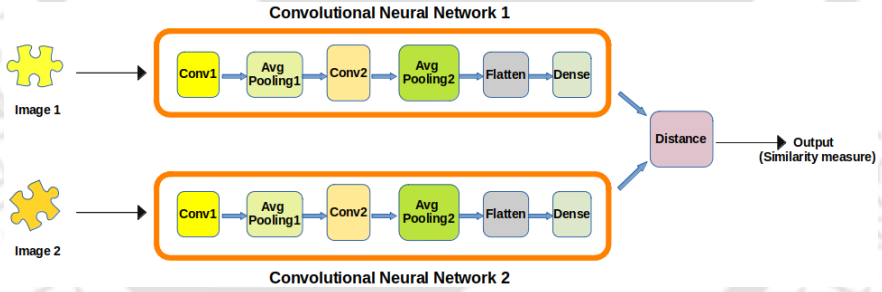


Figure 3.1: A typical CNN based Siamese Neural Network

referred to as *embeddings* in generally a lower dimension. The *embeddings* are then blended together via a distance metric. This is followed by a loss function (cross-entropy loss [53], contrastive loss [54], triplet loss [55], etc.) whose minimisation ensures that the similarity of the inputs is reflected in the output as well.

SNNs have been widely used across varied domains including speech processing, robotics, image analysis, etc. [56]. They have also been used for multi-class classification. However, the transformation from the similarity space to the classification space has been carried out in different ways. Swati et al. [58] have used SNNs with a Multi-Layer Perceptron (MLP) for chromosome classification. The usage of MLP adds an intricate training overhead and also limits the scalability. Nanni et al. [59] attempted animal sound classification by combining SNNs with different clustering techniques such as *K-Medoids* and *K-Means*, for speeding up inferencing. However, the cluster centers did not leverage any inter-class relations. Hindy et al. [60] advocated the usage of SNN for intrusion detection since SNN requires less data and has scalability towards new cyber attacks, thereby obviating the need for retraining. However, classification was accomplished by

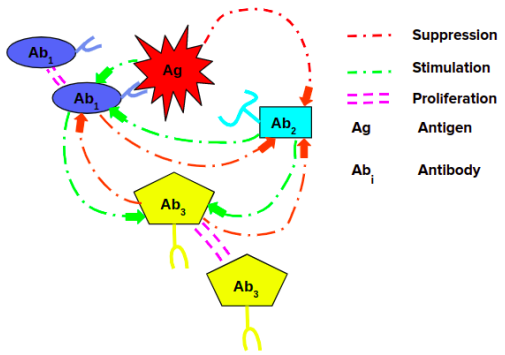


Figure 3.2: Schematic of Antibody-Antigen/Antibody-Antibody Interactions in an Immune Network

combining the SNN model with randomly chosen class representatives that need not always be the best representatives, thus negatively impacting the accuracy. Jiang and Zhang [61] used SNNs for handwritten numeral recognition by plugging a *softmax* layer at the end. This yielded good recognition performance but limited the scalability. Several researchers have proposed classification systems by combining SNNs and the K-Nearest Neighbours (kNN) algorithm [62, 63, 64, 79]. However, as also mentioned in [58], the involvement of kNNs essentially leads to higher inference times and a substantial memory footprint for retention of training data.

3.1.2 The Immune Network

The human Biological Immune System (BIS) is composed of a rich repertoire of cells, aimed at the identification and containment of the invading pathogens. The T- and B-cells together, co-operate to accomplish the task of identifying pathogens. Antibodies, which emanate from the B-cells, have protruding structures on their surface called *paratopes*. Pathogens have antigens on their surface, parts of which could have shapes that can be complementary to some paratopes. These parts are termed as *antigenic epitopes*. When the paratope of an antibody (Ab_1), binds (complementarily) with the epitope of an antigen (Ag), it leads to the recognition of the pathogen. Figure 4.2 depicts such a scenario. This binding results in the suppression of Ag and stimulation of Ab_1, the latter of which then proliferates to create clones of itself. Antibodies, such as Ab_2 whose paratopes bind less, tend to get suppressed, thereby decreasing in population. Thus, *recognisers* dominate the antibody population, allowing better and early containment of the pathogen. Jerne [57] proposed that apart from antigen-antibody interactions, antibody-antibody interactions also take place in the BIS, leading to a complex network referred to as the *Immune Network* (IN). As depicted in Figure 4.2, Ab_1 also recognises parts of Ab_3, resulting in the suppression of Ab_3 and stimulation of Ab_1. After an antigenic encounter, antibodies interact with each other to maintain a good mix of antibodies to be used against antigens. This mix acts as the IN's memory about the antigens. This is summoned when similar antigens are encountered again in future, thereby ensuring a quicker and effective response to the invasion. This theory is often referred to as the Immune Network theory.

Concepts from BIS have been extensively used in applications in the computational domain that have given rise to the AISs. In an AIS, the degree of binding among antibodies and antigens is quantified by *affinity* functions. AISs have been used in a plethora of applications including optimisation, anomaly detection, pattern recognition, robotics, etc. [36]. The proposed work described in the subsequent section, combines an SNN and an IN for multi-class classification to obtain better classification accuracy and also lower inference time.

3.2 Methodology

SNNs create a similarity space to tackle incoming test samples similar to an IN setting in a BIS, wherein antibodies employ their similarity space to identify and contain an incoming antigen. Further, just as a BIS generates *memory cells* which in turn release the right kind of antibodies facilitating a quicker response, an SNN based classification method also requires a mechanism to provide a quick response at the time of inferencing. This stark similarity between the two domains inspired us to draw concepts from the BIS and augment them to SNN based multi-class classification. As suggested in the IN theory, the formation of the IN aids the immune system in two ways:

- The immune system already has the apt collection of antibodies to respond at the time of an antigenic attack, making the response quicker.
- The network formed amongst the antibodies is able to mount a more holistic response as compared to the response of the antibodies in isolation.

The work proposed under this contribution enhances an SNN-based multi-class classification approach by exploiting the aforementioned characteristics of the IN. The training samples within the multi-class dataset used, represent a repertoire of antibodies while the test samples constitute the antigens. Instead of a conventionally used affinity function, we have used an SNN to assess the degrees of binding amongst the antibodies as well as between an antibody and an antigen. These bindings yield good antibodies in the form of GSs. The GSs together form an IN, which in turn, coupled with the SNN, aids in the inferencing process. The overall process is provided in Algorithm 1 and explained subsequently in detail.

Algorithm 1 Multi-Class Classification using Immune Network augmented Siamese Neural Network

Input: *Abs* (Training Dataset), q (Number of classes), *Ag* (Test Data)
Output: Class label of *Ag*

```
Train SNN()
GoldStandards ← SNN(Abs)
Prediction[q] ← 0
for each GoldStandard Ab_GSi in GoldStandards do
    for each GoldStandard Ab_GSj in GoldStandards do
        IN(Ab_GSi, Ab_GSj) ← SNN(Ab_GSi, Ab_GSj)
for each GoldStandard Ab_GSi in GoldStandards do
    Antigenic_Contribution_i ← SNN(Ag, Ab_GSi)
    for each GoldStandard Ab_GSj in GoldStandards do
        IN_Contribution_j ← IN(Ab_GSi, Ab_GSj)
        Prediction_j
        ←Prediction_j + Antigenic_Contribution_i * IN_Contribution_j
return argmax(Prediction)
```

3.2.1 Affinity Function

Due to its notable discriminative ability, an SNN is pre-trained and used as the affinity function to distinguish between samples. The approach begins by training the SNN on a given dataset having

q classes. Every sample, S_j^i, is associated with another sample, $S_{k(k\neq j)}^i$, belonging to the same class, i, leading to the first pair referred to as a *positive pair*, $[S_j^i, S_k^i]$. Another pair is generated by pairing S_j^i, with a sample S_n^m of a different class, m. This constitutes a *negative pair*, $[S_j^i, S_n^m]$. The SNN is trained using all possible positive and negative pairs along with distinct associated labels viz. 0 for positive pairs and 1 for negative pairs. The contrastive loss used for training the SNN is calculated as follows:

$$\mathcal{L} = (1-l) * D^2 + l * max((\tau - D), 0)^2 \tag{3.1}$$

where l is the label of the pair fed to the SNN and D is the Euclidean distance between the *embeddings* of the pair of samples. τ forms the distance margin beyond which the samples of a pair are considered dissimilar. Once the SNN is trained, similar samples will yield a value close to 0 while dissimilar ones yield value close to 1, in accordance with the labelling convention. We have therefore taken the affinity as the inverse of the output of the SNN.

3.2.2 Churning Gold Standards

In a BIS, all antibodies stimulate and suppress each other to eventually yield the best set of representatives that can recognise an antigen. In this work, we have used a similar technique to churn out the best set of GSs, S_{GS}, having a representative from each of the classes. S_{GS} acts as a distilled form of the underlying data, which is used during inferencing as a reference for comparison. Each GS belonging to S_{GS} is chosen via the following process:

- For each class i, for every sample S_j^i, a pre-trained SNN is used to find the affinity of S_j^i with every other sample within the class i. All these affinities obtained for S_j^i are summed together. Since the samples are from the same class, and hence similar in characteristics, the sum of their affinities is considered to be the *stimulation*, ST_j^i, received by S_j^i, as in equation (3.2).

$$ST_j^i = \sum_{k=1}^{t} Aff_{jk}^i \tag{3.2}$$

  where ST_j^i is the *stimulation* for the sample j of class i, Aff_{jk}^i is the affinity of sample k for sample j in class i, having t number of samples. Among all the samples of each class i, the set of top N most stimulated samples, S_i, is chosen.

- With the samples in the set S_i of class i, the affinities of samples in the corresponding sets of other classes is calculated using the SNN. Such affinities are referred as *suppressions*, calculated using equation (3.3).

$$SU_j^i = \sum_{c=1(c\neq i)}^{q} \sum_{k=1}^{N} Aff_{jk}^{ic} \tag{3.3}$$

  where SU_j^i is the *suppression* received by sample j of class i and Aff_{jk}^{ic} is the affinity of sample k of set S_c of class c for sample j of class i.

- For each set, S_i, of a class i, the most suppressed sample is designated as the GS of that class. The suppressive interactions push the GS representative away from class boundaries, thereby refining the GS representation.

The stimulative interactions yield a medoid sample of the class as a potential representative for the GS. If time and computational resources form a constraint, the GS may be found using only *stimulations* and not *suppressions*. This process is performed offline so as to yield the set, $S_{GS} = [GS_1, GS_2 ... GS_q]$.

3.2.3 Creating a Network of Gold Standards

In a BIS, the IN not only yields the best antibodies but also allows antibodies to relay information about one another across the network [57]. As depicted in Figure 4.2, the entire network of antibodies participates in relaying the respective information regarding the compatibility of each of the connected antibodies with the antigen. The GSs, which form the metaphors of the antibodies, mimic such a mechanism as and when a new and unknown test sample (antigen) is presented. Prior to inference, the GSs of different classes form an IN, based on affinity values provided by the pre-trained SNN. Any two GSs interact with one another based on this affinity which represents the weight of an edge between them within the IN.

3.2.4 Classification of a Test sample

Classification of a test sample, S_t, involves finding two contributions towards prediction and aggregating them as shown in Figure 3.3.

- *Antigenic contribution*: The trained SNN is used to compare S_t individually with the members of the set S_{GS}, to generate a set of antigenic affinities, $Aff^{Ag} = [Aff_1^{Ag}, Aff_2^{Ag} ... Aff_q^{Ag}]$ constituting the antigenic contributions of each GS.
- *IN contribution*: A $GS_i \in S_{GS}$ is connected to form the IN, based on its affinities, with all other GSs ($GS_j \in S_{GS}$). Hence, the decision of GS_j regarding S_t, is influenced by the decision made by GS_i regarding S_t. This influence is equal to the mutual affinity, Aff_{ij}^{IN}, between GS_i and GS_j, which constitutes the contribution made by the IN.

Both the antigenic and the IN contributions are combined for each GS. Finally, the net contributions by all the GSs are aggregated to yield an affinity distribution across all the classes (as detailed in Algorithm 2). The class having the maximal affinity is chosen as the prediction.

Algorithm 2 Combining Antigenic and IN contributions for prediction

Input: S_{GS} (GS set), IN (Network of GSs), q (Number of classes), S_t (Test Data)
Output: Class label of S_t

```
Prediction[q] ← 0 for each GS_i in S_GS do
    for each GS_j in IN do
        Prediction_j = Prediction_j + Aff_ij^IN * Aff_i^Ag
return argmax(Prediction)
```

The method used in Algorithm 2 adds a degree of confidence to the contribution made by the IN in proportion to the antigenic contribution. If Aff_i^{Ag} itself is low, the impact made by each of the GSs via the IN (Aff_{ij}^{IN}) over the prediction of the test sample will be curtailed by weighing it down. This interlinking of contributions allows a holistic method of prediction as opposed to prediction by individual GSs in isolation. The overall classification process discussed so far primarily utilises just one set of GSs. However, as also suggested in [60], one may employ multiple sets of GSs to enhance

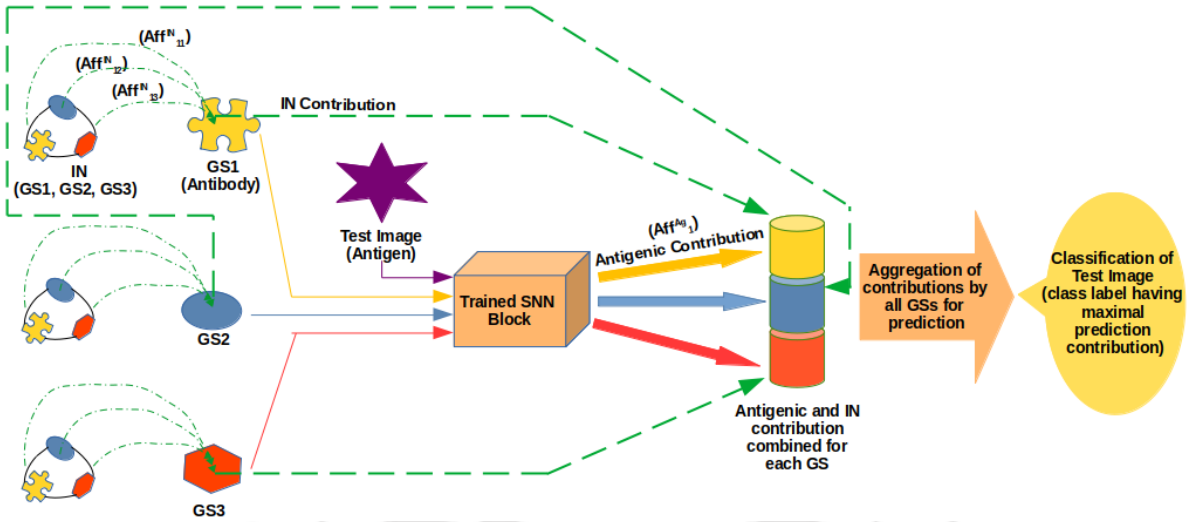


Figure 3.3: Classification of a Test sample using both Antigenic & IN contributions

the decision making process. We therefore propose an additional method as briefed in Algorithm 3 to achieve this. Multiple sets of GSs are found by using *stimulations*. This is followed by execution of Algorithm 1 for each set of GSs so as to yield its corresponding inference. These inferences are combined via majority voting to yield final prediction.

Algorithm 3 Multi-Class Classification using Immune Network augmented Siamese Neural Network with multiple Gold Standard Sets and Majority Voting

Input: Abs (Training Dataset), q (Number of classes), Ag (Test Data)
Output: Class label of Ag

```
Train SNN()
GoldStandardsSet ← SNN(Abs)
CombinedPrediction[q] ← 0
for each GoldStandards_i in GoldStandardsSet do
    ClassIndex ← Algorithm 1(Abs, Ag)
    CombinedPrediction[ClassIndex] ← CombinedPrediction[ClassIndex] +1
return argmax(CombinedPrediction)
```

3.3 Experimental Analyses and Results

Due to high dimensionality and multiple levels of features, images prove to be an apt candidate for the rigorous testing of the approach discussed herein. We have thus, tested the algorithms by conducting experiments using the MNIST [37], the Fashion-MNIST (FMNIST) [38] and the Kuzushiji-MNIST (KMNIST) datasets [39]. These datasets contain 60,000 training and 10,000 testing samples as images of size 28x28 pixels each. The MNIST dataset consists of images of handwritten digits from 0 to 9. The FMNIST dataset consists of ten different classes of images of clothing items. The KMNIST dataset consists of images of characters from the Japanese syllabary, Hiragana, spread across 10 classes. These three datasets differ in terms of features and extent of overlap among classes, presenting diverse scenarios and making classification a rigorous exercise. The experimental setup was realised on *Jupyter* [65], a web-based computation platform. Architec-

tures involving ANNs, were realised using *Keras* [66] and *Tensorflow* [67] frameworks running on a computer with an *Intel CORE i5* processor with 16GB RAM. We have used a CNN based SNN, as depicted in Figure 3.1, to act as the base model. As can be observed, the SNN has two CNN branches connected by a distance layer. The distance layer is used to find the Euclidean distance between the *embeddings* generated from the two CNNs. This distance was used to train the SNN using contrastive loss minimisation method. The configuration of each CNN branch is given below:

- A *Convolutional* layer having four filters of size 5x5 using *tanh* as the activation function
- An *Average Pooling* layer of size 2x2
- A *Convolutional* layer having sixteen filters of size 5x5 with *tanh* as the activation function
- An *Average Pooling* layer of size 2x2
- A *Flatten* layer
- A *Dense* layer having ten neurons which use *tanh* as the activation function

As explained in section 3.2.1, we began the implementation of our approach by creating a *positive pair* and a *negative pair* for each sample belonging to each of the ten classes within the MNIST dataset. These pairs were used to train the SNN. We then followed the next step (section 3.2.2) to obtain the set of GSs, which comprised one representative training sample image per class. This was followed by the creation of a 2-dimensional matrix in the form of a Python list structure that stored the affinities amongst the GSs thereby acting as the IN (section 3.2.3). This matrix, along with the set of GSs and the SNN was then used to classify images from the test set of the MNIST dataset based on the explanation provided in section 3.2.4. In order to analyse the significance of the major components of our proposed method, ablation studies were conducted which have been discussed in subsequent sections. After gathering enough empirical justification for the components involved in our approach, we conducted rigorous experimentation to compare our approach with existing methods, viz. the popular kNN based approach [62, 63, 64, 79] and the recent approach of random sampling of class representatives [60, 79]. In all the experiments, the underlying SNN was trained for 100 epochs. All the methods were tested against 800 samples per class. The details of the experiments and the results obtained are discussed in the following sections.

3.3.1 Comparison of GS Selection Methods

GSs play a major role in representing the underlying knowledge about the dataset. Hence, the strategy used for selection of a GS directly impacts the classification accuracy. For conducting ablation studies of this impact, experiments were carried out using the following four methods for selection of a GS:

- **RS**: *Random Sampling* as in [60, 79]
- **RMV**: Random selection of multiple GS sets, followed by classification based on *Majority Voting* amongst the sets, as suggested in [60]
- **ST**: Using only *Stimulations* (equation (3.2))
- **STSU**: Using *Stimulations* and *Suppressions* (equations (3.2) and (3.3))

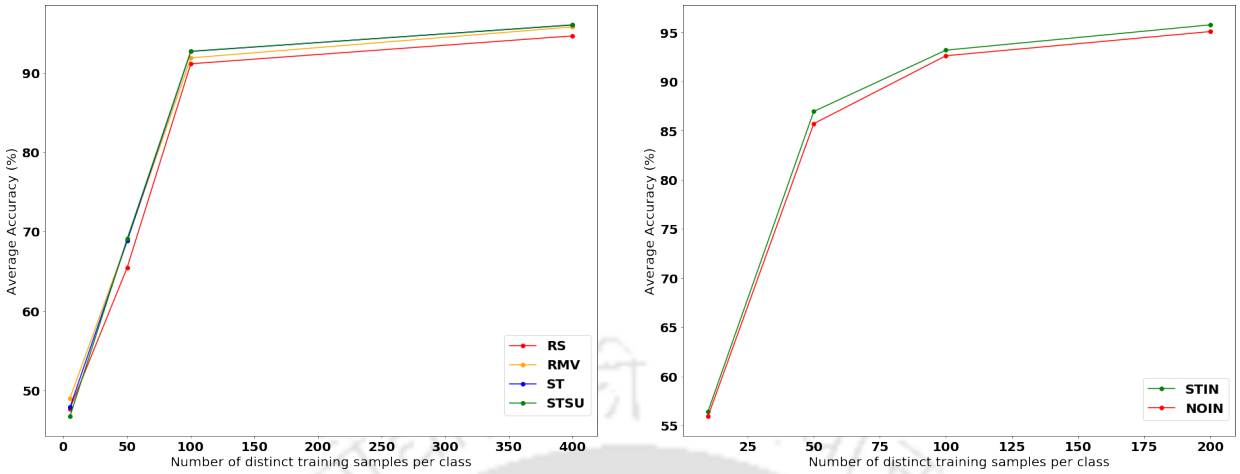
The multi-class classification of the MNIST dataset was performed with the set(s) of GSs obtained using each of the above methods. In order to closely analyse the impacts of the modes of selection of the GS on classification accuracy, we conducted the experiments by removing the IN and hence its own impact. To achieve this, we ignored the portion in Algorithm 2 where IN is used to update the $Prediction_i$ vector (line 3) and also omitted Aff_{ij}^{IN} from line 4. The SNN was initially trained separately using four discrete training dataset sizes made by using 5, 50, 100 and 400 distinct training samples per class. These four SNNs were then used individually for classification in conjunction with each of the above GS selection methods amounting to sixteen different experiments. In order to take care of any possible stochastic variations, we conducted each of these sixteen experiments, five times (leading to eighty experiments in total) and compared their average testing accuracies. In addition, five GS sets were used for voting in the RMV method. As can be observed in Figure 4.12(a), the accuracies in case of ST and STSU methods were found to be consistently more than those of the RS and RMV. This can be attributed to the improved characterisation of features of a class by the corresponding GS selected using ST and STSU methods. This effect becomes more pronounced with increase in the size of the training dataset, thus validating the proposed stimulation-suppression mechanism for GS selection.

3.3.2 Impact of IN during Inferencing

The significance of the involvement of IN for inference was analysed through an ablative procedure. This involved comparing the classification accuracies when both *Stimulations* and the *IN* were used (STIN method) with the case when the *IN* was not used (NOIN method). The procedure discussed in section 3.3.1 was used to remove the IN. In both the cases, the underlying SNN was initially trained separately using four discrete training dataset sizes made by using 10, 50, 100 and 200 distinct training samples per class. These four SNNs were then used individually for classification both under STIN and NOIN setting, leading to eight experiments. Further, each of these experiments were averaged across five repeated separate executions to yield average testing accuracies. It may be seen from Figure 3.4(b) that there is a consistent increase in accuracy across different training sizes in case of STIN against NOIN, highlighting the role of IN.

3.3.3 Comparison of STIN with kNN based approach for Multi-Class Classification

Since kNN is often employed with SNN to perform classification tasks [62, 63, 64, 79], we have compared a kNN based approach (KNN) with our approach (STIN). We have used the same trained SNN for classifying the MNIST dataset while using KNN and STIN methods. After the SNN was trained, the test sample was compared with all the training samples for clustering under the KNN approach. The KNN algorithm works by finding the K nearest neighbours (training samples) to the test sample. The 'nearness' was quantified by using the SNN as the underlying distance metric. The test samples and the sample being compared were fed as inputs to the SNN. The class of the test sample is then determined by considering the majority vote of the neighbours. As the simplest case, requiring least computations, a cluster of size one was used for KNN. Five experiments were performed for both KNN and STIN approaches using a training set size of fifty samples per class. Table 3.1 shows the results of these experiments in terms of classification accuracy and inference time. As can be seen, on average, the STIN outperforms the KNN approach in terms of accuracy. While the increase in accuracy seems marginal, the inference time taken by the STIN approach suggests it to be the clear winner. The average speedup factor of STIN over the KNN approach is around 74. This speedup can be attributed to the *parameterised* nature of STIN where the GSs are



(a) GS selection methods

(b) STIN vs NOIN

Figure 3.4: Classification Accuracies with different training dataset sizes using the MNIST dataset across (a) Different GS Selection Methods, (b) GS based method with IN (STIN) and without IN (NOIN)

already distilled out *a priori* and are directly used as characteristics of the corresponding classes during inferencing. This saves time as there is no need to compare the test sample with each sample of a class during inference.

Table 3.1: Classification Accuracies and Inference Times using the MNIST dataset for kNN-based (KNN) and combined GS-IN based (STIN) methods (max accuracy and min time boldfaced)

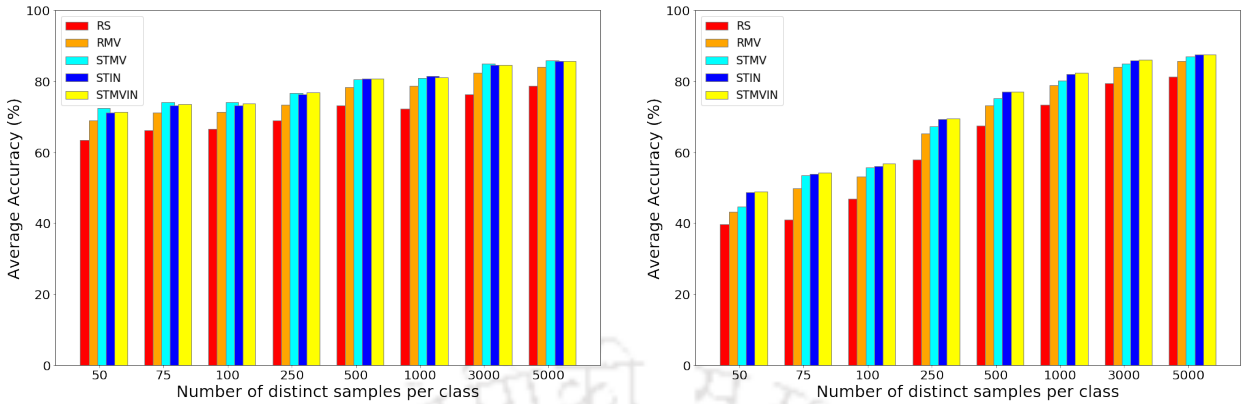
Accuracy (%)		**Inference Time (seconds)**	
KNN	STIN	KNN	STIN
87.57	**88.73**	2477.61	**33.45**
88.21	**88.49**	2722.31	**38.15**
87.0	**87.03**	2463.94	**33.11**
85.99	85.7	2492.50	**33.54**
87.71	**88.88**	2581.44	**33.15**

3.3.4 Comparison of STIN and its variants with Random Sampling based approaches for Multi-Class Classification

In addition to RS, RMV and STIN, we introduced the following methods for experimentation:

- **STMV**: Multiple GS sets are found by using *Stimulations*. The inference is then drawn for a test sample based on *Majority Voting* by the GS sets. IN has not been used here.
- **STMVIN**: Using IN for each GS set along with STMV

In order to test the performance consistency of our proposed methods (STIN, STMV and STMVIN) against existing methods (RS and RMV), we conducted experiments across three different datasets: MNIST, FMNIST and KMNIST. Each of these datasets has different levels of complexity in terms of higher-level features and extent of overlap among classes. For all the methods, the underlying SNN



(a) FMNIST (b) KMNIST

Figure 3.5: Classification Accuracies with different training dataset sizes across different methods using (a) FMNIST dataset (b) KMNIST dataset

was trained separately using eight discrete training dataset sizes made by using 50, 75, 100, 250, 500, 1000, 3000 and 5000 distinct samples per class. These eight SNNs were then used individually for classification using each of the five methods, amounting to forty different experiments. Further to discourage any empirical analysis emerging from stochastic reasons, we conducted each of the experiments twenty times to find the average testing accuracies and inference times (leading to 800 experiments for each of the three datasets). Additionally, five GS sets were used for voting in RMV, STMV and STMVIN methods.

Table 3.2: Classification Accuracies and Inference Times with different training dataset sizes across different methods using the MNIST dataset (max accuracy and min time(s) boldfaced)

Samples	**Accuracy (%)**					**Time (seconds)**				
/class	RS	RMV	STIN	STMV	STMVIN	RS	RMV	STIN	STMV	STMVIN
50	83.30	85.73	**87.68**	86.26	**87.60**	**51.73**	263.26	**52.27**	260.75	266.18
75	87.43	90.99	**92.29**	91.51	**92.16**	**53.71**	264.158	**54.42**	261.96	265.59
100	90.07	92.63	**93.57**	93.01	**93.62**	**54.82**	263.84	**55.34**	263.51	267.57
250	93.42	94.50	**95.34**	94.61	**95.35**	**52.53**	264.55	**53.06**	266.44	268.13
500	95.10	95.69	**96.47**	95.89	**96.52**	**52.54**	264.00	**53.52**	261.2	268.23
1000	96.29	96.73	**97.21**	96.85	**97.21**	**53.01**	262.997	**53.55**	265.82	268.23
3000	97.02	97.60	**97.93**	97.66	**97.92**	**52.91**	261.28	**53.67**	258.57	264.13
5000	97.22	98.00	**98.151**	98.03	**98.154**	**52.55**	262.26	**52.76**	332.59	277.90

The accuracy and inference time (for all the test samples) obtained across the experiments using the three datasets are shown in Table 3.2 and Figure 3.5, leading to the following significant observations:

- Accuracies across all the methods increase with increase in training dataset size. However, for a given dataset size, the accuracies generally decrease across datasets MNIST, FMNIST and KMNIST; in that order. This implies the relatively increasing complexities of these datasets.
- For all the datasets, our proposed methods (STIN, STMV, STMVIN) consistently outperform the existing methods (RS, RMV). This can be observed from Table 3.2 where the top two

Table 3.3: Class-wise comparison of Precision, Recall and F-score for different methods using the MNIST dataset (max values boldfaced for each class)

Class	F-score			Precision			Recall		
	RMV	STIN	STMVIN	RMV	STIN	STMVIN	RMV	STIN	STMVIN
0	**0.985**	**0.985**	**0.985**	**0.988**	0.985	0.985	0.981	**0.985**	**0.985**
1	0.988	**0.989**	0.988	**0.991**	0.987	0.986	0.984	**0.991**	**0.9912**
2	0.971	**0.976**	**0.976**	0.953	0.960	**0.961**	0.990	**0.992**	**0.992**
3	0.965	0.971	**0.972**	0.960	0.971	**0.972**	0.970	0.971	**0.972**
4	0.972	**0.976**	**0.976**	0.987	**0.988**	**0.988**	0.958	**0.965**	**0.965**
5	0.965	0.966	**0.9666**	**0.960**	0.955	0.957	0.971	**0.976**	**0.976**
6	**0.979**	**0.979**	**0.979**	**0.986**	0.983	0.983	0.972	**0.975**	**0.975**
7	0.967	0.972	**0.974**	**0.984**	0.980	0.981	0.951	0.965	**0.967**
8	0.951	**0.958**	**0.958**	0.928	**0.948**	**0.948**	**0.974**	0.968	0.968
9	0.939	0.944	**0.945**	0.949	0.960	**0.961**	0.929	**0.930**	**0.930**

accuracy values for each dataset size (row) of the MNIST dataset are boldfaced. Figure 3.5 also depicts the same trend for FMNIST and KMNIST datasets. This highlights the utility of the proposed mechanisms of *stimulations* (over random sampling [60, 79]) and IN.

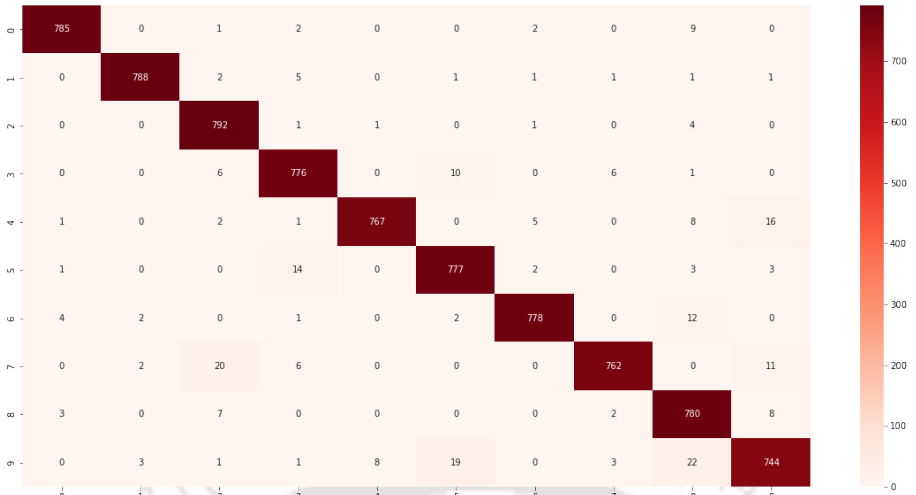
- STMVIN generally performs better than STIN because of the involvement of more than one GS sets. STMVIN also outperforms STMV due to the use of an IN by the former.

- STIN and RS take the least time (speedup of almost 5 over both RMV and STMVIN) as shown in Table 3.2. Experiments conducted using FMNIST and KMNIST datasets also showed similar trends. It may thus, be inferred that when time is a crucial factor, it is best to opt for STIN. STMVIN can be used otherwise, when accuracies are to be increased even further.

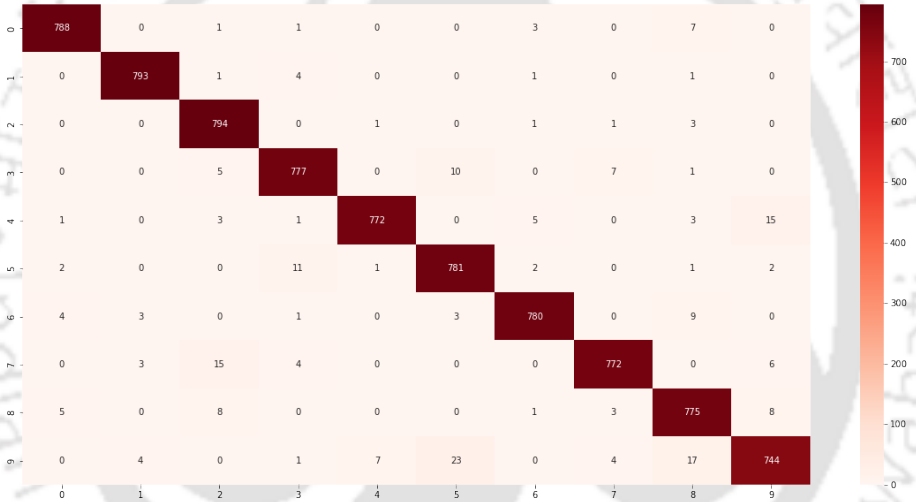
- In few cases, STIN manages to yield a higher accuracy than STMVIN (with lesser inference time). It can also be noted that STIN generally outperforms STMV. These observations strongly suggest the towering impact of the use of *stimulations* and the IN on classification performance as compared to the conventional majority voting mechanism.

3.3.5 Comparison over Multiple Metrics for Multi-Class Classification

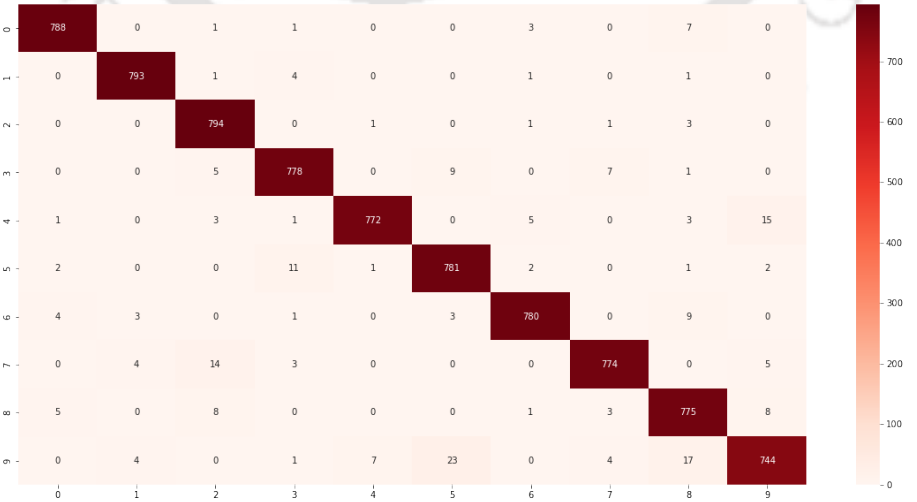
In addition to accuracy and inference time, multi-class classifiers can be analysed in detail by using precision, recall and F-score metrics. F-score is generally the preferred metric for comparing classifiers since it merges both precision and recall. Table 3.3 provides a class-wise comparison of these metrics for RMV, STIN and STMVIN methods for the MNIST dataset. The underlying SNN for all these methods was trained using 900 distinct samples per class from the training dataset. The F-score tends to be the highest for STMVIN in most cases (values boldfaced), highlighting its consistent performance gain, even at the class level.



(a) RMV



(b) STIN



(c) STMVIN

Figure 3.6: Confusion matrices of classwise classification performance using (a) RMV (b) STIN and (c) STMVIN

Figure 3.6 also shares the confusion matrices obtained over the aforementioned experiments. It can be observed that the STMVIN method gives the best class-wise classification performance.

3.4 Chapter Summary

This contribution proposes a method to enhance the performance of an SNN for multi-class classification through an immuno-inspired method. The experimental analyses strongly support the proposal, showing a consistent increment in accuracy with a speedup of up to five times in a dataset agnostic manner. By utilising an SNN as an affinity function for an AIS, this work attempts to bridge both the domains. In addition, the proposed methods (STIN and STMVIN) provide a choice for higher accuracy upon the investment of more time, thereby contributing to an *Anytime* algorithm [80].

We have used a multiplication-based strategy to accommodate the contribution made by the Immune Network during classification. Considering the promising preliminary results, this strategy needs to be investigated further which can also be enhanced by involving parallel *stimulations* and *suppressions*. Since SNNs can also tackle scalability towards new classes, we can widen the scope of our proposed methods in terms of adding new classes and samples. As the approach has been tested with small sized datasets, its usage can also be explored for low-resource devices.

From the perspective of the overall work described in this thesis, the work described in this chapter establishes a bridge between connectionist (SNN) and evolutionary (AIS) ML techniques. This bridge proves that a synergistic amalgamation of the computational efficiency of SNNs with the comprehensive searching capability of the evolutionary ML algorithms, can enhance the overall performance of connectionist algorithms.

Since such a bridge can perform with less data and low computational resources, it opens up new opportunities for hosting ML algorithms onboard resource-constrained devices. Contributions described in subsequent chapters draw inspirations from this and endeavour to enable onboard ML operations on resource-constrained devices.

4

ChaoticImmuneNet: A Chaos-driven immuno-inspired Neural Network paradigm for Embodied Intelligence in Resource-Constrained Devices

While the earlier chapters explored a CPS and an efficient and synergistic ML method, this chapter attempts to embed such ML techniques IN a resource-constrained Edge Device (ED) that could eventually constitute a CPS. This contribution marks the first attempt under this thesis for realising ML, onboard resource-constrained EDs. This chapter reviews the existing solutions and highlights the gap in the efforts made by others towards enabling onboard adaptation in resource-constrained EDs. In this work, an Embodied Artificial Intelligence (EAI) based method has been proposed and then deployed on a real robot, acting as an ED, so as to test its efficacy and practical viability.

With the increased availability of huge datasets and high computational power, the usage of DNNs for ML has percolated into several domains, including Computer Vision, Natural Language Processing, Speech Recognition, etc. [81]. However, the usage of DNN-based approaches in real-world settings as in the case of EDs is rife with challenges. Issues like concept drift, covariate shift, catastrophic forgetting, etc. [82] make it difficult for pre-trained DNNs to deliver the same performance onboard an ED as they would, given a static dataset. This necessitates the usage of EAI with EDs wherein the ED interacts with its environment for better adaptation/learning. However, EDs such as *tiny robots* [83] have limited resources in terms of sensors, actuators, sampling and computational power due to their dependence on microcontrollers as opposed to CPUs and GPUs [84, 85]. This restrains the EDs from relying much on computationally intensive techniques such as DNNs, for realising EAI. Despite these challenges, with the introduction of the Industry 4.0 paradigm [86], application areas such as warehouses and automated retail stores are bound to delegate a major chunk of their processes to automated robots/EDs. Techniques such as Transfer Learning (TL) [87] and Fine Tuning [88] support EAI [89] for EDs but often involve state-of-the-art DNN models that require large storage and target data for onboard adaptation. While SNNs [90] can perform well with less training data, they are primarily used for similarity detection tasks. SNNs are thus, open for novel transformation methods to suit classification or recognition tasks.

Keeping these concerns in view, we propose an EAI method that leverages an SNN along with concepts from AIS [91] and Chaos Theory [17] for a multi-class classification task. Coupled with TL, this method utilises the latent space of an SNN, for adapting to a target environment, without

disturbing the already learnt weights. This is achieved by chaotic hypermutation of the *paratopes* derived from the embeddings of the SNN. Hypermutation of *paratopes* provides a lightweight alternative to gradient-based techniques for conducting model training onboard the ED. This method, nicknamed, *ChaoticImmuneNet*, can be used effectively in resource-constrained EDs and can facilitate EAI in the real world. In contrast to conventional Chaotic Neural Networks [92], which often require the usage of chaotic neurons with specialised neuromorphic model architecture and hardware, this method blends concepts from Chaos theory with Immune Network Theory to allow a more generic approach of utilising chaotic dynamics with a conventional DNN model to boost the classification performance of the latter. To examine its efficacy, we have tested *ChaoticImmuneNet* in an IoT based warehouse setting in which a robot exhibits EAI by learning the identification of pallets in the warehouse.
The main contributions of this work are listed below:

1. A resource-efficient EAI method that can be used to conduct on-the-fly training of models derived from a DNN, onboard an ED.

2. A novel technique to combine an SNN and an AIS, using hypermutation, for multi-class classification.

3. Implementation of a prototype IoT-based warehouse setup for identifying pallets using a mobile robot, henceforth referred to as a robot, and a group of microcontrollers to prove the practical viability of *ChaoticImmuneNet*.

4.1 Related Work

Since the proposed work knits together various domains of EAI, Artificial Neural Networks (ANN) and resource-constrained devices a brief discussion of these domains follows.

4.1.1 Embodied Artificial Intelligence

While AI has gained widespread usage and acceptance across domains, there still exist fundamental differences in views on the characterisation and acquisition of intelligence. EAI stands out as one such notion that shuns static data-based intelligence. EAI emerges from the interaction of an agent with its environment through *sensorimotor* activity [25]. It draws its inspiration directly from the knowledge acquisition process of human infants that involves processes such as incremental learning, physical interaction, exploration, etc. Owing to the failure of conventional AI approaches in physical environments, the need for EAI has been strongly advocated [93]. Giulia *et al.* [23] have shown that off-the-shelf state-of-the-art models perform poorly when deployed in real-world environments. They adapted the model to the target environment by human-assisted sampling of target data. Such manual adaptation is, however, time-consuming and mars the essence of EAI. Roy *et al.* [94] advocated the use of continual learning for ML models. Bartolozzi *et al.* [95] proposed neuromorphic EAI for robots. They emphasized that conventional AI methods are not suited for deployment in dynamic natural settings due to their large data and computation requirements. Their approach requires changes to be made in the underlying hardware and the associated algorithms. While methods involving simulators have been leveraged for EAI [96, 97, 98], they in no case, bridge the *reality gap* [99]. Other EAI approaches [100, 101, 102, 103, 104] often involve sharing of information between the server and the ED. This waives off the burden of onboard computation from the ED but adds latency and communication overheads to the overall process. Pasquela *et al.* [105] leveraged real-world data for training a DNN using fine-tuning, which still

requires substantial computational power to propagate updates across the DNN. Neuman [83] *et al.* advocated the role of onboard learning for resource-constrained *tiny robots* [83] due to constraints on latency and communication with such robots. They alluded to a trade-off between model complexity and practical usability for such robots. State-of-the-art EAI, thus, demands resource-efficient methods that can perform onboard adaptation in real-world environments with limited data and computational overheads.

4.1.2 Artificial Neural Networks on Resource-Constrained Edge Devices

Using an ANN with EDs has its own challenges. including low memory availability and computational power, battery-based power supply and low data sampling capability, etc. The usage of ANNs over EDs has to be judicious so as to accommodate both the ML operations and the underlying control system of the ED. Fan *et al.* [106] have used a low-power FPGA chip to run a Convolutional Neural Network (CNN) for detecting the stock status of store shelves. They relied on external devices for model training. Bechtel *et al.* [107, 108] used CNN to control an RC car. They utilised an onboard camera and a locally created dataset for offline model training. Curtin *et al.* [109] classified wildlife images using a pre-trained CNN deployed on a Raspberry Pi. Chen *et al.* [110] have used an ANN (YOLO) based pet monitoring system with an advanced Raspberry Pi (version 4) using an onboard camera. They have not performed on-device training of the model since the YOLO ANN [111] is difficult to train on a Raspberry Pi. In most of the efforts discussed so far, the concept of onboard training of the model seems grossly missing. In addition, the accuracy of the model suffers when local data samples are used for retraining [108, 109].

4.2 Preliminaries

This section briefly describes the key techniques used in this contribution.

4.2.1 Transfer Learning with Siamese Neural Networks

Although ANNs work well as generic function approximators, they require a substantial amount of well-processed training data. Unfortunately, generating large and high-quality datasets is not always feasible and affordable. Researchers thus, opt for TL[87] which allows ANNs to share learning across domains. If an ANN does not have substantial data for training in the target domain, one can utilise a source dataset, D_S, available for a related domain to train the ANN initially. This pre-trained ANN can then be further tweaked with a small target dataset, D_T, to suit the target domain. Coarse features learnt from D_S, are thus, transferred for use in conjunction with D_T, using TL [87]. Combining TL with an SNN [90] can cater to EDs with limited computing power and memory. This is so because SNNs require less data and can thus, deliver good performance even with a small architecture.

A typical SNN consists of two branches of ANNs (Figure 4.1), each of which converts its respective input to a low-dimensional embedding. These embeddings are then merged through operators like subtraction, concatenation, etc. This whole structure is then optimised using a loss function to yield a value that reflects the similarity of the inputs. If the inputs are similar, the SNN outputs a low value and vice versa, otherwise.

Researchers have used TL with SNNs by utilising several types of pre-trained networks as feature extractors for the target SNN. These techniques include pre-trained CNNs for plant disease classification [112], a pre-trained VGG-16 ANN for face recognition [113], a pre-trained Inception-V3 ANN for geochemical anomaly identification [114], a pre-trained MobileNet-v3 ANN for face

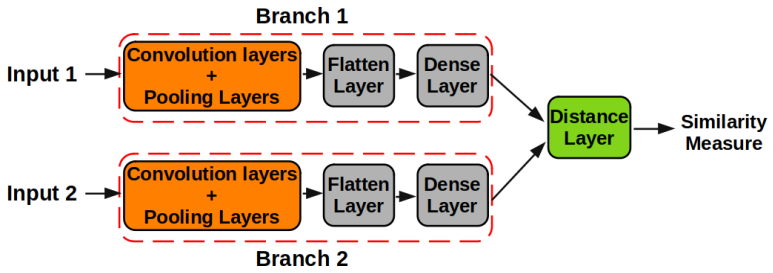


Figure 4.1: A Siamese Neural Network composed of two CNN branches

recognition [115], a pre-trained VGG16 ANN for tracking veneers in a factory, etc. A combination of TL and SNN was found useful for resource-constrained devices [115], for low data regime [112] and for enhancing the classification performance [112]. However, most of these techniques rely on large ANN architectures that form the branches of the SNN. The resulting SNN thus becomes too large to be accommodated in resource-constrained EDs. Thus, exploring the usage of TL with smaller SNN architectures for use in resource-constrained EDs is required. Since SNNs deal with similarity measures, there is also a need to convert the same into the classification space, to meet the demand of our multi-class classification task. In the next section, we inspect immuno-inspired techniques to achieve the same.

4.2.2 The Immune Network Theory

The human BIS acts as an excellent recogniser of antigens invading the human body. It largely relies on the T- and B-cells that operate together to accomplish the recognition task. Both the B-cells and the antigens have protruding structures on their surface called *paratopes* and *epitopes*, respectively (Figure 4.2). For an antigen to be recognised, a *paratope* of an antibody, Ab_1, needs to bind to an *epitope* of the antigen, *Ag*. This can happen only if the concerned *paratopes* and the *epitopes* are morphologically complementary. This binding action results in the suppression of *Ag* and the stimulation of Ab_1. Since Ab_1 gets rewarded, it proliferates by cloning proportionate to its stimulation. The antibodies of type Ab_2, unable to bind well with *Ag* are suppressed, thereby decreasing in population. Thus, the presence of an antigen makes the set of antibodies compete with each other, yielding a high population of the type of antibodies that perform better, eventually containing the antigenic attack. Jerne [40] extrapolated this theory by highlighting that antibodies also *stimulate/suppress* themselves, forming an IN, to regulate their own population. This helps in creating an effective repertoire of antibodies against a known antigen.

Antibodies (Ab_3) also use somatic hypermutation [116] to adapt their *paratopes* for better detection of antigenic *epitopes*. Hypermutation involves selective and controlled mutation of the *paratopes*. This process, called *affinity maturation* [116], plays a crucial role in refining the quality of antibodies in a guided manner.

Techniques inspired by a BIS have been widely used in the computational domain [91, 117], constituting an area referred to as *Artificial Immune Systems*. An AIS quantifies the binding among antibodies and antigens via *affinity* functions. The ability of an AIS to evolve similarity spaces for classification and the low resource requirement of an SNN, make the integration of the two an ideal system for multi-class classification in resource-constrained devices. Pandey and Nair [48] have conducted a preliminary analysis of using an AIS with an SNN for multi-class classification. They utilised GSs as class representatives with and without an IN (STIN and NOIN, respectively) for classification. However, their GSs were limited in terms of class representation capability due

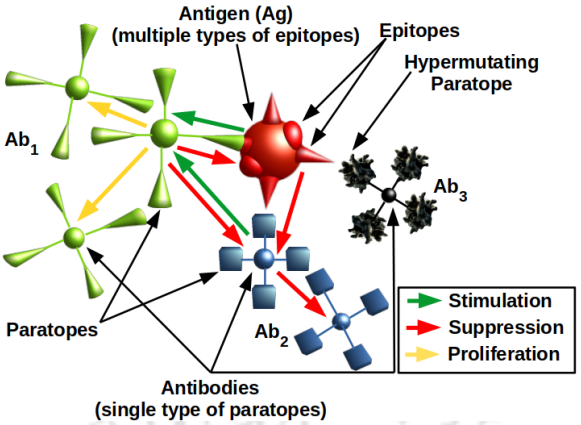


Figure 4.2: Antigen Recognition in a BIS: Antibodies interact with the Antigen and among themselves to recognise the antigen

to the lack of adaptation to the local dataset. They have not explored the *affinity maturation* mechanism of the BIS and have also not targeted resource-constrained EDs. This opens up ample scope to explore the combination of concepts from an AIS with an SNN to realize EAI.

4.2.3 The Logistic Map

Optimisation techniques involve the traversal of unknown search spaces. For resource-constrained devices, one needs to be parsimonious while undertaking such traversals, by maximising the area swept under each traversal. Chaotic traversal is one such technique that maximises the coverage of the search space for a given number of iterations. Chaos is known for evading issues such as local minima, early convergence and slow progress towards the optimal solution [17, 18, 19, 20]. The logistic map equation (equation 4.1) [118] depicts a rich set of behaviours, one of which is chaotic.

$$x = r.x.(1 - x) \tag{4.1}$$

For growth rates r greater than 3.7, the values of x obtained from the equation 4.1 exhibit chaos with momentary non-chaotic behaviour. From the context of an AIS, the logistic map equation allows a method to leverage chaos for hypermutation of *paratopes*. The work described herein makes a novel attempt at amalgamating concepts from AISs, Chaos Theory and SNNs for efficient EAI in resource-constrained EDs.

4.3 Methodology

The proposed method, termed *ChaoticImmuneNet*, describes an approach to realise EAI for a multi-class classification task over an ED. It derives its name from the underlying mechanism that combines *Chaos* and the *Hypermutation* mechanism of an AIS with a *Neural Network*. Every step in this method strives to achieve better classification performance and caters to EDs that have limited resources and data access. Figure 4.3 depicts the manner in which *ChaoticImmuneNet* can be used for Embodied Learning.

The method includes the following major steps:

- Training the source SNN (SNN_S) on D_S

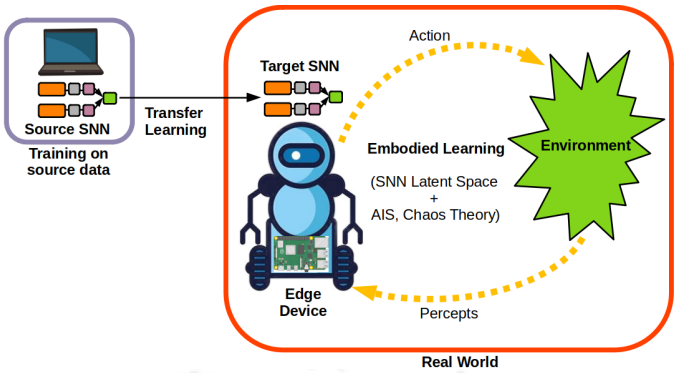


Figure 4.3: *ChaoticImmuneNet*: An embodied learning method that utilises Transfer learning, AIS, SNN and Chaos Theory to facilitate onboard learning on resource-constrained devices

- Transferring learning from SNN_S to a partially frozen target SNN (SNN_T) deployed on an ED.
- Transforming SNN_T into a multi-class classification model using concepts from both AIS and Chaos Theory to extract a population of embeddings. This involves conducting Embodied Learning by retraining the model using D_T collected online through the ED and utilising embeddings using Hypermutation.
- Using SNN_T along with the embeddings for multi-class classification

Subsequent sections describe each of these steps in detail.

4.3.1 Training the Source SNN$_S$

SNNs with simple architectures are known to perform fairly well with limited data [119]. Since the methodology is targeted towards resource-constrained and limited-data scenarios, we have used an SNN as the base architecture. However, in order to decrease the burden of learning on the ED, the source SNN, SNN_S, is pre-trained on a remote system using an already available D_S which contains data similar to the one perceived by the ED. Training of the SNN_S is carried out by pairing every sample within D_S with two other samples, randomly choosing one from the same class and the other from a different class, so as to yield corresponding positive (similar) and negative (dissimilar) pairs. A positive pair $[S_j^i, S_k^i]_{k \neq j}$ includes samples of the same class, i, while a negative pair $[S_j^i, S_n^m]_{i \neq m}$, includes samples from different classes. Each of the pairs is assigned a label, viz., 0 for a positive and 1 for a negative pair. All such pairs, along with their respective labels, are used to train the SNN_S. The following contrastive loss function is used for training:

$$\mathcal{L} = (1 - l).D^2 + l.max((\tau - D), 0)^2 \tag{4.2}$$

where l denotes the label of the pair, D is the Euclidean distance between the embeddings of the SNN and τ, a hyperparameter, is the distance margin beyond which the samples of a pair are considered dissimilar. A trained SNN_S infers the similarity/dissimilarity of the sample pair fed as input through a numeric value between 0 and 1. A value closer to 0 implies stronger similarity and vice versa.

4.3.2 Transfer Learning from Source SNN_S to Target SNN_T

A novel method to utilise TL for SNN has been used to maximise the performance gain on the ED. A target SNN, SNN_T, identical in structure to SNN_S is deployed on the ED. Figure 4.4 depicts the method of TL from SNN_S to SNN_T. The transfer process starts by copying all the weights of the trained SNN_S to SNN_T. Thereafter, the weights of the layers of both the branches of SNN_T, up till the *Flatten* layer, are frozen. With the partial structure frozen, SNN_T is then trained on the target dataset, D_T, collected by the ED. This approach gives an added performance advantage to SNN_T in terms of the already learnt similarity features by SNN_S. This is not possible in the commonly used case of TL for SNN, where the source network is a CNN (further analysis shown in Figure 4.9). Samples and the corresponding labels for D_T are obtained by the ED.

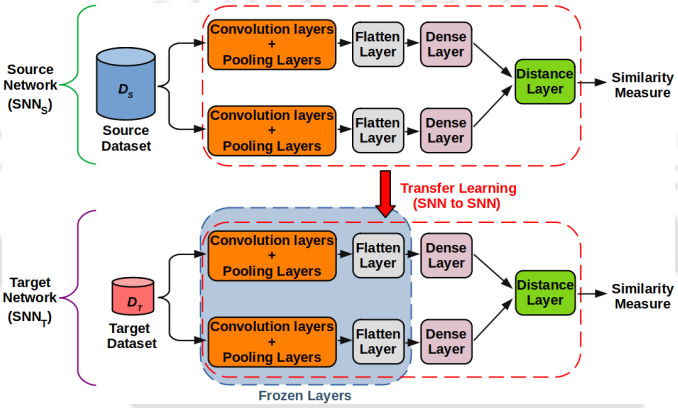


Figure 4.4: Transfer Learning for SNN: Using an SNN as the source network

4.3.3 Transforming SNN_T into a Multi-Class Classification Model

After training SNN_T, one needs to transform its similarity space to a classification space. This was achieved in an earlier contribution of ours [48] by involving GSs and an IN. A GS is a class representative used while inferencing. It provides class-specific information not captured directly by the SNN and is obtained by distilling samples of each class. The SNN compares a test sample with the GS of each class to arrive at the final inference. A GS obtained in the STIN method of [48], is the sample that is least distant from the rest, in its own class. The GS creation process begins by drawing concepts from AIS by treating the training data as antibodies and the test data as antigens. A set of GSs, S_{GS}, comprising one representative from each class, is obtained via stimulative interactions between antibodies. *Stimulations* are based on the affinity between samples of each class. The affinity measure, Aff^i_{jk}, between two samples S^i_j and S^i_k of a class i, is based on the output of the SNN with the corresponding samples input to it. The net stimulation, ST^i_j, for the sample S^i_j is calculated based on the equation (4.3) below.

$$ST^i_j = \sum_{k=1}^{Q} Aff^i_{jk} \tag{4.3}$$

where Q is the total number of samples in class i and $j \neq k$.

The sample S^i_j, having the highest value of ST^i_j is chosen as the Gold Standard, GS^i, for class i as per equation (4.4).

$$GS^i = \arg\max_j (ST^i_j) \tag{4.4}$$

The GS^i obtained through equation (4.4), as also used in [48], does not yet hold the capability to further generalise to other samples of class i. Equation (4.5) provides the net distance of the sample, chosen as the GS of class i from all other samples of the same class. A GS is a sample that has the least value of D_{GS^i} for class i. Thus, D_{GS^i} inherently governs the choice of a sample to be considered as the GS for representing that class:

$$D_{GS^i} = \sum_{k=1}^{Q} dist(S_k^i, GS^i) \tag{4.5}$$

It may be observed from equation (4.5) that the distance, D_{GS^i}, of GS^i with all other samples, S_k^i, of class i, cannot be reduced further. This is so because both the parameters GS^i and S_k^i in the function, $dist$, are actual existing samples whose features cannot be changed. An S_{GS} obtained using this method ([48]) is fixed to a specific location in the feature space, thus limiting the scope of the algorithm to accommodate features of other samples within the respective classes of each GS. It also renders S_{GS} less useful when faced with a new target dataset, having a different feature space. What is thus required is a method to evolve a GS that is closer to all other samples of its corresponding class. To tackle this, we propose a novel approach that augments each GS with an evolving population of filters, F, indirectly facilitating the evolution of the GS, as shown in equation 4.6. The evolution process of the filters is explained later in detail.

$$D_{GS^i} = \sum_{k=1}^{Q} dist(S_k^i, (GS^i.F)) \tag{4.6}$$

We utilise hypermutation to evolve F, causing the function, $dist$, to change as per the rest of the samples of the corresponding class or the samples of a new dataset. Due to the presence of F, the new function, $dist$, in equation (4.6) is not a constant anymore. Thus, hypermutation indirectly allows the movement of the features of S_{GS} across a target feature space, facilitating the generalisation of S_{GS} and allowing better recognition of the target dataset. Equation (4.6) forms the basis of the proposed hypermutation-based approach followed for onboard training whose in-depth details follow.

Owing to the high dimensionality of the features, direct hypermutation over S_{GS} is not feasible, especially in resource-constrained devices. To tackle this issue, we have leveraged a lower dimensional latent space formed by the embeddings of the SNN_T when fed with S_{GS}. The embeddings are obtained by extracting the activations of the hidden *Dense* Layer of SNN_T. Hypermutating such embeddings of the fixed S_{GS}, as part of the SNN (along with the connections to other components of SNN_T), is bound to be computationally expensive and could also disturb the rest of the architecture of the SNN_T.

We, therefore, use the proposed workaround for affinity maturation by extracting and then combining the embeddings externally with a population of *filters*, F, as detailed in Algorithm 4 and depicted in Figure 4.5. A filter is a vector having the same dimension as that of an embedding. The embeddings E_{GS} and E_{tr} corresponding to a GS and a training sample, respectively, are extracted from the SNN_T. A random population of filters, F, is generated. Each filter $F_i \in F$ is combined separately with the E_{GS} and the E_{tr} to yield Par_{GS} acting as *paratope* and Epi_{tr} acting as the *epitope*, respectively. Thus, for a given filter, we have as many *paratopes* as there are GSs and as many *epitopes* as the number of training samples. Therefore each *paratope* and each *epitope* is mapped to the corresponding class or training sample, respectively.

$E_{S_{GS}}$ is the set of embeddings obtained from all the GSs in S_{GS}. Similarly, $E_{D_{tr}}$ is the set of embeddings for all training samples in the dataset D_{tr}.

The set of *paratopes* ($Par_{S_{GS}}$) and *epitopes* ($Epi_{D_{tr}}$) are now evolved, indirectly, by hypermutating the population of filters. This evolution is guided by the performance of each filter on the training dataset D_{tr}. The performance of each filter, $F_i \in F$ is found by assessing its inference accuracy over D_{tr}. This process involves finding the Euclidean distances between all the *paratopes* the *epitopes*, generated using F_i. For each training sample, S_t, the *paratope* $Par_{S_{GS}i}$ closest to its corresponding *epitope* is found. The class i corresponding to $Par_{S_{GS}i}$ is inferred as the label of S_t and compared with the actual class label of S_t. The accuracies of all the filters are thus found. We have used an elitist mechanism of selection by keeping top-performing filters (F_{top}) as it is and hypermutated the rest of them (F_{bottom}). Hypermutation was achieved by using a chaotic mechanism. Specific regions within the filters in F_{bottom}, were identified by using equation 4.7 and mutated using equation 4.8.

$$Rw = \sqrt{\frac{1}{num_f} \sum_{i=1}^{num_f} (F_i - \bar{F})^2}. \tag{4.7}$$

$$F_i = (r.F_i.(1 - F_i)).(1 - Rw) + F_i.Rw \tag{4.8}$$

Having chosen a filter F_i for mutation, equations 4.7 and 4.8 together allow F_i to exhibit dual mutability behaviour for its constituent regions. During the mutation process, for all the filters in F, there will exist regions that are similar and thus, have not been exploited much. These regions ought to be chosen for mutation while giving less priority to the already diverse regions. Equation 4.7 captures this relation in terms of the vector Rw, representing rewards for each region based on the standard deviation across all the num_f number of filters. $\bar{F}$ is the average vector of all the filters in F. A region having a higher standard deviation across F has a higher reward and vice versa. Equation 4.8 uses Rw to allow F_i to depict either a chaotic behaviour (former part of the equation) over a region or retain its original form (latter part of the equation). The former part of this equation uses the logistic map equation (equation 4.1) with r kept equal to 3.8 so as to exhibit chaotic behaviour. Equation 4.8 thus, mutates F_i only at regions of less diversity while ensuring that the other regions remain untouched. It may be noted that the rewards are dynamic and change as the population of filters evolves.

The hypermutated F_{bottom} filters along with F_{top} are again used to make inferences over D_{tr} and sorted according to the newly obtained accuracies. This process is repeated for M mutation steps. Since filters and embeddings together yield the *paratope*, this process captures the essence of using affinity maturation to evolve the *paratopes* in a guided manner. The chaotic behaviour is chosen for hypermutation as it quickly maximises the coverage in the given search space of filters [17]. This allows efficient evolution of filters on the ED. It may be noted that direct perturbation of embeddings would have meant a change in the parameters of the preceding layers, hence disturbing the learnt information within the network. Thus, instead of directly using the embeddings for hypermutation, we combined them with filters to obtain the *paratopes*. At the end of the affinity maturation process, a refined form of F is obtained that when plugged with S_{GS} to give *paratopes*, yields a more generalised representation for each class of the target dataset.

Algorithm 4 Affinity Maturation of GS using *Paratopes* (and *Epitopes*)

Input: D_{tr} (Training Dataset), num_f (Number of Filters), M (Number of Mutations), C (Number of classes)

Output: $E_{S_{GS}}.F, E_{D_{tr}}.F$ (*Paratopes* and *Epitopes*)

$r \leftarrow 3.8$,iteration$\leftarrow 0$
Train $SNN()$
S_{GS}(GoldStandards) $\leftarrow SNN(\mathrm{D}_{tr})$
$E_{S_{GS}}$, $E_{D_{tr}}$ $\leftarrow$SNN(S_{GS}, D_{tr}) /* Extracting embeddings from both branches of SNN using S_{GS} and D_{tr} */
$F \leftarrow$[Random Distribution]$_{num_f.d}$ /* d is the dimension of each embedding */
while *iteration* $<$ M **do**
    $accuracies_F \leftarrow []$
    for *each Filter* F_i *in* F **do**
        $Par_{S_{GS}} \leftarrow \mathrm{E}_{S_{GS}}.F_i$ /* *Paratopes* of all GS */
        $Epi_{D_{tr}} \leftarrow \mathrm{E}_{D_{tr}}.F_i$ /* *Epitopes* of all training samples */
        $Distance_{|D_{tr}|.C} \leftarrow getDistance(Par_{S_{GS}}, Epi_{D_{tr}})$
        /* get distance of each *epitope* from *paratope* of each class */
        for *each Sample* S_i *in* D_{tr} **do**
            inference $\leftarrow index(min(Distance_i))$
            /* Find the GS having least distance of its *paratope* with *epitope* of S_i */
        $accuracies_F$.append($accuracy_{F_i}$)
    $F \leftarrow sortByAccuracy$(F,accuracies$_F$) /* rearrange filters in increasing order of accuracy over D_{tr} */
    $Rw = StandardDeviation(F)$
    for *first* n F_i *in* F **do**
        $F_i = (r.F_i.(1\text{-}F_i)).(1\text{-}Rw) + F_i.Rw$ /* Reward based Chaotic Hypermutation of the n least accurate filters */
    iteration $\leftarrow iteration + 1$
return $E_{S_{GS}}.F, E_{D_{tr}}.F$

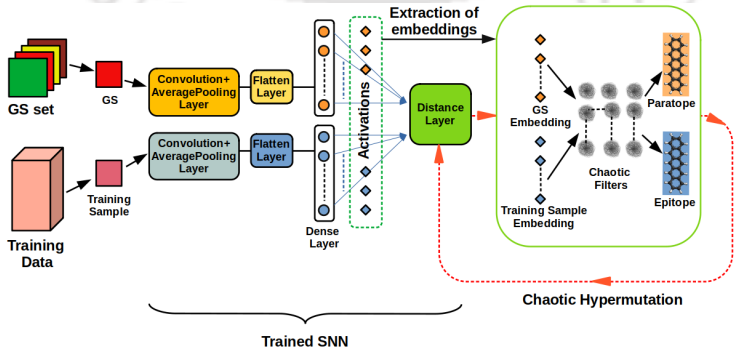


Figure 4.5: Extracting Embeddings from a Siamese Neural Network for *Paratope/Epitope* formation (GS set obtained from D_{tr} and SNN)

4.3.4 Using *Paratopes* and *Epitopes* for inference

The inference process for classifying a test sample (S_{te}) utilises the evolved filters, as summarised in algorithm 5. The embeddings of S_{GS} and S_{te} are extracted out as $E_{S_{GS}}$ and E_{te}, respectively. A population of top num_{top} filters is picked as the new F. Each filter $F_i \in F$ is combined with $E_{S_{GS}}$ and E_{te} to yield a set of *paratopes*, $Par_{S_{GS}}$, and an *epitope* Epi_{te}, respectively. The distance between each of the *paratopes* in $Par_{S_{GS}}$ and Epi_{te} for every F_i is calculated and added together under the corresponding GS classes, denoted by their indices. The index of the *paratope* having the least distance with Epi_{te} is inferred as the label of S_{te}.

Algorithm 5 Using Refined *Paratopes* and *Epitopes* for inference

Input: F (Filter), S_{GS} (GS set), S_{te} (Test sample), num_{top} (Number of top Filters), C (Number of Classes)
Output: Label of S_{te}

$Distance \leftarrow [0]_{1.C}$
$E_{GS}, E_{te} \leftarrow SNN(S_{GS}, S_{te})$
`/* Extracting embeddings from both branches of SNN using` S_{GS} `and` S_{te} `*/`
$F \leftarrow$F[$len(F)$ - num_{top}:len(F)]
`/* Using top` num_f `filters */`
for *each Filter* F_i *in* F **do**
    $Par_{S_{GS}} \leftarrow \mathrm{E}_{GS}.F_i$
    $Epi_{te} \leftarrow \mathrm{E}_{te}.F_i$
    $Distance \leftarrow Distance + getDistance(Par_{S_{GS}}, Epi_{te})$
label $\leftarrow index(min(Distance))$
`/* Find the GS having least distance of its` *paratope* `with` *epitope* `of` S_{te} `*/`
return label

It is worth noting that due to the resource constraint of the ED, most portions of SNN_T are frozen during onboard training with D_T, to allow Embodied Learning. This severely restrains the model from adapting to the local data. The chaotic *paratopes*, used in the proposed method, circumvent this issue by compensating the learning via an alternate efficient method that propagates parameter updates only through the latent space of the SNN instead of the whole architecture. It may be noted that the *ChaoticImmuneNet* should not be looked upon as a Chaotic Neural Network. Unlike existing Chaotic Neural Networks [92] that often require fundamental changes to the model and the underlying hardware, *ChaoticImmuneNet* utilises a conventional DNN model. It uses Chaos theory to alter the behaviour of an Immune Network, which consequently enhances the performance of a conventional DNN.

4.4 Experiments and Results

Experiments were performed using both software and hardware, to analyse the performance and efficacy of *ChaoticImmuneNet*. This section commences with detailed analyses of the multi-class classification performance of *ChaoticImmuneNet* and its comparison with existing methods and concludes with analyses of experiments conducted using a real robot, acting as an ED, in a warehouse prototype environment. The image classification performance of *ChaoticImmuneNet* was tested with three different datasets: the MNIST [37], the *Local* and the KMNIST datasets [39], each having 10 classes. Ancillary details of the datasets can be found in section 4.5
Currently available datasets such as MNIST, contain images that are fairly noise-free. However,

real images taken by autonomous mobile EDs such as mobile robots can be highly noisy and can degrade the performance of an algorithm. Since there is limited availability of such vision-based datasets, we have created one, referred to herein, as the *Local* dataset, comprising 2000 image samples of digits, similar to MNIST. These images were captured by a camera onboard a robot. It may be noted that both the choice of the dataset and the classification task were kept simple to validate the preliminary findings of *ChaoticImmuneNet*. However, the three datasets used, viz., MNIST, the *Local* and the KMNIST, together provided a diverse landscape in terms of higher-level features (Figure 4.6) and additional noise to rigorously test the proposed *ChaoticImmuneNet* method. As can be observed from Figure 4.6, all three datasets differ in the underlying feature distribution. These distributions were obtained by utilising the t-distributed Stochastic Neighbor Embedding (t-SNE) method that plots the data distribution based on relative similarities of data points. While the distribution corresponding to samples of each class label in the case of the MNIST and the KMNIST dataset is more clustered, the *Local* dataset exhibits a more scattered distribution. The relative position of the distribution of classes also varies across the three datasets. This emphasises the diverse landscape these datasets provide to test the proposed method.

The CNN and SNN architectures were realised using Keras [66] and Tensorflow [67]. All the source networks used for TL were implemented and trained on an Intel CORE i5-based computer with 8GB RAM. After the transfer, the complete algorithm, including the SNN, was run on a Raspberry Pi 3 (Model B V 1.2 with 1 GB RAM) based robot. The limited computational, storage and power capabilities of this robot served as an apt scenario to test the utility of the *ChaoticImmuneNet* method for resource-constrained EDs. We used Jupyter [65], a browser-based platform in our implementation.

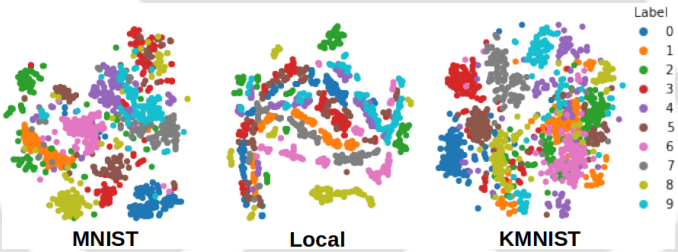


Figure 4.6: t-SNE based feature spaces depicting the difference in the feature distribution of the datasets used

4.4.1 Transfer Learning: CNN to SNN vs SNN to SNN

The difference in the features of the source and the target data, chosen from the datasets referred to in Figure 4.6, mandates the usage of TL. However, using a CNN as the source network to perform TL for a target SNN may negatively impact the transfer due to the difference in the underlying features. To investigate this impact we conducted experiments based on the following two TL approaches ('»' indicates the transfer of weights from a source network (left) to the target network (right)):

- **CNN»SNN**: The architectures of the source CNN and target SNN are shown in Figure 4.7. This TL approach has been depicted in Figure 4.8. The source network (CNN) was trained for the image classification task on the D_S. The weights of the trained source network were copied until the *Flatten* layer to both the branches of the target network (SNN). Thereafter, weights of the layers of both the branches of the SNN, until the *Flatten* layer, were frozen.

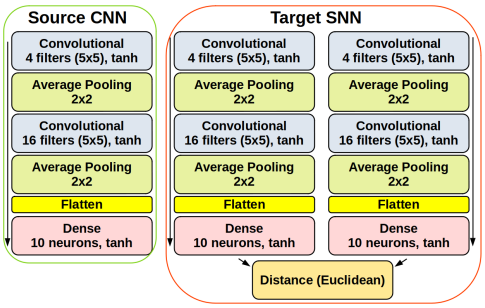


Figure 4.7: Network architectures used in the CNN»SNN approach

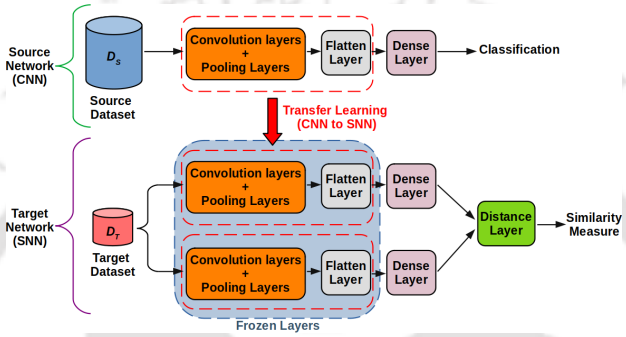


Figure 4.8: Transfer Learning for SNN: Using a CNN as the source network

With the transferred weights frozen, the SNN was then trained on the D_T. This is the conventional TL approach for SNN.

- **SNN»SNN**: This is the proposed TL approach for SNN, that uses an SNN as the source network, as explained in section 4.3.2 (Figure 4.4).

The number of trainable parameters (2570) of the target SNN remains the same in both cases, due to the freezing of weights, consequently keeping them computationally at par for the target ED. Experiments were conducted with different datasets (MNIST, *Local*, KMNIST) as the D_T using the same D_S (MNIST). For both approaches, the source network was trained separately using four different training dataset sizes, viz., 50, 100, 500 and 1000 samples per class. As for the target network, 100 samples per class were used for both training and testing. An experiment comprised transferring a learnt model from a source network trained on the D_S to a target network using one of the approaches, after which the latter network was trained on the D_T. The experiment concluded by testing the resultant target network on the test data of the D_T. Five such experiments were conducted for each training dataset size of D_S for both approaches to obtain the average accuracy values. This amounted to a total of 120 experiments (including all the datasets). The target network in both the methods was employed for a multi-class classification task over the D_T using the NOIN method [48].

The experimental results shown in Figure 4.9 clearly depict the gain in accuracy through the proposed SNN»SNN transfer method across all the D_T for various dataset sizes. Since in all the experiments, MNIST was used as the D_S, the highest accuracy was obtained when the same dataset was used as the D_T. The TL approach also gave a good boost to the accuracy when the *Local* dataset was used as the D_T. This is so because the D_S, namely MNIST and the D_T are closely related. Interestingly, when KMNIST was used as the D_T in the SNN»SNN approach, though the

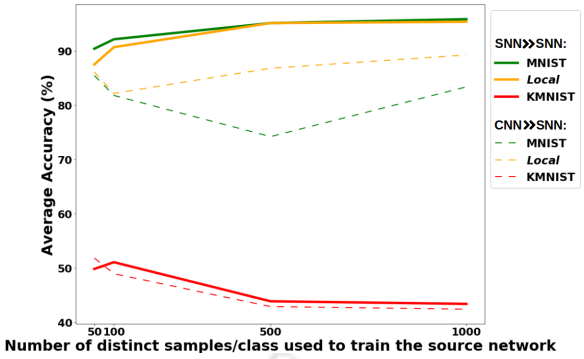


Figure 4.9: Accuracies of different Transfer Learning methods for SNN with different D_T (MNIST as D_S in all cases): SNN as the source, instead of a CNN, works better for the target SNN

overall accuracy is low, it still manages to outperform the CNN»CNN approach.

The overall gains in accuracy in the case of all the SNN»SNN-based experiments may be attributed to the similarity of the source and target networks. The usage of SNN as the source network allows the relevant similarity information to be passed to the target SNN, giving a head start for exploring the similarity space in the target network.

4.4.2 Impact of different configurations of the *ChaoticImmuneNet* steps on its performance

ChaoticImmuneNet involves multiple steps, each having a substantial impact on the performance of the algorithm. Ablation studies on the following steps were thus, carried out to gain further insights regarding their impact on the accuracy of the algorithm:

- **Using TL** : This step involved the transfer of a pre-trained source SNN, trained on an MNIST dataset with 1000 samples per class.
- **Training the target SNN onboard the robot** $_{(D_T)}$: The partially frozen target SNN, as detailed in section 4.4.1, was trained on the D_T.
- **Creating the GSs** $_{(D_T)}$: GSs obtained from D_T were used to leverage the target SNN for classification.

Icons have been used to annotate each step for brevity. NOIN approach [48] was used to utilise SNN for classification. 60 samples per class were used for training the model onboard the robot. For a particular configuration of onboard training and GSs creation step, samples were taken from either the MNIST, the *Local* dataset or a *Mix* dataset (pairing samples from MNIST dataset with that of the *Local* dataset). 17 experiments, using different configurations of the above three steps were performed and the testing was done onboard the robot with 100 samples per class of the *Local* dataset. Each experiment was performed three times and the averages of the individual test accuracies, in each case was found. Details of the same are portrayed in Table 4.1.

Each row of Table 4.1, represents one of the 17 experiments, reflecting a particular configuration of choices made for each of the steps. For instance, the first row shows the accuracy for the experiment in which the GSs were obtained using MNIST as the D_T. In addition, TL was not used; nor was training performed on the robot. Likewise, the last row represents a case where TL was performed by training the source network on the MNIST dataset. The target network onboard

Table 4.1: Ablation studies: Test Accuracy of the target Network based on different configurations of *ChaoticImmuneNet* steps

S.No.	Configuration of Steps used for Training the Target Network	Test Accuracy (%) on the *Local* dataset
1.	$(MNIST)$	12.9
2.	$(Local)$	29.2
3.	$(MNIST)$, $(MNIST)$	14.43
4.	$(MNIST)$, $(Local)$	20.73
5.	$(Local)$, $(MNIST)$	15.80
6.	(Mix), $(MNIST)$	22.39
7.	(Mix), $(Local)$	38.66
8.	$(Local)$, $(Local)$	66.63
9.	, $(MNIST)$	20.2
10.	, $(Local)$	42.7
11.	, $(MNIST)$, $(MNIST)$	20.7
12.	, $(MNIST)$, $(Local)$	45.53
13.	, (Mix), $(MNIST)$	65.13
14.	, $(Local)$, $(MNIST)$	22.96
15.	, (Mix), $(Local)$	68.93
16.	, (Mix), $(Local)$	70.3
17.	, $(Local)$, $(Local)$	87.93

the robot was then trained using the *Local* dataset which was also used to create the GSs. The importance of each step is highlighted through the following observations:

- The importance of can be realised by comparing the first and the ninth row. The use of the weights of the pre-trained source network boosted the accuracy by 7.3 per cent in the latter row.

- Using (D_T) does not always yield better accuracies (third row) unless the datasets used in the training pipeline are directly or indirectly related to the test dataset. This can be observed by comparing the third and the sixth row where the latter yields an accuracy boost of 7.96 per cent. In both cases, has not been used, the MNIST dataset was used for creating the GSs while the *Local* dataset was used for testing. Higher accuracy is achieved in the sixth row due to the presence of both of these datasets in the algorithm's pipeline (via the *Mix* dataset), thus capturing the relation between the two different datasets in (D_T) step. Further, when all the datasets used across the algorithm (on the robot) are in accordance to the testing dataset, a substantial boost in accuracy can be observed (eighth row). It may be noted here that this increment is obtained without using .

- (D_T) is also crucial for enhancing the accuracy of the algorithm. This can be observed from the 14th and the last row. As the testing was done on the *Local* dataset, choosing the same dataset for GS creation in the latter case yielded an accuracy boost of 64.97 per cent.

- Compatibility of all the choices made across these steps is crucial to maintain a good performance. It can be observed from Table 4.1, showing accuracies as low as 12.9 and as high as 87.93.

- From the context of practical applicability, it is pertinent to note the large difference in accuracy between the ninth and the 17th row. In both cases, the same source network, trained on the MNIST dataset was used. The configuration in the ninth row, however, provides poor performance when tested with a semantically similar dataset (*Local*) in the presence of minor aberrations (lighting, gradient, etc.). Thus, it is crucial that the robot uses onboard sampling for training the target network, followed by the steps $_{(D_T)}$ and $_{(D_T)}$.

4.4.3 *ChaoticImmuneNet* versus other methods

This section compares *ChaoticImmuneNet* with other multi-class classification methods when performed onboard an ED. Since on-device training is expensive, only the last *Dense* layer(s) of the target networks were made trainable in all these cases. Experiments were conducted using the following methods:

- **CNN»CNN**
- **CNN»SNN**
- **SNN»STIN**: NOIN [48] method used for classification. Unlike in [48], we partially froze the target network to run it on an ED and utilised TL from a source SNN.
- ***ChaoticImmuneNet***: Used top nine filters for inference.

The network architectures used for all four methods were the same as discussed in section 4.4.1, with corresponding layers frozen in the target network. The MNIST dataset was used as D_S. The *Local* dataset was used as D_T, on a Raspberry Pi. To obtain better statistical insights, in all the experiments, the source network was trained separately using four different D_S sizes, viz., 100, 500, 2000 and 6000 samples per class. D_T had a fixed size of 100 training and testing samples per class. The average accuracies and inference times obtained by conducting 5 experiments for each method (totalling 80 experiments) were found, as shown in Table 4.2 (highest values have been highlighted). The inference time indicates the total time required for predicting labels of all the test samples. As observed from Table 4.2, CNN»SNN delivers better accuracies than CNN»CNN due to the discriminative power of the SNN. This observation strongly supports the choice of an SNN for *ChaoticImmuneNet*. SNN»STIN is able to further boost the accuracy by utilising immuno-inspired plugins on top of the target SNN. *ChaoticImmuneNet* pushes the accuracies even further by utilising hypermutation-based onboard training. The CNN»CNN method takes the lead when it comes to inference time but at the cost of low accuracy. Time is saved here as the method does not compute similarities in the embeddings nor use immuno-inspired plugins. However, a good trade-off between accuracy and inference time is achieved in the case of *ChaoticImmuneNet* that delivers the best accuracy across all training sizes of D_S, within a reasonable time.

It is pertinent to note that to store the S_{GS} specific information, *ChaoticImmuneNet* uses only 190 floating-point values (10 embeddings and nine filters, each of size ten) while the SNN»STIN method used 7,840 floating-point values (10 matrices of size 28x28), a saving of 41 times in the former case. This saving is achieved in all the experiments, irrespective of the size of the training dataset. Thus, while the improvement obtained in terms of accuracy and inference time by using *ChaoticImmuneNet* may seem marginal, the use of the lightweight chaotic *paratopes*, renders *ChaoticImmuneNet* to be a better choice for resource-constrained scenarios such as in case of EDs. While these experiments establish the performance gain of *ChaoticImmuneNet* over other approaches, the practical usability of the former on actual hardware is also to be tested. Owing to the

Table 4.2: Classification Accuracies and Inference Times of *ChaoticImmuneNet* and other Existing Methods (Highest values are emboldened)

Samples /Class	**Accuracy (%)**				**Time (s)**			
(Source Network)	**CNN»» CNN**	**CNN»» SNN**	**SNN»» STIN**	**Chaotic Immune Net**	**CNN»» CNN**	**CNN»» SNN**	**SNN»» STIN**	**Chaotic Immune Net**
100	49.02	83.54	87.52	**87.82**	0.44	**5.89**	4.63	4.67
500	37.70	75.95	91.62	**92.08**	0.32	4.19	**4.77**	**4.77**
2000	51.00	78.04	88.31	**88.88**	0.28	4.16	**4.71**	4.55
6000	57.29	78.45	87.69	**88.24**	0.45	**4.96**	4.81	4.55

resource constraints of hardware, we have not tested other approaches (CNN»CNN, CNN»SNN, SNN»STIN) again on actual hardware. As the relative performance insights observed for the methods analysed in the current section are hardware agnostic, we proceeded with testing only the *ChaoticImmuneNet* method on actual hardware, as discussed in the subsequent section.

4.4.4 Deploying *ChaoticImmuneNet* on a Warehouse Prototype

To cater to fast-paced supply chains, warehouses need to keep real-time checks on the inventory. Pallets used for storage and retrieval of goods, play an important role in mobilising the goods of the inventory [120]. However, mobile pallets are often displaced around the warehouse due to which their manual identification is difficult[121]. This necessitates the use of robots [122]. Robots can navigate autonomously within the warehourse and identify these pallets using technologies like RFID, barcode readers, cameras, etc. [123, 124, 121, 125]. With onboard cameras for resource-constrained devices easily available, vision-based recognition methods in warehouses have gained popularity[126, 125]. Several works focus on camera-based recognition of pallets and other objects in a warehouse [121, 125, 127, 128, 129, 130, 131]. However, they suffer from issues such as infrastructural overheads [125], lack of mobility of the camera [125, 129, 128], absence of onboard training [131], etc. Some of the works also use simulated warehouse environments to test vision-based algorithms [132]. A majority of these methods refrain from using EAI. In this work, we have attempted to address these research gaps and test the efficacy and practical viability of *ChaoticImmuneNet* in the real world, by designing a prototype of a warehouse (Figure 4.13). It comprised pallets numbered 0 to 9 equipped with microcontrollers along with a mobile robot as the resource-constrained ED. This prototype constitutes a CPS including the computational model as the cyber component and the mobile robot and the pallets as the physical components of the CPS. Further details of the prototype can be found in section 4.6.

To begin with, *ChaoticImmuneNet* was used to train a model, with the MNIST and the *Local* datasets as the D_S and D_T, respectively. A robot, carrying this model along with the *Local* dataset, was allowed to autonomously move within the warehouse to identify the numbers on the pallets. During this testing process, when the robot is positioned in front of a pallet, it captures four shots of the digit in front, from different orientations, and predicts the number within each. For every shot captured, a microcontroller equipped with the corresponding pallet provides its actual number to the robot. After visiting all 10 pallets, the robot calculates the overall prediction accuracy for this test round. The onboard model is retrained using the captured images coalesced with the onboard dataset and the robot is made to perform another test round. Four experiments constituting four such rounds each were performed, whose accuracies are depicted in Figure 4.11.

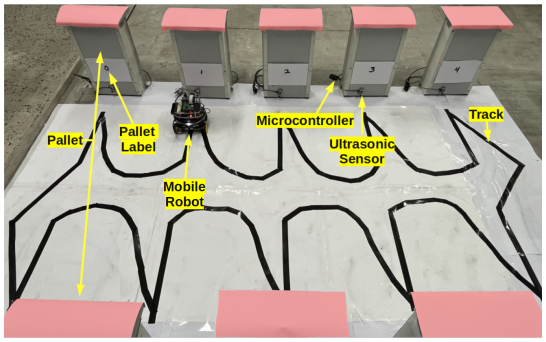


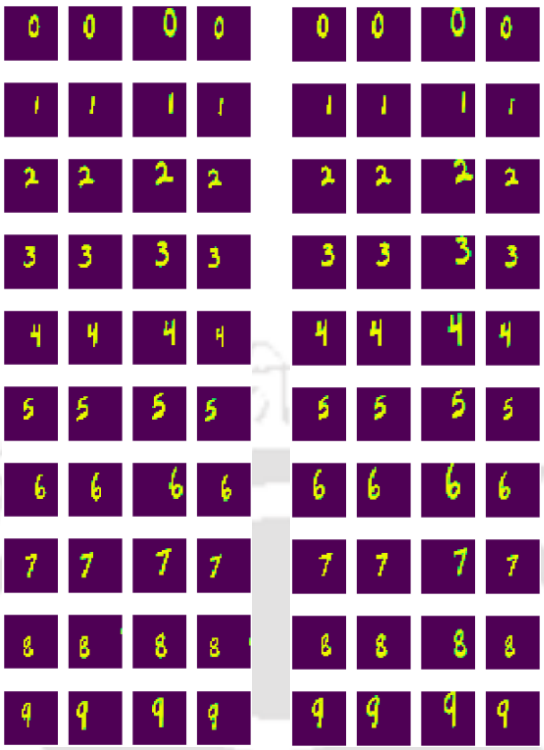
Figure 4.10: Warehouse prototype with the mobile robot and the pallets (full of real-world noise such as varying shadow and lighting)

All the experiments were conducted under different ambient lighting, yielding different but low accuracies for all the initial rounds. However for each of them, *ChaoticImmuneNet* was able to boost and maintain the classification accuracies up to and beyond 90 per cent by the end of the last round. An average increment of 15.625 per cent from the first to the last round can be observed across all the experiments. This consistent increment can be attributed to the onboard training capability of *ChaoticImmuneNet* that may be leveraged further for more rounds to achieve even better accuracies. This real-world experiment thus proves that *ChaoticImmuneNet* can be hosted on resource-constrained devices and adapt well to real-world settings by using EAI.

Figure 4.11: Pallet classification accuracy achieved using a mobile robot across multiple rounds of the warehouse for different experiments: All the accuracy trajectories saturate to a value of above 90% using the *ChaoticImmuneNet* embodied learning method

4.5 Description of the Datasets used

Three datasets have been used in the proposed work: the MNIST [37], the *Local* and the Kuzushiji-MNIST (KMNIST) [39] datasets. Both the MNIST and the KMNIST datasets contain 60,000 training and 10,000 testing samples as images (28x28 pixels). The MNIST dataset comprises images of handwritten digits from 0 to 9 while the KMNIST dataset comprises images of characters from Hiragana (a Japanese syllabary) spread across 10 classes. In order to better analyse the efficacy of the proposed algorithm, we required a dataset that is as close as possible to the real world and is also compact enough to be tested with EDs. Thus, we created a *Local* dataset in the lab.



(a) Training Data (b) Testing Data

Figure 4.12: The *Local* dataset captured via the onboard camera by the mobile robot

environment by using a camera onboard a robot. This dataset consists of images of handwritten digits 0 to 9 that were written on a placard pasted on pallets located within a prototype warehouse setup (Figure 4.13). The robot was made to autonomously move in the warehouse environment and collect image samples and corresponding labels, from its perspective, as detailed in section 4.6. 2000 such samples were split equally into training and testing sets. Each class thus contained 100 samples for training and another 100 for testing.

Figure 4.12 shows some of the samples from the *Local* dataset. Unlike MNIST, the presence of noise, gradients of different hues, shifting/scaling of the digits and jagged edges of the digits can be observed in the *Local* dataset samples. This makes the learning exercise closer to the real world and thus more challenging.

4.6 Details of the Warehouse Prototype

The warehouse prototype setup (Figure 4.13) comprised the following components:

- **Pallets**: Each pallet represented a storage space for items of a specific category as denoted by the label pasted on it.
- **Robot**: The robot is a crucial component of the setup. It was able to steer around the warehouse and perceive objects using an onboard camera. It was used to recognise the pallets using *ChaoticImmuneNet*. It comprised a Firebird V robot [133], a Raspberry Pi and an onboard camera. Figure 4.14 depicts the overall robot. Firebird V is an ATMEGA2560-based robotic research platform. It has two DC geared motors and a castor wheel for differential

motion-based steering. It comes with a large collection of sensors that include line and IR sensors. It supports multiple communication protocols out of which we have used the USB-based communication method to connect it to a Raspberry Pi. The Raspberry Pi 3, Model B V 1.2, was used as the master controller which controls the Firebird's motion using the line sensor. The Raspberry Pi used had 1 GB of RAM and an onboard WiFi module. A camera [134] (PiCam) was plugged into the Raspberry Pi. This allowed the robot to get the camera stream of the warehouse setup to perform the image classification task.

- **Track**: The track of the robot in a warehouse plays an important role in guiding the locomotion of the robot towards the pallets. In accordance with the arrangement of pallets, as shown in Figure 4.13, we chose such a track that allowed the robot to reach every pallet (with the camera facing the pallet) using a line following mechanism. Thus, once the robot is kept on track, it kept on moving forward without having to turn back to reach other pallets. This ensured that the robot reached every pallet as it moved within the warehouse.

- **Microcontroller and Ultrasonic Sensor**: While following the track, the robot stopped upon sensing a pallet by using its IR sensor. Each of the pallets was equipped with a microcontroller that had an ultrasonic sensor attached to it. Whenever the robot stopped in front of a pallet, the corresponding microcontroller sensed the robot's presence using the ultrasonic sensor. The microcontroller on the pallet communicated the number or label of the associated pallet to the robot and other microcontrollers, so as to make them aware of the same pasted on the pallet. This served as the ground truth for the robot during the training phase. Communication among the robot and the pallets was achieved using WiFi and MQTT [135]. Additionally, the subsequent pallet's microcontroller enabled its ultrasonic sensor to start sensing for robot's presence, while the current pallet disabled it. This sequential enabling of ultrasonic sensors was implemented to remove the interference caused by them when used concurrently. The NodeMCU kit [136] was used as the microcontroller. It is an ESP8266-based open-source firmware and development kit that has a WiFi module, CPU, RAM and GPIO pins for connecting peripherals. Due to its low cost and low power consumption, it is a reasonable choice to IoTize the warehouse setup. The microcontroller was programmed using the Arduino IDE [137]. This IoT setting enabled the robot to get feedback from its environment so as to exhibit Embodied Learning. It may be noted that the labels provided by the microcontrollers could be changed according to the items associated with the pallets without reprogramming the robot.

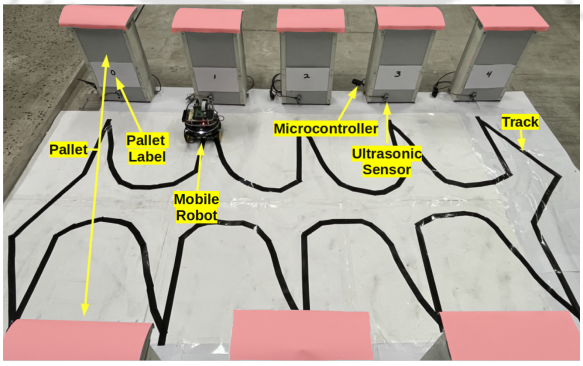


Figure 4.13: Warehouse prototype with the mobile robot and the pallets

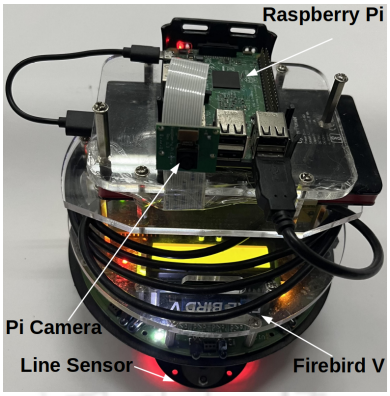


Figure 4.14: Firebird V: The mobile robot used in the experiments

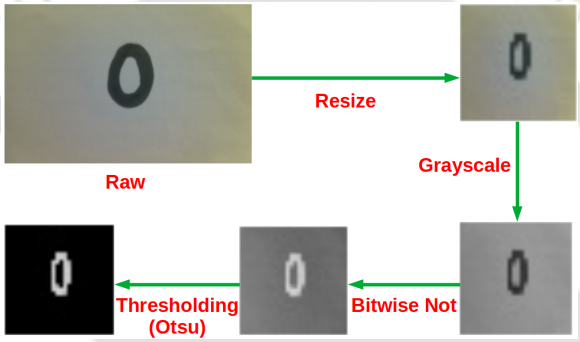


Figure 4.15: Processing steps for the raw image captured by the mobile robot

The experiment commenced by positioning the robot at a distance from the first pallet (labelled *0*). In the absence of an obstacle, the robot started following the black track beneath. On encountering a pallet, the robot stopped and captured an image via PiCam. Since such an image could contain environmental noise, it was processed using the steps detailed in Figure 4.15. The processing converted the noisy raw image into a clean binary image which is then normalised into values between 0 and 1. The robot then deflected sideways to obtain another image. Four such images were obtained from each pallet in a single round of the warehouse. Thus, for each round, the robot sampled forty images with corresponding labels. The *ChaoticImmuneNet* method was used to make inferences out of these forty images. Since *ChaoticImmuneNet* involved training the target SNN onboard the robot, the same was trained using the *Local* dataset and stored on the robot for repeated future access. This saved the robot from the burden of retraining the target SNN at the beginning of each round of the warehouse. For the subsequent round, the SNN was retrained after merging the newly collected set of forty samples with the already available data. The newly collected data could also involve samples captured under drastically different lighting and orientations. In order to reduce the impact of such outliers, it was combined with the older data. This process was performed for four rounds, after which, the robot learned to autonomously identify the labels on the pallet. This marked the end of an experiment. Since in the end, the robot learnt to identify all the labels pasted on the pallets by itself, all the microcontrollers were removed as they were no longer needed to communicate the label to the robot. The appendix of this thesis (and the footnote below[1]) shares a link to a video containing a schematic animation along with the

[1] Video Link

recording of an experiment conducted using the *ChaoticImmuneNet* method on a mobile robot in a prototype warehouse setting.

4.7 Chapter Summary

This contribution proposes *ChaoticImmuneNet*, an EAI method for resource-constrained EDs. *ChaoticImmuneNet* stands out from the existing ML models by combining the latent space of the underlying SNN with AIS and Chaos Theory. This allows minimisation of compute-intensive gradient-based training thereby allowing the ED to learn from the real-world data under an embodied setup by interacting with its environment. The proposed choice of SNN over CNN for *ChaoticImmuneNet* enhances the classification performance, as shown in the experiments. Pivoting the training method on embeddings offers a storage saving of around 41 times as compared to other methods. The usage of embeddings also allows the onboard training to be invariant to the size of the DNN, thereby also making it useful for large networks and resource-constrained devices. In addition, *ChaoticImmuneNet* delivers competent accuracies and inference times when compared to existing onboard training methods. The whole setup is hardware and sensor agnostic and thus, can be used in conjunction with a wide variety of EDs required for automation.

ChaoticImmuneNet introduces a bridge between resource-contained devices and EAI. The efficacy of *ChaoticImmuneNet* is proven by its successful usage on real resource-constrained hardware. In addition to warehouse automation, *ChaoticImmuneNet* may also be used in other resource-constrained real-time settings such as drone-based surveillance, precision agriculture, etc. As *paratopes* provide a cheaper optimisation method, the impact of combining *paratopes* of different layers over the algorithm's performance needs further investigation. Implementing *ChaoticImmuneNet* in tandem with parallel hardware and acceleration facilities provided by resource-rich devices is an area that could also be explored.

While this contribution shares an onboard learning method, there still exists a notable gradient-based compute-intensive portion that is performed onboard the resource-constrained ED. subsequent contribution, described in the next chapter, strives to address and accommodate this issue as one of its primary objectives.

5

GADANN: A Virtual-to-Real Knowledge Transfer & Adaptation method for Edge Devices

The ML technique portrayed in the previous chapter leveraged both evolutionary and connectionist techniques for usage onboard the EDs. The work described in this chapter takes this forward and completely eliminates the involvement of gradient-based optimisation for onboard ML operations. In order to remove the dependence on gradient-based methods, the method discussed in this contribution relies on simulated data and domain adaptation that compensates for the absence of onboard gradient-based optimisation.

The fall in hardware costs and the form factors of EDs has increased their presence across several fields including manufacturing, smart grids, healthcare, agriculture, wearables, etc. [138]. EDs provide automation support by allowing local data acquisition and analysis, thereby minimising their dependence on a central server [139]. However, owing to their low computational and memory resources, utilising EDs for real-time decision-making is a challenging task. The noisy and unpredictable nature of real-world data further aggravates this issue. While ML approaches such as DNNs can tackle noisy data, they rely on a substantial amount of data being available *a priori*, for training. Capturing and labelling such a huge amount of data via EDs is both costly and time-consuming. This has given rise to the usage of simulators for realising environments close to real-world settings faced by EDs. Generating data on such a simulated environment hosted on a resource-rich *Cloud* is easy and also allows faster setup of the experimental conditions. But using the simulated data to train an ML model for an ED deployed in the real world suffers from the *domain shift* [41] issue. A model well-trained on the simulated data will perform poorly when faced with real-world data, rife with aberrations and noise. This has paved the way for techniques of transferring knowledge obtained from virtual environments to the real world, often known as Virtual-to-Real (V2R) transfer [42]. V2R has been attempted in several ways, including Domain Augmentation, DA, Domain Randomisation (DR), etc. [42]. However, most of the V2R approaches are based on the assumption that the combination of the simulated and the real-world static data used for training the model will reflect the complete data distribution that an ED will observe post-deployment. This assumption does not always hold well, especially when the ED is vision-based, such as a mobile robot operating in a warehouse. The data observed by such an ED will be affected by several factors such as varying lighting conditions, scaling and warping due to motion, occlusion, etc. Thus, while DA compensates for the low data sampling power of an ED, the pre-trained model will still be susceptible to failure in the absence of onboard ML on ED. This calls for

the investigation of V2R ML techniques for resource-constrained devices like EDs. TinyML [140], an emerging sub-field of ML, focuses on using ML models on resource-constrained devices. The number of TinyML devices is expected to reach around 2.3 billion units by 2030 [141]. However, a majority of the TinyML techniques rely on offline training of Artificial Neural Networks (ANN) on *Cloud*/CPUs/GPUs which are then transferred to the target ED [142]. While online learning based TinyML methods for ANNs exist [143], the usage of gradient-based training exacerbates the issue due to its high susceptibility towards local optima, high computational overheads, high sensitivity to initial conditions and high memory requirements for saving intermediate tensors. GAs are known to perform better than gradient-based methods for optimising ANN parameters [43, 44]. However, GA-based optimisation of DNNs is a challenging task due to the large number of high-dimensional parameters involved, especially in vision-based applications. Accomplishing this as a TinyML method on an ED makes it all the more challenging.

This contribution proposes a novel TinyML method, nicknamed ***GADANN***, that leverages ***DA*** for V2R and ***GA*** for onboard training for a vision-based multi-class classification task on EDs using a *DNN*. The proposed method has been tested under a CPS setting comprising a mobile robot as an ED equipped with a camera, that performed the task of pallet recognition in a prototype warehouse environment.

5.1 Related Work

Since *GADANN* draws concepts from multiple areas including Virtual-to-Real transfer, TinyML and Genetic Algorithms, the subsequent sub-sections elaborate on each of these.

5.1.1 Virtual-to-Real Transfer

A major challenge in employing a DNN-based ML method in a real-world setting is the lack of availability of training data. The scarcity of real-world data arises from issues of data imbalance, labour-intensive labelling process and data privacy [42]. This has resulted in alternate approaches relying on the usage of virtual environments such as simulators for training the ML models. Such simulators are either used to generate synthetic training data (Syn-to-Real) [42] or are used to train an agent possessing the ML model that interacts within the virtual environment (Sim-to-Real) [42]. However, the difference in the features of the data observed in the virtual and the real domains causes a gap in the knowledge learnt by the model. This domain gap has been addressed using DR and DA techniques. DR involves randomisation of the simulated training parameters such as position, lighting, textures, etc. to minimise the bias of the virtual environment [144]. DR has been used in several V2R transfer-based applications, including object localisation [145], navigation [146], long-horizon multi-stage task [147], etc. DA methods generally utilise labelled data from a source domain to prepare a model for a target domain [148]. Most of the DA approaches learn a feature space shared across the virtual and the real data so that the knowledge learnt in the virtual setting can be directly mapped to the real world. Approaches involving DA differ both in terms of methodology and applications. DA has been realised by using Generative Adversarial Networks (GAN) [149], similarity learning [150], Siamese Neural Networks (SNN) [151], among others. In spite of such a vast acceptance, DR and DA-based methods pose their own challenges. The involvement of randomness in DR does not always align with reality and may harm the learning process [42]. Both in the case of DR and DA-based approaches, the evaluation of the effectiveness of the large virtual data in a real-world setting is difficult [42] and involves labour-intensive reiterations with ED in the loop. Moreover, hosting and operating a full-fledged ML-based DA model on an

ED is itself a challenging task. TinyML can cater to this challenge by smoothening the process of integrating ML with EDs.

5.1.2 TinyML: ML on the Edge

In a conventional Internet of Things (IoT) setup, the EDs share data with the *Cloud* which in turn, reverts with an appropriate decision for the ED to follow. With the rise in the amount of data available to the *Cloud*, the decision-making process is often governed by an ML algorithm running on the *Cloud*. However, such an exchange of data and decision has the following drawbacks [152, 142]:

- *Bandwidth constraint*: Sharing a continuous stream of heavy data such as images or videos with the *Cloud* is a costly exercise.
- *Energy constraint*: EDs are mostly battery-operated and communication across the IoT network drains a substantial amount of power.
- *Privacy*: In privacy-sensitive scenarios such as industries, scientific projects, healthcare, etc., sharing of data is not a feasible option.
- *Latency*: Due to the accumulation of large amounts of data on the *Cloud*, ML algorithms take their share of time to process the data and respond. Further, the sharing of data across the IoT network takes additional time. Thus, latency becomes a major issue when dealing with real-time applications.

These issues have given rise to the concept of TinyML that aims at minimising the need for an ED to consult the *Cloud*. It strives to make EDs *intelligent* by facilitating ML services onboard the ED [142]. TinyML has been used across a rich diversity of domains, including keyword spotting, anomaly detection, predictive maintenance, activity recognition, image classification, disease detection, farming, wildlife conservation and tracking, etc. [153, 152]. However, sparing a few[154, 143], most of the efforts in the TinyML domain are focused on the train-then-deploy approach. After training an ML model on *Cloud*, it is deployed as a static model on the ED without support for local adaptation. This introduces issues like concept drift, covariate shift and catastrophic forgetting [82], where the model trained on a different domain is no longer able to perform well on the data locally available to the ED [154]. Thus, extending the capacity of the TinyML method for online/incremental learning [155] is essential to keep the model's performance within permissible limits. However, issues like the low availability of local data and low computational and storage resources of EDs make onboard incremental learning for TinyML a challenging task. This exposes an opportunity of leveraging DA to compensate for the low data availability. It also calls for investigating alternative approaches to the compute-intensive gradient-based methods, such as the GA, for allowing onboard learning for TinyML models.

5.1.3 Genetic Algorithms

GAs are population-based metaheuristic algorithms that utilise multiple candidate solutions to search for an optimal solution in a search space [156]. These are inspired by the biological evolutionary process. A typical GA starts with a population P of C randomly initialised chromosomes/candidates. This is followed by a *selection* stage wherein top T candidates are selected based on their fitness values for further propagation. Candidate pairs from these T chromosomes are selected as parents and *crossed over* to yield a set of offsprings, O. These offsprings are merged

with a portion of the current parent population to yield a new population of size C. Thereafter, some of the candidates within this population undergo probabilistic *mutation*, yielding the next generation of candidates. The processes of *selection*, *crossover* and *mutation* are repeated until a candidate, having the desired fitness value, is obtained. GAs have been used across a wide variety of domains including information security, image processing, video processing, medical imaging, etc. [156]. GAs have also been used in conjunction with ANNs. While a majority of the usage of GAs for ANNs is focussed around Neural Architecture Search [157], GAs have also been employed for training ANN parameters [158]. GAs are able to reach better DNN parameter spaces, quicker than gradient-based methods [43] with lesser chances of getting stuck in a local optimum [44]. However, GA-based training of ANNs with high-dimensional data such as images is difficult, due to the proportionately increasing size of candidates and the resulting computations. This task gets more difficult when attempting it on EDs.

5.2 Methodology

The following insights were churned for devising $GADANN$, by surveying the state-of-art for ML on EDs:

- Since EDs have low sampling power, the method should leverage virtual data to support the ML training process.
- To dampen the effect of artefacts introduced by virtual data, our method should involve DA for transferring knowledge from the virtual domain to the real one.
- Since EDs are low on computational and storage resources, this method should use a TinyML approach for deploying the ML model on the ED.
- The statistical distribution of the real-world data observed by the model deployed on the ED may diverge from the data seen during training. Thus, this method should allow incremental onboard learning for the model. To conform to the computational resources of ED, the method should employ alternatives to gradient-based methods like GAs.

Keeping the aforementioned points in view, each of the steps of $GADANN$ are explained in th following subsections. An algorithmic schema of the whole approach is also shared in the Algorithm 6.

5.2.1 Data generation in the Real and the Virtual domain

Given an ED (ED_r), to be used in a real-world environment (Env_r), a virtual replica of the ED (ED_v), and the environment (Env_v) is created. ED_r interacts with the objects within Env_r to generate the dataset D_r, in the real domain while ED_v does similarly within Env_v to generate the dataset D_v. The modality of both the D_r and the D_v is the same. However, owing to the ease of conducting experiments in Env_v, the obtained size of the D_v is substantially greater than that of D_r.

5.2.2 Training a model for Domain Adaptation across Virtual and Real Domains

Training an ED operating in the real world on the small-sized D_r does not yield good performance. In addition, training a model from scratch on ED is a difficult exercise. Usage of the D_v, which is

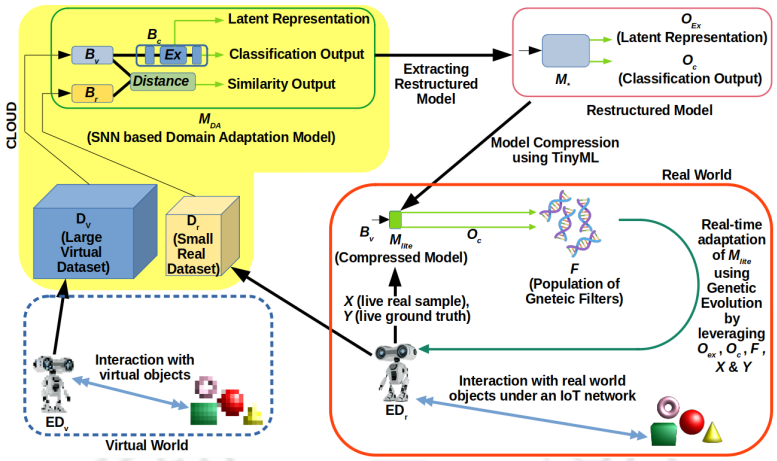


Figure 5.1: A schematic of the proposed *GADANN* method

available in abundance and with ease, can compensate for the sparse availability of data. However, due to the substantial difference in the features of D_v and D_r, a DA method similar to the one in [151] is used. As depicted in Figure 5.1, the DA model, M_{DA}, comprises two branches of ANNs, B_v and B_r, that bring the D_v and the D_r samples to lower representative embeddings, E_v and E_r, respectively. Both E_v and E_r are then merged into a *Distance* layer, as done in an SNN, to bring them to a common similarity space. B_v is further extended to form a classification block B_c. Additionally, we also add an *Extraction* layer Ex, before the last layer of B_c, so that it can be used later for onboard learning operations. During training, M_{DA} is fed with pairs of samples, one each from D_v and D_r along with their respective labels. An additional *similarity* label is associated with each pair, which is *1* when both inputs belong to the same class (positive pairs) and 0, otherwise (negative pairs). Unlike [151], we keep an equal number of positive and negative pairs to limit the training dataset size. This saves on the computations required while training the SNN, without substantially affecting the performance. Thus, M_{DA} takes two images (a virtual and a real one) as inputs and gives as output the measure of similarity between the inputs and the class label of the virtual input. The model is trained by using two loss functions: the contrastive loss [159] associated with the *similarity* label and the classification loss associated with the class labels. Training the model forces the network to bring the representative embeddings, E_v and E_r, closer to each other. At the same time, the presence of B_c in M_{DA} ensures that the network retains the class-specific information of D_v. Simultaneously chasing these two objectives results in DA, as the network learns to yield the class information of embeddings generated from any of the datasets D_v and D_r. Ex plays a crucial role in storing a concise latent representation of an embedding that can later be utilised for onboard learning under the resource-constrained environment of an ED. Owing to the complex and large architecture of the M_{DA}, this portion of *GADANN* is trained on the *Cloud*.

5.2.3 Extracting a simpler model for deployment on the Edge Device using TinyML

Since EDs are resource-constrained devices and already equipped with local sampling and control tasks, one needs to be judicious when enabling ML on EDs. TinyML enables the usage of ML on EDs by allowing the transfer of a learned model as a compressed file. We leverage this approach to bring the knowledge learnt by the M_{DA} in the *Cloud*, down to the ED. However, to remove unnecessary input channels and add the ability for onboard learning, we create a new model, M_*,

by picking specific components from the M_{DA}. We keep only one input channel for the M_* as it will only have an image from the real domain as input. On the output side, we remove the similarity output and retain the classification output. Lastly, we add the latent representation provided by the Ex layer as the second output of the M_* as shown in figure 5.1. Thus, for a given input, M_* gives its class information and a latent representation as the outputs. M_* is then compressed to a lighter representation, M_{lite} using TinyML, and deployed on the ED.

5.2.4 Onboard adaptation of TinyML model on the Edge Device using Genetic Algorithm

TinyML, in general, facilitates ML-based inference onboard an ED by bringing a pre-trained model down from the *Cloud* to the local storage of the ED. While this proves to be an advantage on the latency, the communication cost and the privacy fronts, it removes the capability of further adaptation by the model based on the data observed by the ED. Further, given the low computational and storage resource available with EDs; compute-intensive gradient-based incremental training, onboard the EDs, is not a good proposition. Thus, we leverage concepts from GA to utilise the latent representation output, O_{Ex}, and the classification output, O_c, of M_{lite}, to perform onboard incremental learning based on new data. O_{Ex} extracts a lower dimensional representation of the input, which minimises the computational overhead for online adaptation. However, directly tweaking O_{Ex} may disturb the underlying model, thus losing on the already learnt information. To avoid this, we rely on a population of genetic filters, F. These filters in conjunction with O_{Ex} and O_c, are used to infer about the input. The inference accuracy obtained over the input guides, in real-time, the changes to be made to the filters using GA. Thus, the usage of filters allows guided online adaptation without affecting the already learnt model. The adaptation process begins with randomly initialised genetic filters, F. For a given data sample X with ground truth vector Y obtained by the ED, each filter $f_i \in F$ yields a filtered representation F_{Ex} by using O_{Ex} corresponding to X as per the equation (5.1) below.

$$[F_{Ex}] = [O_{Ex}] * [f_i] \tag{5.1}$$

where '[ ]*[ ]' denotes matrix multiplication.

The dimension of f_i is chosen in such a way that the equation (5.1) transforms the high dimensional O_{Ex} down to the dimension of O_c. F_{Ex} is then used to weigh each of the elements of O_c as per the equation (5.2) below.

$$Pred = O_c * F_{Ex} \tag{5.2}$$

where $Pred$ corresponds to the prediction vector obtained by using f_i for the sample X. The performance of f_i over X is found to be good if $Pred$ is close to Y. The accuracy of f_i across all data samples is found and assigned as its fitness. Fitness values for all $f_i \in F$ are found, yielding a fitness distribution across the population of filters. This is followed by selecting top T f_is as parents, $f_{parents}$, for the next generation. To complete the population for the next generation, children, $f_{children}$, are generated using $f_{parents}$. Each child $f_{child} \in f_{children}$ is obtained by crossing over two parents equidistant from the middle of the $f_{parents}$ collection. The child inherits the former half of its contents from the first parent and the latter half from the second. $f_{parents}$ and $f_{children}$ are merged together. From the resulting population of filters, Mut number of filters are randomly selected for mutation. Each selected filter is mutated by multiplying its content with a uniformly distributed random noise. This ends the process of creating the next generation of filters, which proceeds to the next round of fitness evaluation. The process stops after a pre-decided number of

generations for onboard training is over. For inference over a newly observed sample, the fittest filter is selected and used as per the equations (5.1) and (5.2) for prediction. It may be noted that this onboard training process, including all the generations, can be continued as new data arrives. It may also be stopped once the desired performance is achieved.

5.3 Experiments and Results

Since the proposed method involves V2R transfer, the experiments were conducted both in virtual and real settings. The former part of this section provides a comparative analysis of the existing approaches of V2R with $GADANN$, for image classification. This is followed by an ablative study of the impact of various components of the GA mechanism used for onboard adaptation, on the performance of $GADANN$. We also conducted experiments to assess the impact of the ratio of the size of D_r to that of D_v on the performance of $GADANN$. The section ends with experimental analyses of the proposed method deployed on a virtual and a real robot for classifying pallets in a virtual environment and a real-world prototype warehouse, respectively.

The D_r and the D_v datasets were created in the lab., both in the real and the virtual worlds, respectively. The real-world experimental setup, involving the ED within the warehouse, and the equivalent virtual setup are shown in figure 5.2. A Firebird V [133] robot controlled by a Raspberry Pi (RPi) [160] with an onboard camera [134] was used as the ED_r. ED_r moved in the real-world setup, capturing images of the numbers (0-9) pasted on each pallet placed in the warehouse. The NodeMCU microcontroller [136] attached to each pallet communicated the pallet's number as the ground truth to the ED_r. This communication took place under an IoT network realised through the MQTT [135] protocol. Image samples and labels collected by ED_r formed the real dataset, D_r. Similarly, the virtual dataset, D_v, was created by using an e-puck robot situated within the Webots [161] simulation environment. Webots is an industry-standard 3D robot simulation software. Snapshots of the real pallet images were used as it is, to simulate the same within the Webots environment. As can be observed from figure 5.2, images of D_v are blurred due to pixelation of the virtual environment while those in D_r are sharp images. Due to their high dimensional nature (28x28 pixels per image), the D_v and the D_r datasets provided a comprehensive landscape for evaluating the proposed work. The ANN architectures were implemented using Keras [66] and Tensorflow [67]. The architecture of the M_{DA} used can be found in figure 5.3. The DA and the generative networks used in the experiments were trained using a *Cloud* GPU (NVIDIA T4(x2)). TinyML was realised by using the TensorFlow Lite library [162] for model compression and onboard inference.

5.3.1 Comparing Classification accuracies of existing V2R methods with $GADANN$

The classification performance of the following methods, trained using D_v and/or D_rr, were compared:

- CNN_V: A CNN similar in architecture to M_{DA}, with the lower non-classifying Siamese branch removed, was used. Only D_v was used as the training data.
- *CycleGAN*: A CycleGAN [163] generative model was trained using the D_v as the source and the D_r as the target dataset. The model was then used to generate a large dataset of synthetic images belonging to the domain of D_r. This synthetic dataset was then used to train a classifier similar to CNN_V.

Algorithm 6 GADANN for V2R based knowledge transfer & onboard adaptation for EDs

Input: D_v (Virtual Training Dataset), D_r (Real Training Dataset), D_{test} (Real test Dataset), P (Population size of Filters), Gen (Total Number of Generations), M (Number of Mutations)

Output: M_{lite} (TinyML model) & f_{best} (locally adapted filter) for usage on ED

Train $M_{DA}(D_v, D_r)$ on *Cloud*
M_* ←M_{DA} /* Create M_* from M_{DA} */
M_{lite} ←M_* /* Compressing M_* to M_{lite} */
O_{Ex}, O_c (outputs of M_{lite}) ←M_{lite}
F (Filters) ← $random.uniform(P, sizeOf(\mathrm{O}_{Ex}),$ sizeOf$(O_c))$ /* create a population of filters with randomly initialised values spread uniformly between 0 and 1. */
/* onboard incremental learning begins */
while $Gen > 0$ **do**
    $fitnesses_F \leftarrow []$
    for f_i *in* F **do**
        correct ← 0
        total ← 0
        for X, Y *in* D_{test} **do**
            f_{Ex} ←$[\mathrm{O}_{Ex}] * [f_i]$
            Pred ←O_c*f_{Ex}
            if $Pred = Y$ **then**
                correct ← $correct + 1$
            $total \leftarrow total + 1$
        accuracy ← $(correct/total) * 100$
        $fitnesses_F$.append(accuracy)
    F ← $sortByFitness$(F,fitnesses$_F$) /* rearrange filters in increasing order of accuracy over D_{test} */
    $f_{best} = F$[len(F-1)]
    $parents \leftarrow latterHalf(F)$
    $children \leftarrow []$
    for P_1 *in parents* **do**
        P_2 = pickParent($parents$) /* 2nd parent, equidistant from the centre in the other half of list of *parents* */
        $child$ = concat(formerHalf(P_1),latterHalf(P_2))
        $children$.append($child$)
    F ←parents+children
    $mutation\_indices \leftarrow random(len(\mathrm{F}))$
    for j *in* $mutation\_indices$ **do**
        $F[j]$ = uniformRandomNoise($F[j]$)
        $M \leftarrow M - 1$
    Gen ← $Gen - 1$
return M_{lite}, f_{best}

- SNN_{DA}: An SNN-based architecture for domain adpatation as suggested in [151] was used. An extra *Dense* layer was added in the classification block to make the architecture similar to

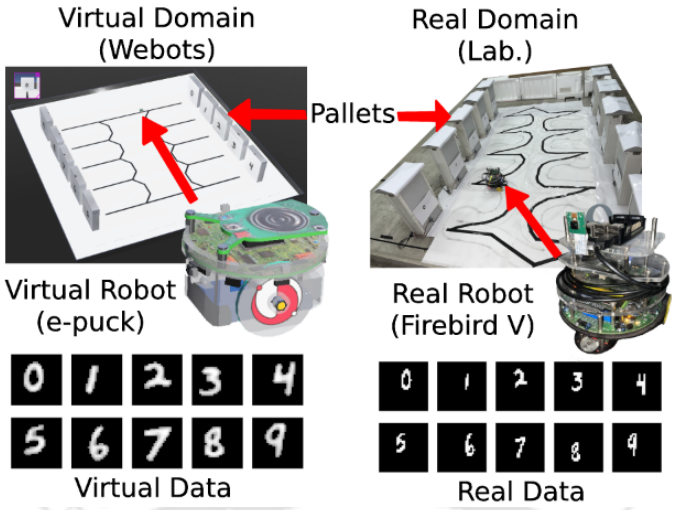


Figure 5.2: The Virtual and the Real experimental setup

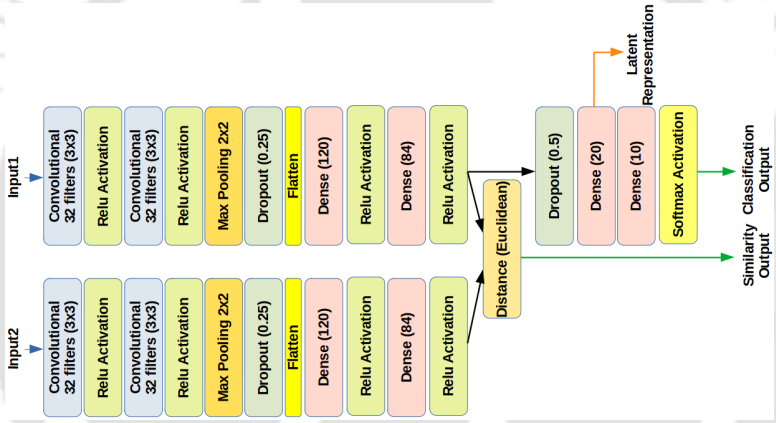


Figure 5.3: Architecture of M_{DA}

the ones used in other methods. D_v and D_r were used as the source and the target datasets, respectively.

- *GADANN*: This is the proposed method wherein instead of deploying the model on the robot, the same was tested on a real-world test dataset. The population size of the set of filters, F, was taken to be 100 while the number of generations and mutations per generation were fixed at 10 and 20, respectively.

- $GADANN_{Sh}$: This method is similar to *GADANN* except that, instead of feeding all the test data together, the data was shredded into 25 equal parts and fed sequentially as separate rounds. Each such round ran the GA portion of *GADANN* for all the generations. The filters, F trained in each round acted as the initial population for the next round. Each filter was represented by a matrix of size 20x10. Each value within the filter matrix can be considered as a weight used to hinder or enhance the corresponding features of the underlying input vector that undergoes filtration. The best filter at the end of all the rounds was then used to find the performance on the complete test data.

The test dataset for all the methods consisted of one thousand samples of images from the

Table 5.1: A comparison of the classification accuracies (%) of different V2R methods (Top 2 values of each column emboldened)

#Samples/Class (D_r)	**5**	**5**	**5**	**10**	**10**	**10**	**20**	**20**	**20**
CNN$_V$ (D_r not used)	9.89			10.10			13.09		
CycleGAN	8.60	14.00	6.70	10.70	9.49	12.09	11.99	9.30	12.89
SNN$_{DA}$	**63.8**	**65.0**	**69.5**	**84.3**	**84.1**	**83.5**	**95.4**	**94.7**	**92.7**
GADANN	56.4	54.6	64.1	77.4	80.5	77.8	94.1	90.7	84.8
GADANN$_{Sh}$	**65.3**	**66.0**	**70.8**	**81.8**	**86.9**	**81.7**	**94.1**	**94.5**	**91.9**

real domain.

The size of D_v was fixed at one thousand samples for all methods, while the size of D_r, wherever used, was varied as 5, 10 and 20, samples per class. This yielded correspondingly different models, each of which was trained from scratch. Further, three experiments were conducted using each of these models to remove any possible stochastic variations thus, totalling 39 experiments. The classification accuracies obtained on the test dataset are shown in table 5.1, with the top two accuracy values for each column, emboldened.

Following insights may be drawn from the table 5.1:

- Training a CNN by using only D_v and testing it on the real-world data, without DA, yields poor accuracies, as observed in the case of CNN_V. This shows the adverse impact of the virtual artefacts introduced by simulators when creating a dataset for an almost identical real-world setting. This not only necessitates the need for domain adaptation but also exposes the limitation of normal CNN-based methods in V2R scenarios.

- *CycleGAN* is known to perform well in DA tasks by translating an image to another domain. However, it is extremely computation and data-hungry. Thus, it performs poorly on the given V2R task due to the low availability of images from the target (real) domain. Figure 5.4 shows some of the images generated by *CycleGAN* (size of D_v & D_r as 1000 & 200, respectively), whose poor quality uncovers the reason for the low performance.

- SNN_{DA} maintains a lead over the proposed *GADANN* method. However, it must be noted that the population size and the number of generations in *GADANN* were kept fixed for the experiments which can be further tweaked. The crossover function used in the case of *GADANN* was very simple and could be changed to a more 'feature-preserving' method. Most importantly, unlike SNN_{DA}, *GADANN* is not a static method and allows incremental onboard learning which thus, gives the latter an edge over the former.

- $GADANN_{Sh}$, in general, is able to maintain a lead over SNN_{DA}. This highlights the utility of *GADANN* for real-world scenarios, where all of the data is not available at once. $GADANN_{Sh}$ continues to learn on the new data and performs better across rounds by adapting to the data observed at each round.

To further distinguish between the *GADANN* and the $GADANN_{Sh}$ methods, we conducted a set of five experiments to analyse the inference and the training times in each of them. Experiments using *GADANN* used the complete data at one shot while those for $GADANN_{Sh}$ comprised 25 rounds where the test data was split into 25 subsets and fed incrementally across each round. The *GADANN* and the $GADANN_{Sh}$ methods leveraged 50 candidates that evolved across 10 generations. These experiments were run on a *Kaggle Cloud Kernel* (operating on an Intel(R)

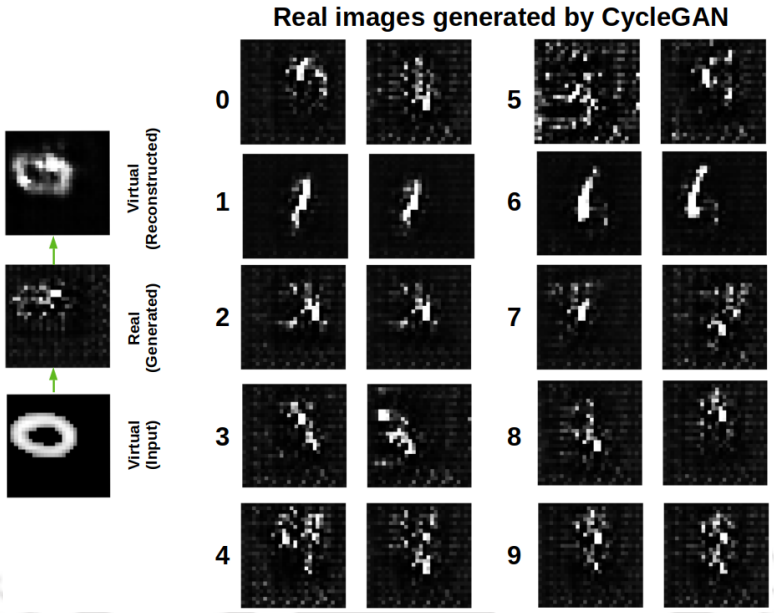


Figure 5.4: Low-quality Real images generated by CycleGAN due to low availability of data from the Real domain

Xeon(R) CPU @ 2.20GHz) [164]. Figure 5.5 shows the inference and training times observed for the $GADANN$ and the $GADANN_{Sh}$ methods. As observed, $GADANN_{Sh}$ takes lesser time than the $GADANN$ method for each round of the former. This is so because each round of $GADANN_{Sh}$ uses a small subset of the total data, leading to lower computational iterations than in the case of $GADANN$.

5.3.2 Ablation study of the impact of various components of GA on $GADANN$

The presence of GA in $GADANN$, not only facilitates onboard adaptation but also substantially enhances the algorithm's performance. To further analyse the impact of its major components, namely crossover and mutation, we conducted an ablation study of the GA portion of the $GADANN$ method by using the following strategies:

- Mut: This $GADANN$ based strategy utilised GA with only mutation. Crossover was suppressed.
- Mut_{str}: Here $GADANN$ was used by following a stratified crossover approach. Given two parent filters, the child's content was partitioned into multiple strata. Each stratum's content was then filled by copying the corresponding stratum's content chosen alternatively from two parents, followed by mutation.
- Mut_{cont}: This strategy resembled Mut_{str} but for the fact that instead of partitioning the child's content into multiple strata, the content was split into two continuous halves. The first half borrowed content from the first half of the first parent and vice versa for the other half.
- Cr_{str}: The strategy used herein was similar to Mut_{str} but for the absence of the mutation operator.

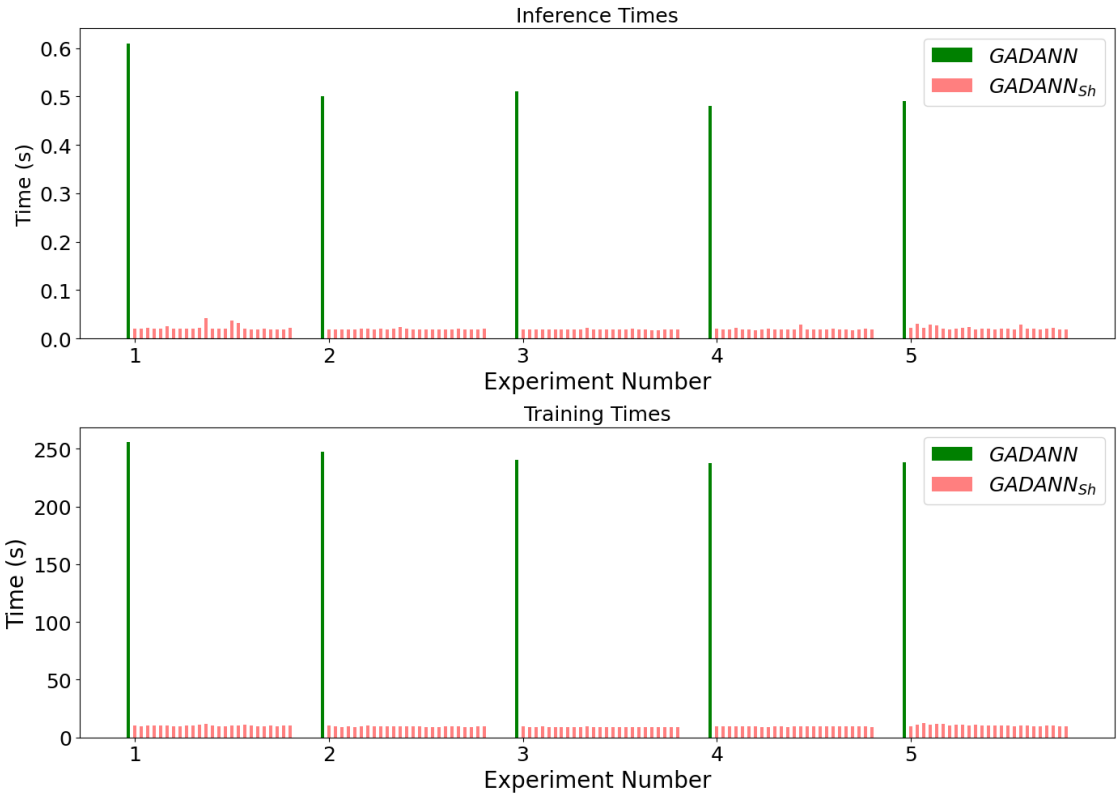


Figure 5.5: Comparison of the inference and training times of $GADANN$ and $GADANN_{Sh}$

- Cr_{cont}: This strategy was similar to Mut_{cont} but did not use mutation.

Each of the above strategies was tested on a given test dataset from the real domain having 1000 samples. The size of D_v was kept fixed (1000 samples) while the size of D_r was kept as 5, 10 and 20 samples per class, yielding correspondingly 15 different models. Three experiments were conducted for each of the resulting models under each strategy totalling 45 experiments. Table 5.2 shows the classification accuracies achieved for the aforementioned experiments. The GA parameters comprised a mutation rate of twenty per cent and a population size of 100 maintained across a total of ten generations. The crossover scheme included creating half of the population as half of the best-fit parents and the other half of the population as the offspring created by crossover.

It can be observed from the table that both the Cr_{str} and the Cr_{cont} strategies yield similar classification performances. This trend is also followed by the Mut_{str} and the Mut_{cont} strategies. Thus, choosing amongst the given crossover methods does not seem to make much difference. However, the inclusion of the crossover step in the GA portion boosts the performance of $GADANN$. This can be observed by the performance increments observed in general, for Mut_{str} and Mut_{cont}, when compared with Mut. Further, the mutation step also plays a substantial role in enhancing the performance of $GADANN$ which can be observed from the boost in the accuracies shown by the Mut, Mut_{str} and Mut_{cont} strategies when compared to the Cr_{str} and Cr_{cont} strategies.

5.3.3 Impact of the ratio of size of D_r & D_v on $GADANN$

Too much use of virtual data may bias M_{DA} while using too less of it may prove to be futile. Thus, a congenial ratio was found between the sizes of D_r and D_v so as to provide optimal performance.

Table 5.2: An Ablative analysis of the impact of the components of Genetic Algorithm on the classification accuracy (%) of *GADANN*

#Samples/ class (D_r)	*Mut*	*Mut_str*	*Mut_cont*	*Cr_str*	*Cr_cont*
5	41.1	57.1	54.4	46.0	43.8
5	37.0	57.6	56.5	41.8	35.0
5	55.3	65.4	62.7	53.1	53.1
10	58.9	72.8	77.2	54.1	51.5
10	66.0	79.8	77.4	66.0	59.4
10	57.9	79.0	74.4	54.0	57.9
20	83.7	94.4	94.7	77.3	83.2
20	85.8	78.4	81.1	74.0	72.4
20	75.0	84.0	91.5	63.7	67.7

With the size of D_r fixed at 20 samples per class, the size of D_v was varied as 20, 50 and 100 samples per class. For each D_v size, three different M_{DA} models were created from scratch.

For each such model, *GADANN* was run three times (totalling 27 experiments) using a real-world test dataset comprising 1000 samples. Figure 5.6 shows the accuracies obtained across all the experiments. It can be observed that increasing the size of D_v increases the accuracy. The accuracies saturate around a D_r-D_v proportion of 20:100 samples per class.

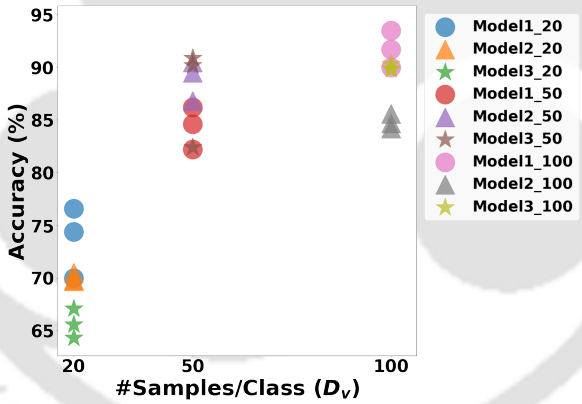


Figure 5.6: Impact of varying D_v size with D_r size fixed at 20 samples/class

5.3.4 Deploying *GADANN* on a Virtual and a Real Robot

The efficacy of *GADANN* was tested by deploying it both on a virtual and a real robot in a simulator and a real-world CPS setting, respectively, as shown in figure 5.2. The real-world CPS involved a prototype warehouse environment having pallets with numbers (0-9) pasted on them. A Firebird V robot, controlled by an RPi with an onboard camera, was used in the experiments. The robot's task was to start from the first pallet and autonomously move around the warehouse, following the line and recognising the number on each pallet. Upon encountering a pallet, the robot took four shots of the number pasted on the pallet, from different orientations. Thereafter, the robot used the onboard portion of *GADANN* (M_{lite} + GA filter) to arrive at an inference. A microcontroller attached to the corresponding pallet provided the ground truth label to the

robot using the IoT network within the warehouse. The robot made a round of the warehouse by covering each pallet (totalling 40 shots). At the end of the round, the robot calculated its inferencing accuracy. The 40 samples collected by the robot were then used for onboard training of the M_{lite} using the $GADANN$ method by evolving F using GA. This process was continued for four rounds under one experimental run. The filters, F, trained in one round were used as the initial population for the next round.

Table 5.3 provides the times taken by the mobile robot to cover each round together with the combined times taken to make inferences about the current round and use the collected samples to train for the next round using the $GADANN$ method, for two different real-world experiments. The data for the fourth round has not been included in the table as it did not have a corresponding training time at the end due to the end of the experiment. It may be noted that unlike the inference and training times shown in Figure 5.5 where experiments were run on a PC, the ones shown in Table 5.3, involved $GADANN$ running onboard a mobile robot, controlled using an RPi. This is the reason why the inference and training times are different from those shared in Figure 5.5.

Experiment No.	**Round Time**	**Inference & Training Time (combined)**
1	7m 1s	3m 41s
	7m 8s	4m 31s
	7m 39s	3m 31s
2	7m 8s	3m 44s
	6m 47s	3m 40s
	7m 15s	3m 40s

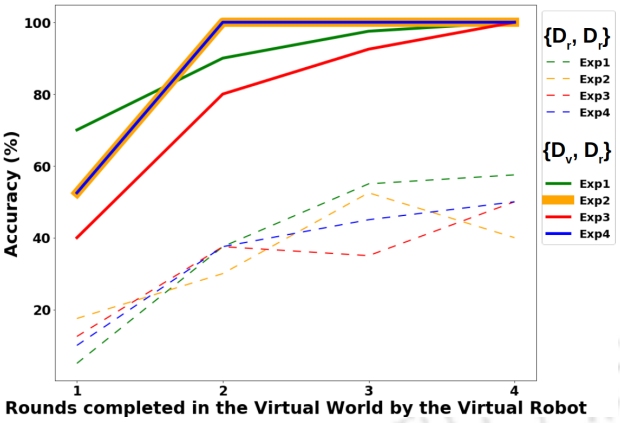
Table 5.3: Time analyses of real-world execution of $GADANN$ on a mobile robot

The same set of experiments was also performed in a virtual environment created in the Webots simulator using an e-puck robot. Furthermore, each of these virtual and real experiments was conducted in the following ways:

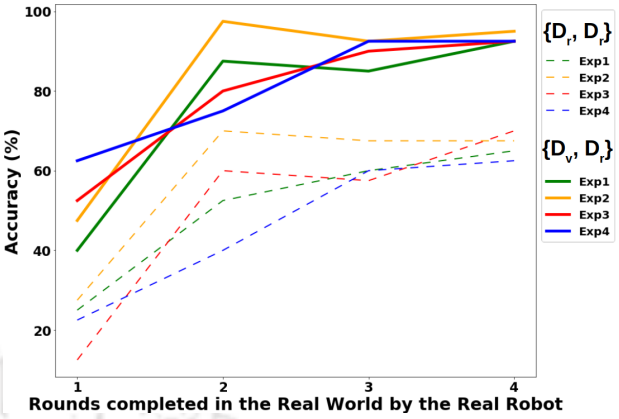
- $\{D_r, D_r\}$: Using only a small D_r (200 samples) for training M_{DA} as both the source and the target
- $\{D_v, D_r\}$: Using a large D_v (1000 samples) as the source and a small D_r (200 samples) as the target for training M_{DA}

This was done to analyse the impact of DA and the virtual data on the real-time performance of $GADANN$, deployed on an ED interacting with the environment. Four sets of experiments were conducted for each of the two ways in both the real and the virtual worlds (totalling sixteen experiments). Figure 5.7 details the inference accuracies obtained for all the experiments in both the virtual and the real domains.

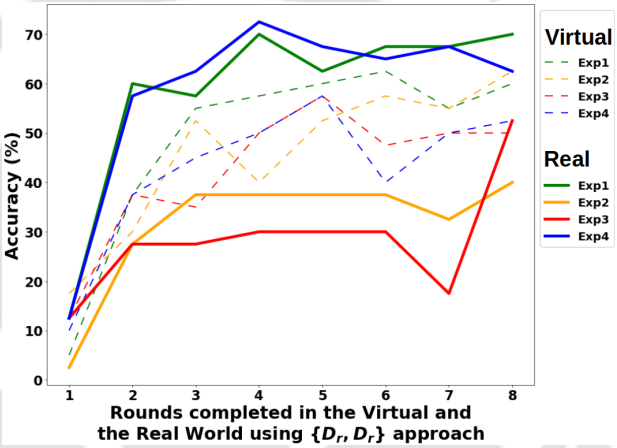
It can be observed from figure 5.7(a) that the absence of usage of DA using the D_v ($\{D_r, D_r\}$), forces the accuracy values to converge to a low mark of around 60 per cent. This convergence to a lower accuracy has been tested by increasing the number of rounds to eight in case of $\{D_r, D_r\}$ (figure 5.7(c)). It can be observed that, in general, the accuracy values beyond the fourth round do not increase substantially. The same experiment when conducted by involving the D_v in the DA process ($\{D_v, D_r\}$) pushes the accuracy beyond ninety per cent (figure 5.7(a)) within four rounds. This highlights the importance of the usage of DA and the virtual data for training, in $GADANN$. However, even in the absence of the D_v (figure 5.7(c)), the local adaptation capability of $GADANN$, imparted by GA, can still be appreciated by observing the general trend of increasing accuracies



(a) Virtual-World Performance



(b) Real-World Performance



(c) Converging performance of $\{D_r, D_r\}$ approach in both virtual and real worlds

Figure 5.7: Pallet recognition accuracy of GADANN deployed on the Virtual & the Real Environment

across the eight rounds. Figure 5.7(b) shows an overall decrement in the accuracies achieved across each corresponding round in the real world. This highlights the effect of *domain shift* which makes the well-trained model perform poorly in the noisy real-world environment. It may also be observed that the accuracy in the first rounds of all the experiments is very low. Since the first round does not involve any GA-based adaptation, it is equivalent to running the SNN_{DA} approach onboard. This is a crucial observation as it highlights the limitation of static methods such as SNN_{DA} when brought to the real world. Such static models fail miserably even on semantically close data, rife with real-world aberrations. However, as the GA portion of $GADANN$ takes over, a substantial jump in the accuracy can be observed in the second round of all the experiments, followed by a general upward trend. This warrants the need for incremental methods like $GADANN$ for real-world deployments, that can adapt to the environment, thereby increasing the performance. The cyber and the physical interplay of the computational model and the mobile robot along with the pallets, under this CPS, facilitate onboard adaptation with respect to the real-world data. The appendix of this thesis (and the footnote below[1]) shares links to videos of the recording of the

[1] Video 1 Link, Video 2 Link, Video 3 Link

experiments conducted using the *GADANN* method on a real and a virtual mobile robot in a prototype warehouse setting.

5.4 Chapter Summary

Enabling Machine Learning capabilities on EDs is crucial for supporting several automation scenarios. Given the low availability of real-world data for training, Virtual-to-Real knowledge transfer methods are often utilised for training Machine Learning models for deployment on EDs located in the real world. However, the difference between the virtual and the real-world data hampers the post-deployment performance of the model. While Domain Adaptation based methods allow a model to learn features shared across the virtual and the real world, the resulting model is static and too big to be used in conjunction with an ED. TinyML allows the usage of Machine Learning models on resource-constrained devices by compressing the models into small transferable files. However, most of the existing TinyML onboard operations are restricted to drawing inferences and do not facilitate onboard training. To tackle these challenges, this contribution proposes *GADANN*, a TinyML-based Virtual-to-Real knowledge transfer method that facilitates onboard adaptation. *GADANN* extends Machine Learning onboard the EDs by leveraging Deep Neural Networks, Domain Adaptation, TinyML and Genetic Algorithms. Unlike the resource-hungry generative methods of domain adaptation, *GADANN* is able to consistently transform knowledge from the virtual to the real domain in scenarios involving sparse availability of data and computational resources. The method has been tested successfully in a real-world setting by deploying it on a real robot in a warehouse prototype to identify pallets using computer vision. Being adaptive in nature, *GADANN* delivers a substantial performance boost in real environments and stands out from the static models that fail to accommodate the uncertainty and inconsistency of real-world data.

The contribution highlighted in this chapter introduces a completely gradient-free resource-efficient approach for onboard ML on resource-constrained devices. It also leverages a cost-effective way to deal with scarcity of data and computational resources by relying on simulated data. The simulated data allows pre-training on a larger pool of data, giving a head start to the onboard ML model before being deployed on a real world CPS. This chapter also highlights the role and the requirement of datasets that can be used to evaluate DA for robot vision based tasks, which is explored in the next chapter.

6

RoboDA: A Dataset for Domain Adaptation in Robot Vision

The contributions cited in the preceding chapter identified a void in the availability of a proper dataset for evaluating a robot vision task. In the process, with both the real-world and the simulated experimental setup within reach, a good-quality dataset had to be collected and used to validate the experiments. In practice, while experimentation, the availability of both these real and simulated setups cannot always be ensured. There is thus, a strong requirement for a dataset that covers both the domains - real and simulated. This chapter describes the dataset that was created and subsequently used in this research. This chapter has been included to describe this dataset pertaining to robot vision, nicknamed, RoboDA. The dataset involves standard open-source components that aid in the easy realisation and reproducibility of the associated experiments. The dataset is particularly useful for evaluating CPS-based settings that involve physical components in addition to software, requiring data samples to reflect interactions with the real world.

Modern AI techniques such as DNNs struggle to deliver the desired performance in scenarios where data is sparse or expensive to capture [165]. This issue is predominant in tasks involving interactions within the real world, such as in the field of robotics. While the field of computer vision has benefited immensely from DNNs, robot vision still holds numerous challenges. Robot vision involves the use of a vision model as an active entity in an embodied and situated robot. The performance of the model is thus, susceptible to the actions of the robot as well as the changes in its environment. These changes are due to issues such as occlusion, differences in illumination, scaling, rotation, noisy sensor values, etc. [26]. While DNN-based methods can tackle these issues, collecting a large amount of vision data using a robot is a costly and slow process. Synthetic data generated using simulators, are easy to create and inexpensive to collect and often compensate for this data crunch [166]. However, DNNs trained purely on synthetic data fail to perform well on real-world data perceived by the robot due to the domain transfer issue [165]. TL, data augmentation, DA, domain randomisation, etc. have been used to overcome this issue [167]. Each of these solutions leverages the knowledge learnt on the simulated data to perform better on the real-world data, often called as the SimToReal or SyntheticToReal approach (StoR). A variety of datasets for evaluating the StoR methods are now available[168, 169, 170]. However, most of these existing datasets used for StoR evaluation either consist of images captured from static cameras or those created using specialised or expensive apparatuses such as industrial robot grippers, cars, etc. Such constraints on the experimental apparatus limit the reproducibility of the experiments for robot vision tasks using such datasets. In most of the works dealing with DA, separate datasets are merged to represent different domains, without involving any robotic component in the acquisition

process. Those that involve a robot, often lack the usage of an onboard egocentric camera that could correctly represent the challenges faced in robot vision. This contribution attempts to cater to these issues by introducing a novel dataset, nicknamed **RoboDA**, suited for testing StoR based DA methods for robot vision. **RoboDA** comprises images from both the real-world and the simulated domain, captured using a camera-mounted *mobile* robot. The dataset utilised resource-efficient and standard apparatus, allowing easy extension of the experiments. The contribution covers a comprehensive survey of the existing datasets for DA, along with their limitations, followed by a detailed explanation of the acquisition process and the description of the **RoboDA** dataset. The latter part of the contribution comprises the experimental analyses of various state-of-the-art DA methods performed using the **RoboDA** dataset, justifying the challenging feature landscape that the dataset provides for vetting DA-based methods. A comparison of the **RoboDA** and MNIST [37] datasets has also been provided, highlighting the relevance of the former to the domain of robot vision. The contribution also provides a qualitative analysis of the **RoboDA** dataset by visualising its sample space and features landscape along with the impact of DA on it.

6.1 Related Work

While DNNs deliver impressive performance by using large datasets, creating and annotating such massive datasets is not always feasible in certain domains. DA is a technique used to adapt to a target domain, having limited data, by leveraging a source domain having a large amount of data. This is generally achieved either by weighing the source samples during training or by bringing the source and the target data distributions into a common space [171]. DA has gained a lot of interest in the research community for various applications, including robot vision[26], pedestrian detection [166], image classification [150], face recognition, object detection, style translation [171], etc. This has resulted in the generation of a vast variety of datasets that are used for benchmarking DA applications. Table 6.1 enumerates some of the most widely used datasets for evaluating DA-based methods for computer vision applications.

The **Industrial Metal Objects** dataset [169] comprises images of industrial metal objects with 6D object pose labels. It has 31,200 real-world images and 5,53,800 synthetic images. The real-world images were captured by an industrial robot using multiple cameras. While this dataset has high-quality and well-annotated images, the reproducibility of the apparatus is difficult due to the costly and specialised apparatus. The **Virtual Pedestrian Dataset (ViPeD)** [166] contains synthetic images collected using videos of different urban scenes from the GTA V video game organised into a training and a test set, having 256 videos each. The dataset lacks a real-world counterpart. The absence of the usage of a robot makes this dataset less usable for evaluating DA methods for robot vision. The **ParalleEye** dataset [172], often used for segmentation-based tasks, has seven subsets dealing with roads and traffic. Each subset has a variation in the number and types of objects, occlusion, etc. The dataset was generated using the Unity3D platform with geographical data imported from OpenStreetMap [173]. It includes six classes of targets for object detection and twelve labelled categories for semantic segmentation. However, the dataset lacks a real counterpart and involves an intricate annotation process. The **LLR106 and LLS500** datasets [174] consist of 3D point clouds of three classes of objects namely *car*, *tree* and *people*, sampled using a LIDAR sensor. The datasets have 500 point clouds for each class in the synthetic domain and a total of 484 point clouds in the real domain. The *Webots* [161] simulator was used to create the synthetic data. While the presence of the real-world counterpart helps in easily evaluating DA methods, any further extension of the related work is hindered by the requirement of specialised/expensive equipment such as the LIDAR and the IMU sensors. The **Visual Domain**

Dataset	Description	Size	Domain
Industrial Metal Objects	RGB images with 6D object pose labels of industrial metal objects	5,85,000 images	Simulated and real
ViPeD	Synthetic images from the graphical engine of the GTA V video game	5,00,000 images	Simulated
ParallelEye	Synthetic images of visuals from roads	40,251 images	Simulated
LLR106, LLS500	3D point clouds of objects	1,984 samples	Simulated and real
VisDA	Images of objects for classification and segmentation	3,10,000 images	Simulated and real
Scene Classification	Images of scenes from different categories	9,600 images	Simulated and real
SIP-17	Images of industrial parts	33,566 images	Simulated and real
TuSimple	Images of roads for lane detection	6,408 images	Real
CULane	Images from scenes of urban, rural and highway settings	1,33,235 images	Real
Office-31	Images of common objects encountered in office	4,652 images	Real
Office-Home	Images of common objects encountered in office and home	15,500 images	Artistic and real
ImageCLEF-DA	Images of object and living things with varying picture quality	1,800 images	Real
MNIST, MNIST-M, USPS, SVHN, Synthetic digits	Images of digits in various domains	70,000; 1,49,002; 9,298; 99,289; 5,00,000 images	Simulated and real
DomainNet	Images of common objects from various domains	600,000 images	Artistic and real
Oxford RobotCar	Images, LIDAR and GPS data from a vehicle	23.15 TB	Real
Cityscapes	Images captured from moving vehicle	25,000 images	Real
GTA V	Images taken from the GTA video game	24,966 images	Simulated
SYNTHIA	Synthetic images of urban scenes	9400 images	Simulated
KITTI	Vehicle data from multiple modalities	44,997 samples	Real
Sim10k	Images of objects from GTA V	10,000 images	Simulated

Table 6.1: Existing datasets used for Domain Adaptation tasks across simulated, real and artistic domains

Adaptation (VisDA) dataset [168] deals with cross-domain object classification (VisDa-C) and image segmentation (VisDa-S). VisDa-C has objects from 12 categories, split into three domains. Images for each of the domains are generated from 3D CAD models, from the COCO dataset [175] and the Youtube Bounding Box dataset [176]. VisDa-S has segmentation data extracted from the GTA V video game, the **Cityscapes** dataset [177] and the Berkley-Nexar dataset [178]. **VisDA** has a rich variety of domains that allows extensive evaluation of DA methods. The **Scene Classification** dataset [179] focuses on the classification task, involving six types of environments, namely, *forest*, *field*, *bathroom*, *living room*, *staircase* and *computer lab*. Each environment has synthetic images (1000 per environment) generated from the Unity video game engine and real-world images (600 per environment) collected using a web crawler. The **SIP-17** dataset [170] deals with the classification of seventeen types of industrial objects. It has both synthetic and real-world images, generated from 3D CAD models and captured from various industrial scenes, respectively. The **TuSimple** dataset [180] consists of images of lanes captured from the real world. The **CuLane** dataset [181] comprises real-world scenes from urban, rural and highway settings. The test data contains nine different driving scenarios. It also includes occluded images, thereby making the data more challenging. The **Office-31** dataset [182] contains real-world images of daily-use office objects spread across 31 categories. The dataset draws images from three different sources, namely, the Amazon website, the webcam and the DSLR. The **Office-Home** dataset [183] comprises images of objects available at offices and homes, spread across 65 classes. It has artistic images such as sketches, paintings, clipart and real-world images captured from a regular camera. The **ImageClef-DA** dataset [184] combines images from three sources, CalTech-256 [185], ImageNet [186] and Pascal VOC [187]. Each source has twelve categories having fifty images in each. However, the image sources vary in their quality. A substantial amount of DA methods use digit classification as one of the applications. Datasets like **MNIST** [37], **MNIST-M** [188], **USPS** [189], **SVHN** [190] and **Synthetic digits** [191] are commonly used to represent separate domains under such applications. Most of these datasets contain images of numerals with variations in background, colour channels, image dimensions, etc. While their ease of use makes them an attractive choice for DA methods, the lack of real-world noise such as occlusions, illumination differences, scaling and rotation, makes them a less challenging choice for evaluating robot vision DA methods. The **DomainNet** dataset [192] consists of images of objects from 345 different categories. The dataset is spread across six different domains, namely, *clipart*, *infographics*, *painting*, *quickdraw*, *real* and *sketch*. The **Oxford RobotCar** dataset [193] is a large dataset comprising samples with different modalities including stereo images, 2D and 3D LIDAR points, inertial data and GPS data. All these samples were recorded across 100 iterations of a route covered through a vehicle in Oxford, United Kingdom, with sensors onboard. The dataset captures various combinations of weather, traffic, pedestrians, roadworks and constructions. It is also used in autonomous driving scenarios. The **Cityscapes** dataset [177] comprises frames acquired from a moving vehicle covering spring, summer and fall across fifty cities. It has 5,000 high-quality pixel-level annotations and 20,000 coarse annotations. The **GTA V** [194] dataset contains synthetic images collected from the GTA V video game. The **SYNTHIA** dataset [195] contains synthetic images of urban scenes with pixel-level annotations for thirteen classes of components visible in the scene. The synthetic data is generated using the Unity platform. The dataset contains a set of images captured from randomly moving cameras in a virtual city and another set of images extracted from videos captured through a virtual car moving across different seasons. It is often utilised for autonomous driving scenarios. The **KITTI** dataset [196] was captured by driving around Karlsruhe, Germany. It includes samples with different modalities such as images, laser scans, GPS, IMU data, etc. The dataset is primarily targeted for autonomous driving and robotics. The dataset has three parts, including a 2D object detection dataset, a 3D object detection dataset and a Bird's Eye view dataset. Lastly, the **Sim10k**

dataset [197] contains synthetic images rendered using the GTA V video game. It is primarily utilised for object detection applications. The datasets discussed so far are well-suited to evaluate generic vision-based DA methods. However, they do not suffice well for evaluating robot vision DA methods. For instance, a CNN can perform well for generic computer vision tasks but can fail to achieve StoR DA for a dataset involving closely related domains where the target domain has limited data and is rife with real-world noise. Thus, a dataset for evaluating StoR DA for robot vision should be focused more on embodying such real-world challenges than on having multiple modalities or large size. Some more similar issues regarding the mentioned datasets are as follows:

- Many of the discussed datasets deal either with the real or the simulated domain, which requires the user to look for a counterpart to realise DA. Searching and processing such counterparts adds to the implementational overhead.
- Experimental apparatus involving real vehicles, specialised robots with multiple cameras, expensive sensors, etc. used in conjunction with some of the discussed datasets is expensive and difficult to set up. This hinders the reproducibility of the apparatus for extending the experimental analysis further.
- Very few datasets capture samples using a camera onboard a mobile robot. Such a configuration allows more mobility to the robot and introduces noise to the captured samples due to the motion of the robot, jerks, changes in illumination, scaling and rotation, etc.

The **RoboDA** dataset is introduced to meet the aforementioned challenges.

6.2 The RoboDA Dataset

This section shares the acquisition process and the description of the **RoboDA** dataset.

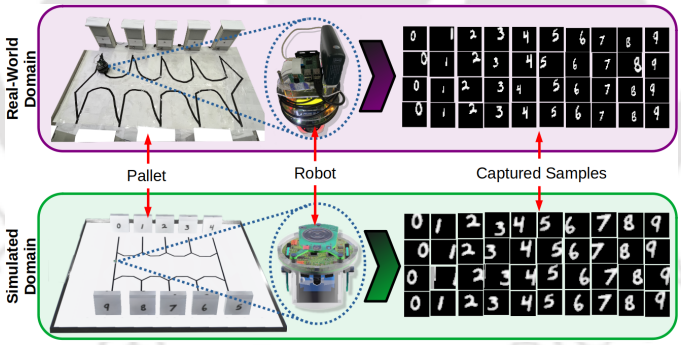


Figure 6.1: A schematic of the RoboDA dataset acquisition setup

6.2.1 Acquisition Process

The **RoboDA** dataset contains images from two domains (*Real-world* and *Simulated*). For each domain, the data acquisition process was carefully crafted to have standard, open-source and affordable components, that ensure easy reproducibility of the related experiments. The data acquisition process is completely automated, thereby allowing the dataset to embody all the imperfections and noises encountered in robot vision tasks. Both the real and simulated domains used an autonomous mobile robot, equipped with an onboard camera, to collect the images. A description of the manner in which this was performed for the two domains follows.

The Real-World Setup

The real-world domain required a setting that exposes the dataset to maximal possible noise such as changes in illumination, scaling, rotation, occlusion, etc. At the same time, the setup needed to be easily deployable for reproducibility. Thus, a prototype warehouse as shown in Figure 6.1 was created in the lab. so as to emulate the real-world domain. The warehouse comprised ten pallets with a unique number (0-9) pasted on each of them. Each of the pallets, P_i, was also deployed with a NodeMCU [136] microcontroller, M_i, that was interfaced with an ultrasonic sensor, U_i. The images of the dataset were autonomously captured by the camera [134] onboard a Firebird V [133] robot, controlled using a Raspberry Pi [160] board. The robot also used infrared sensors to detect obstacles. The floor of the warehouse had a path that the robot could follow, using a line sensor, so as to reach each pallet and capture the corresponding images. The robot and the pallets were connected together as an IoT setup, using the MQTT [135] protocol. The acquisition process starts by placing the robot on the path before the first pallet. The robot then starts to follow the path using its line sensors, till it encounters the initial pallet, P_0. The robot halts on sensing this pallet using its infrared sensors, and in the process, its onboard camera comes face to face with the number pasted on the pallet. The robot then captures four images of this number on P_0 from four different orientations by changing its directions. Capturing images from different orientations ensured that the dataset had a good variety of samples and included additional possible noises due to rotation, scaling and occlusion. P_0, whose sensor U_0 is enabled initially, senses the presence of the robot through M_0. P_0 provides its corresponding number, which is zero, as the label to the robot via the IoTized network. This done, P_0 disables U_0 and broadcasts the label of the next pallet to all other pallets using the IoT network. On receiving this broadcast, the next pallet, P_1 enables its sensor, U_1, to begin sensing the presence of the robot in front. Broadcasting is done to avoid crosstalk between the pallet sensors by allowing only one of the sensors to be active at a time. Meanwhile, the robot saves the sampled images in its onboard memory and once again commences following the path leading to the next pallet, P_1. This process is repeated for all ten pallets, till P_9, to complete one round of the warehouse. Thus, each round yields forty (4 orientations x 10 pallets) images sampled along with the corresponding labels. Such rounds are repeated until the desired amount of samples are collected. Real-world data is tough to capture and thus, always available in small amounts. This challenge is also needfully reflected in the **RoboDA** dataset, containing a limited amount of real-world data. Owing to the generic type of the sensors and the robot used, the real-world data can easily be replicated if required to increase the corresponding dataset size.

The Simulated Setup

To replicate the warehouse prototype in simulation, *Webots* [161], an open-source and industry-standard 3D robotics simulator was used. The warehouse setup was recreated in *Webots* by creating the corresponding number of pallets and the floor. A virtual e-puck robot (Figure 6.1) armed with a distance sensor, a light intensity sensor and an onboard camera, was used to perform the manoeuvring and sampling tasks. The virtual robot collected the simulated samples by following the same procedure as explained in the previous subsection, including preprocessing of the images. An advantage of using the *Webots* simulator was the ability to speed up the experimental rounds. This allowed us to capture a large amount of simulated data within a relatively shorter duration than the real-world scenario. To maintain consistency, high-quality images of the real-world pallet numbers were captured and overlaid on the virtual pallets. The following subsection describes the details of the images within the collected dataset.

6.2.2 Description

The **RoboDA** dataset contains grayscale images of size 28x28 pixels, consisting of a number from zero to nine (ten classes in total). The image content is kept simple so that the evaluation of DA methods is subject to their adaptation ability rather than to their ability to represent the content itself. Further details regarding each domain are as follows:

The Real-World Domain

This domain comprises a training and a test sub-dataset. Both the sub-datasets comprise 100 images per class, amounting to a total of 1000 images for each sub-dataset. Every image has an associated label provided alongside.

The Simulated Domain

The dataset belonging to this domain has 2,000 images per class, amounting to a total of 20,000 images with corresponding labels provided alongside.

Every image, in each of the classes within both the domains, is stored (after preprocessing as per Figure 6.13) as a normalised matrix containing pixel intensity values, ranging from 0 to 1, as a floating point number. Figure 6.2 shows a glimpse of the **RoboDA** dataset. The simulated images appear blurred due to pixelated edges, unlike the relatively sharper real-world images. Some of the images are also substantially occluded due to the movement of the robot. Despite both semantic and visual similarities in the images from the two domains, the **RoboDA** dataset provides a challenging landscape for many knowledge transfer techniques that are detailed in the subsequent section.

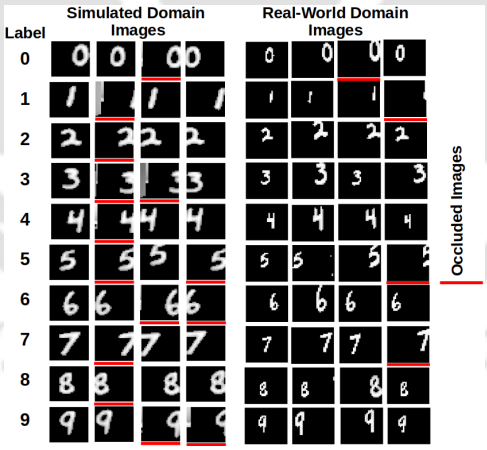


Figure 6.2: A glimpse of the RoboDA dataset

It may be noted that while one may choose to apply image augmentation techniques to create arbitrary noise-infused datasets, the applicability of such data for real-world robot vision is limited due to the presence of several sources of noise such as lighting, motion, orientation, speed, battery power, etc. The **RoboDA** dataset aids by providing images collected from robots that incorporate actual noises experienced by the robot both in the simulated and the real world. Thus, unlike in the case of image augmentation, the noises are neither exaggerated nor under-represented but kept to their actual amount.

6.3 Experimental Analyses and Results

In order to analyse the quality of the **RoboDA** dataset, we conducted multiple experiments, that can also serve as preliminary benchmark results. The former set of experiments compares naive approaches to knowledge transfer such as TL and fine tuning using the **RoboDA** dataset. This is done to analyse the degree of difficulty this dataset poses to the naive approaches. This is followed by a set of experiments comparing various DA methods applied to the dataset, depicting how much performance gain, if at all, the existing DA methods can achieve. In the latter part of the section, we compare the seemingly close dataset, MNIST, with **RoboDA** for the DA task. The section ends with a qualitative analysis of the distribution of the **RoboDA** dataset and how DA impacts this distribution space, using the t-SNE method [198]. All the DNN models were implemented using Keras [66] and Tensorflow 2 [67]. The models were trained and executed using a *Cloud* GPU (NVIDIA T4(x2)). For all the experiments, unless otherwise specified, the simulated data, D_S (20000 samples), acted as the source domain and the real-world data, D_R (1000 training and test samples each) acted as the target domain.

6.3.1 Comparing performances using RoboDA for Naive Methods of Knowledge Transfer

The classification accuracies of the following methods were compared for the test sub-dataset of the D_R:

1. ***R2R***: Training and testing on the training and the test sub-datasets of the D_R, respectively.
2. ***S2R***: Training using the D_S and testing using the test sub-dataset of the D_R.
3. ***S2R***$_{FT}$: Training using the D_S, fine tuning and testing using the training and the test sub-datasets of the D_R, respectively. Fine tuning involved retraining the whole model.
4. ***S2R***$_{TL}$: Training using the D_S, again training on the training sub-dataset of the D_R using TL and testing using the test sub-dataset of the D_R.

The above methods used the CNN model (depicted in Figure 6.3), similar to the classifier models utilised in existing DA methods [199]. In the $S2R_{TL}$ method, all three convolutional layers were frozen during TL. The model for each experiment was trained for 100 epochs with a batch size of 32. For each method, five experiments were conducted to avoid any stochastic observation. Figure 6.4 shows the classification accuracies obtained on the test sub-dataset of D_R using each of the methods, for all the experiments. All the methods for a given experiment number in Figure 6.4 shared the same base model, thus, starting with identical parameter values.

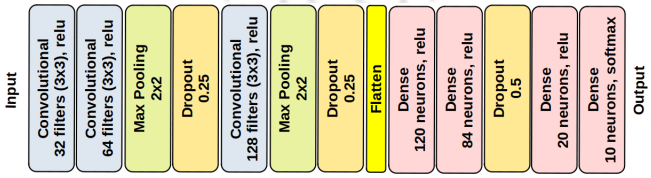


Figure 6.3: CNN model used for knowledge transfer methods

It can be observed from Figure 6.4 that generally, the R2R method delivers the least performance, due to the small size of the training dataset. By utilising the large simulated dataset for training, S2R boosts the accuracy, without even using any real-world training data. However, a

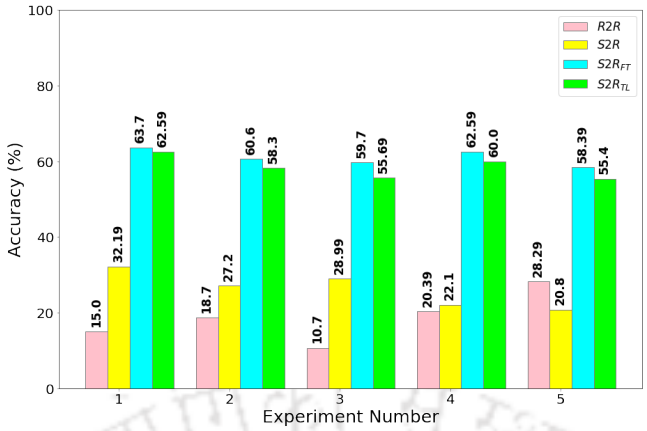


Figure 6.4: Comparing classification accuracies of naive Knowledge Transfer methods for RoboDA

substantial accuracy boost is only observed in the case of S2R$_{FT}$ and S2R$_{TL}$ methods, that utilise training data from both domains. While these experiments justify the requirement of simulated data and knowledge transfer approaches, the resultant accuracy still stays below seventy per cent. This can be ascribed to the difference in the underlying feature distribution of data of both domains of the **RoboDA** dataset, which otherwise seem similar. This validates the quality of the dataset which provides a complex feature landscape to optimize. Thus, further analysis using DA methods was conducted as detailed in the subsequent section.

6.3.2 Comparing performances using RoboDA for Domain Adaptation based Methods

DA methods specifically focus on bringing the feature space of different domains closer. This can potentially lead to better classification performance. Thus, the following DA-based methods were analysed to find their classification accuracies on the test sub-dataset of the D_R:

1. ***UDAB***: This method, as proposed by Ganin and Lempitsky [191], uses a DNN model comprising a deep feature extractor, a deep label predictor and a domain classifier. Training the model on the **RoboDA** dataset yields domain-invariant features for both the D_S and the D_R. The classifier can then be used to classify data from any one of the two domains.

2. ***UDSDAG***: This method, as proposed by Motiian [199], utilises a Siamese Neural Network (SNN) architecture that brings the embeddings from the D_S and the D_R together to make them indistinguishable. It utilises this embedding space to perform domain-agnostic classification. A penultimate dense layer with twenty neurons was added in the classifier branch of this method to make it equivalent to other methods.

3. ***CycleGAN***: This method is based on the CycleGAN model proposed by Zhu [163]. It involves the usage of CycleGAN to convert images from the D_S to the D_R. The converted images, now larger in number, are used to train the classifier.

These methods are extensively used for DA and at the same time vary substantially in their approach, thus exposing the **RoboDA** dataset to diverse and contemporary challenges. The classifier of the UDSDAG and the CycleGAN methods was similar to the one shown in Figure 6.3. It was trained for 100 epochs with a batch size of 32. For UDSDAG, 100 samples per class of D_S

were used from the source domain (total 1000 samples). Five experiments were conducted for each method. Figure 6.5 shows the accuracies obtained across all the experiments. It can be observed that both UDAB and CycleGAN struggle to achieve good accuracies due to the small size of the real-world data which does not allow good adaptation towards that domain. The low performance of CycleGAN is due to the poor quality of the generated images which is elucidated further in section 6.3.7. The UDSDAG method achieves a good performance due to the underlying SNN structure that can handle data scarcity. The large difference in the accuracies of the discussed DA methods highlights the role of **RoboDA** in challenging the DA methods from various aspects (size and noise) that even certain popular DA methods, such as CycleGAN, cannot tackle well.

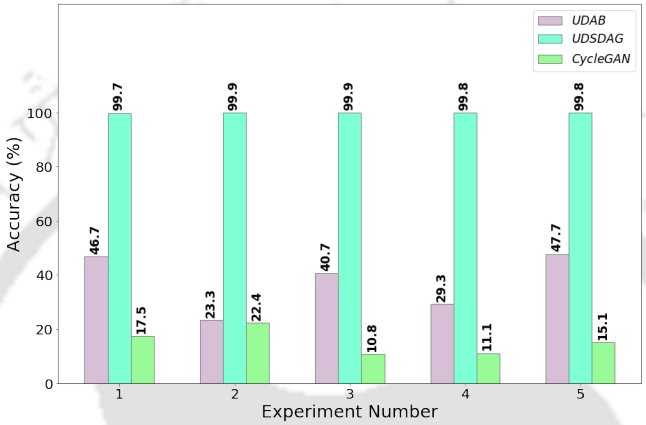


Figure 6.5: Comparing classification accuracies of Domain Adaptation methods for RoboDA

6.3.3 MNIST vs RoboDA for Domain Adaptation in Robot Vision

In this section, we delve deeper into the **RoboDA** dataset to analyse the utility of its simulated domain. Our dataset is apparently close to the MNIST dataset, which is often used to evaluate DA methods [188]. Thus, we analyse the impact of the two datasets for DA in robot vision by comparing S2R, S2R$_{FT}$ and UDSDAG with S2R$_{(MNIST)}$, S2R$_{FT(MNIST)}$ and UDSDAG$_{(MNIST)}$ where the latter three methods differ by using the MNIST dataset in place of D_S. However, instead of the full training data of MNIST, 20,000 training samples were used, to match the size corresponding to the D_S in **RoboDA**. Five experiments were conducted for each method. Figure 6.6 shows the accuracies obtained across all the experiments. As can be observed from the figure, the involvement of MNIST in the training process proves to be detrimental to the classification performance of the S2R$_{(MNIST)}$ and the S2R$_{FT(MNIST)}$ methods. Unlike the MNIST dataset, the **RoboDA** dataset comprises a more realistic pool of samples for the simulated domain, which is rife with noises similar to those encountered in the real-world test domain. The MNIST dataset fails to contribute substantially when tested against a target domain having noisy images. However, UDSDAG is able to deliver a good classification performance even with the MNIST dataset due to the underlying SNN structure.

6.3.4 Qualitative Analyses of the RoboDA Dataset

This section presents the qualitative analyses of the **RoboDA** dataset by using the t-SNE method [198] (perplexity value of 40) to create 2-D maps of the data and its features.

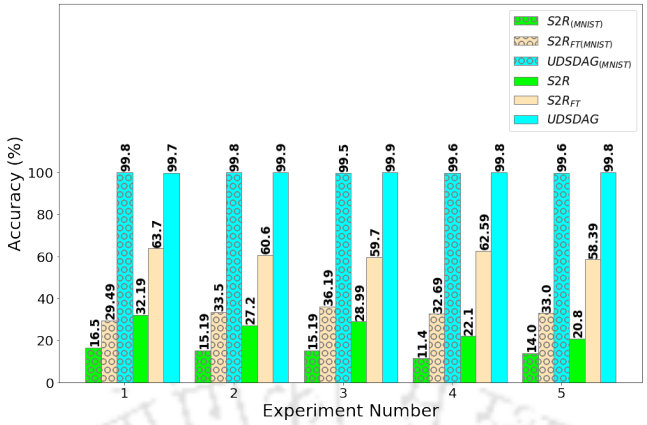


Figure 6.6: Comparing classification accuracies of Domain Adaptation methods for RoboDA and MNIST

Visualising the Data Distribution of RoboDA

Owing to the involvement of a mobile robot in the data acquisition process, the **RoboDA** dataset contains a substantial amount of real word noise. Moreover, the dataset is high-dimensional. Visualising such a dataset is a non-trivial task. Thus, we utilise the t-SNE method to visualise **RoboDA** in a low-dimensional space, as shown in Figure 6.7. Each colour in Figure 6.7 represents samples belonging to a particular class label. The t-SNE method plots the data points probabilistically, based on their relative similarities. Similar data points stay together while dissimilar ones move apart.

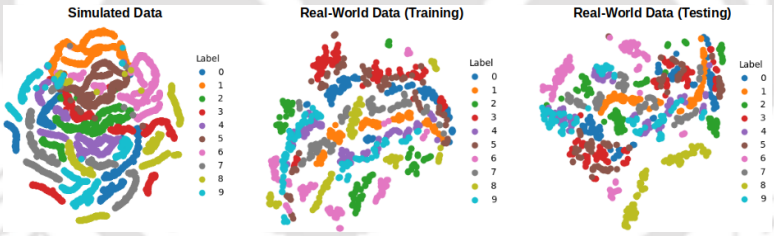


Figure 6.7: A t-SNE based 2D visualisation of the RoboDA dataset

The following observations can be made from the t-SNE maps of the Figure 6.7:

1. Data points of certain classes are constrained to specific regions in the map (class '1'). Such data points share limited similarities with others. This is consistent with the class semantics, where class '1' is similar to a very limited set of other classes such as class '7'.

2. Data points from some of the classes are spread all over the maps (class '9', '0' and '8'). This implies their similarity with many other classes.

3. The simulated data has more samples thus, depicting more continuity in the corresponding map.

4. In all three maps, the data points of all the classes are heavily intermingled. This highlights the lack of stark demarcation amongst inter-class samples. However, one can also observe

intra-class clustering of data points, highlighting strong intra-class similarities. Thus, the dataset presents a complex distribution.

5. Across the domains, the relative positions of the data points of classes do not stay the same. For instance, in the simulated domain, data points of the classes '1', '9', '6' and '5' are close. However, this trend is not strictly maintained in the real-world domain, contributing to the domain transfer issue.

6.3.5 Visualising the impact of DA on RoboDA

This section visualises the impact of UDSDAG, on the **RoboDA** dataset (Figure 6.8). For each of the three data portions (the simulated, the real-world training and the real-world test sub-datasets) 1000 samples were used for this analysis. The upper part of the figure shows the t-SNE distribution of the samples before DA. As discussed earlier, the distribution suffers from inter-class overlaps with differences in the relative positions of classes across domains. However, these issues are addressed by the usage of the UDSDAG method. To visualise this, the trained UDSDAG model was used on the samples of each of the three data portions. The corresponding features from the penultimate dense layer of the model were extracted and fed to the t-SNE method. As can be observed in the lower part of the figure, UDSDAG isolates the features of the **RoboDA** dataset, thus removing inter-class overlaps. Moreover, it also manages to retain the relative positions of features of each class, across all the domains. This explains the good performance of the UDSDAG method on the **RoboDA** dataset.

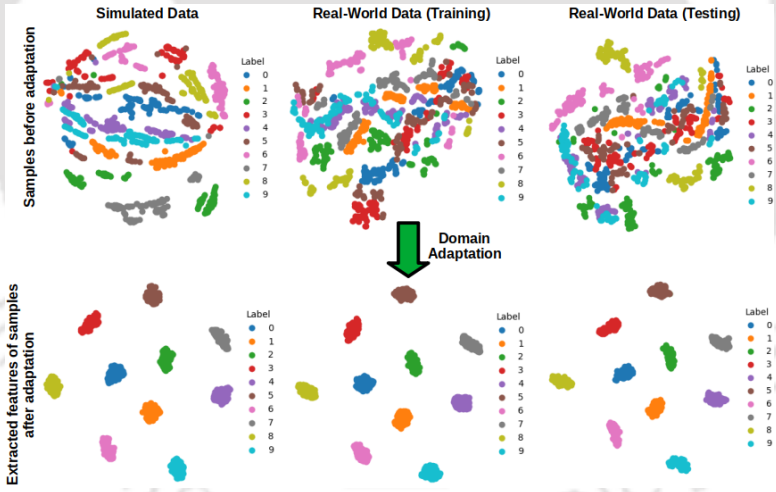


Figure 6.8: A t-SNE based 2D visualisation of the impact of domain adaptation on the RoboDA dataset

6.3.6 Comparing the impact of the simulated data of RoboDA against noise-induced data

One may suggest using artificial noise in conjunction with the real-world data of RoboDA to create the simulated counterpart, marginalizing the utility of the data from the simulated domain present in the **RoboDA** dataset. However, the mere inclusion of artificial noise is not always an effective option to compensate for the lack of data. To assess this claim we conducted DA experiments, using two methods with varying complexity, for two kinds of datasets. The first dataset was

identical to the **RoboDA** dataset, being used as it is. The second dataset involved replacing the simulated domain data with artificially created 'noisy' data, created by adding noise to the real-world domain data of the **RoboDA** dataset, and bloating it up to the size of the simulated domain data. The training data in both datasets amounted to 21,000 samples formed by combining the simulated/noisy source data with the training data from the target real-world domain of the **RoboDA** dataset. The test data comprised 1,000 test samples from the target real-world domain of the **RoboDA** dataset. The noise addition process involved adding a Gaussian random noise to each image sample which was followed by clipping the resulting values to keep them within the range of zero to one. The two methods used for DA are explained briefly as follows:

1. Domain-Adversarial Neural Network (DANN): This method leveraged the DANN architecture [188] for the DA task, which uses adversarial training to improve model robustness across different domains. The DANN architecture consists of a shared feature extractor followed by two classifiers: one for the primary task (classification) and another for distinguishing between the source and target domains. By minimizing the classification loss while simultaneously maximizing the domain classification loss, the model learns to extract domain-invariant features, thereby enhancing its generalization capabilities to the target real-world data. The neural network used in this technique used four convolutional layers with increasing filter sizes (64, 128, 256, and 512), complemented by dropout layers to mitigate overfitting. Maxpooling layers were also used after each convolutional layer.

2. Deep Neural Network Classifier (DNN): This method utilized a standard DNN for the DA task. The DNN model was trained to classify the training dataset that comprised the source and the training data of the target domain, allowing it to learn the features of both domains. The neural network used in this technique used two convolutional layers with increasing sizes (32, 64), followed by two dense layers (64, 10), out of which the final layer was a 'softmax' layer. Maxpooling layers were also used after each convolutional layer.

For both methods, the model was run for fifteen epochs. Five experiments were conducted in each method, for both datasets. This amounted to twenty DA experiments in total. Figure 6.9 shares the accuracies obtained on the real word test data for each method across each type of dataset. It can be observed from the figure that the usage of noisy data in place of the simulated data (depicted by dashed lines) lowers the corresponding accuracy in both methods, for all the experiments. Thus merely adding noise instead of capturing images from a simulated mobile robot in a simulated environment, is detrimental to the DA performance of the methods. This highlights the role of the simulated data in supporting the DA task, validating its presence in the RoboDA dataset.

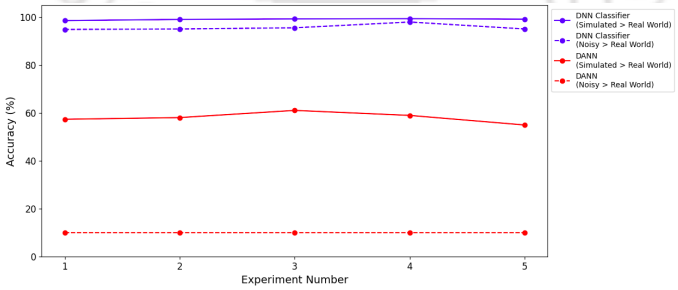


Figure 6.9: Comparing classification accuracies of different domain adaptation methods using simulated and noisy data as the source for RoboDA

Section 6.3.8 elucidates the intra-class metrics of the dataset as an ancillary analysis.

6.3.7 Analysing the failure of CycleGAN on the RoboDA dataset

CycleGAN [163] is a generative Deep Learning (DL) technique that allows image translation across domains. A CycleGAN model trained on two domains can convert images from one domain to the other. However, CycleGAN struggles in scenarios where the dataset size is small or the size of both the domains is not comparable [200]. This is the reason CycleGAN could not perform well with the **RoboDA** dataset. Figure 6.10 depicts three experiments performed using CycleGAN with the **RoboDA** dataset. For each of the experiments, 20,000 simulated samples of the **RoboDA** dataset were used as the source domain and 1,000 samples of the training sub-dataset were used as the target domain for training the CycleGAN model. The trained CycleGAN model was then used to generate 21,000 samples for each of the experiments. This generated dataset is further used to train a CNN-based classifier. The classification accuracy of this classifier is then found for the test sub-dataset of the **RoboDA** dataset. As can be observed from the Figure 6.10, the generated datasets shown for each of the three experiments contain a lot of noise. Thus, their features are too distant from the test sub-dataset of the **RoboDA** dataset. This explains the low accuracies obtained by the corresponding classifiers trained on each of the generated datasets.

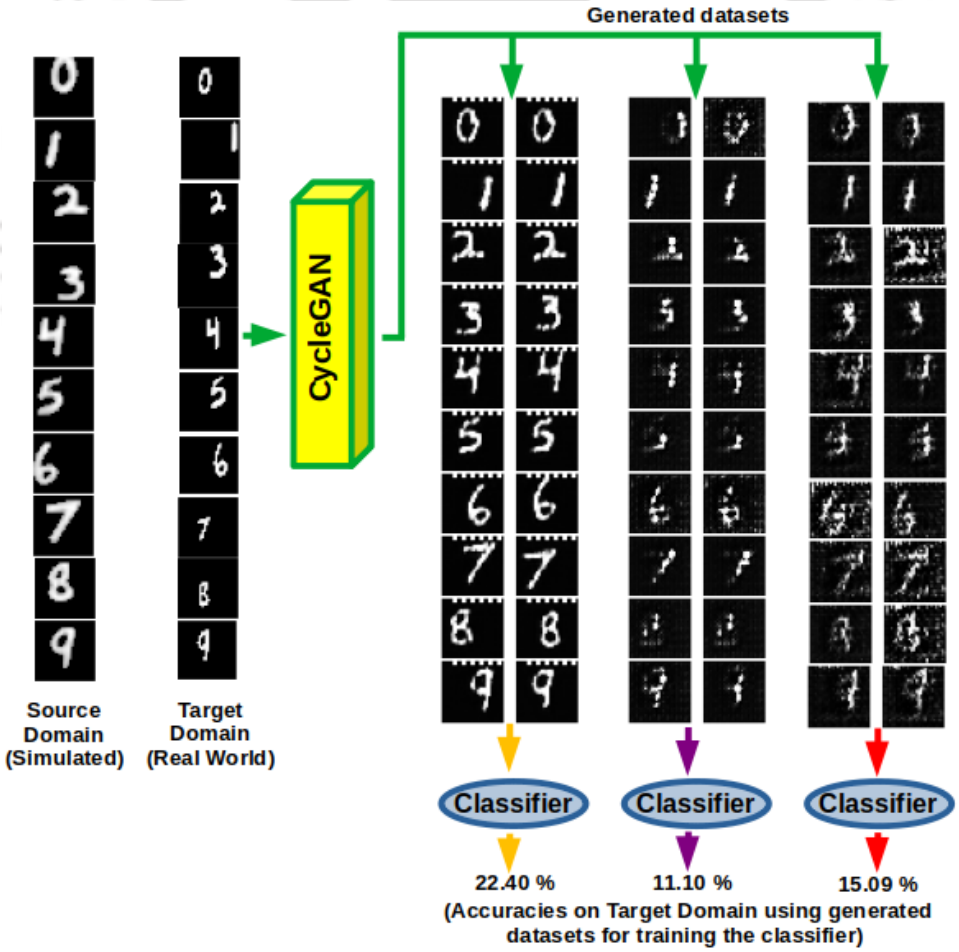
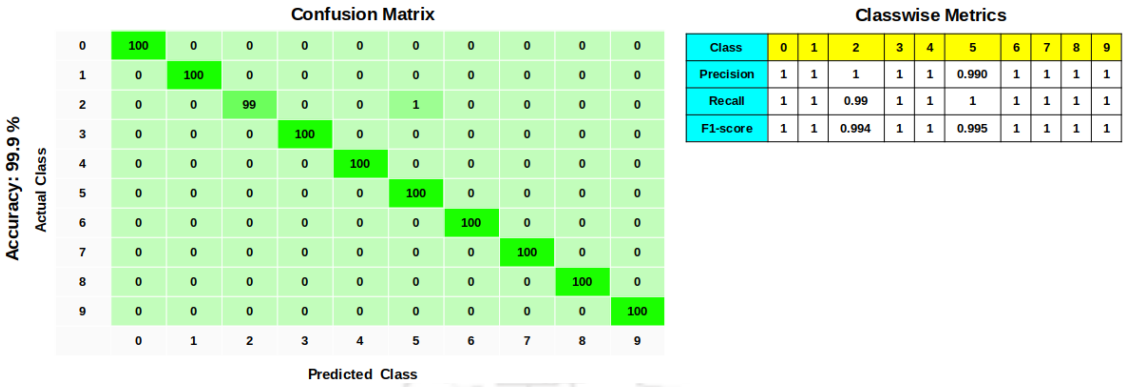


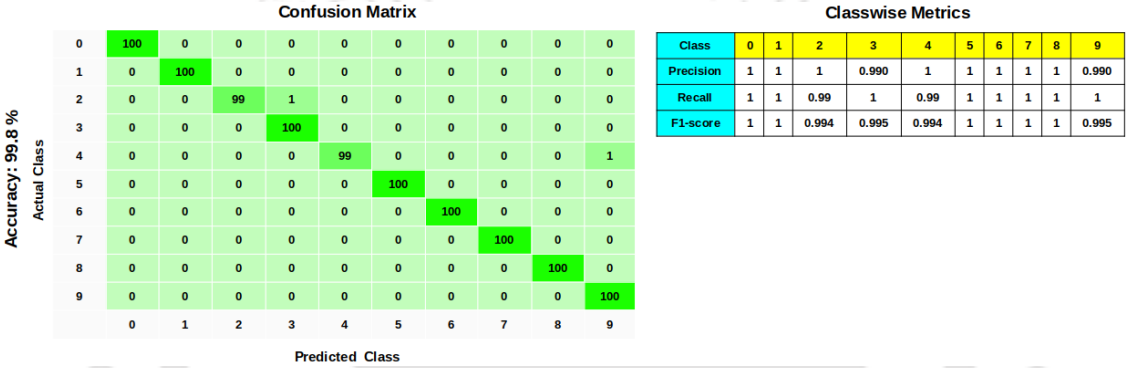
Figure 6.10: A schematic of 3 classification experiments performed with CycleGAN using the RoboDA dataset

6.3.8 Intra-class analysis of the RoboDA dataset

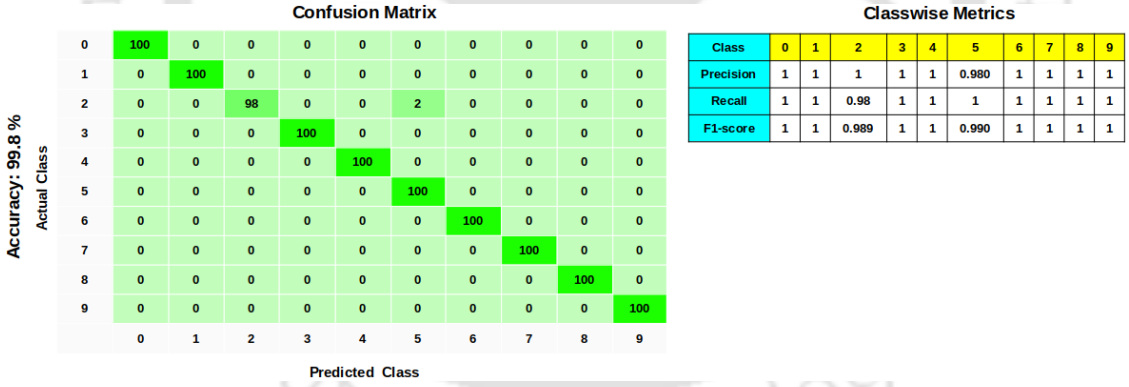
This contribution shares the benchmark classification accuracies on the **RoboDA** dataset using various methods. While accuracy serves as a guiding metric for analysing the classification performance, it does not provide a comprehensive quantification of the performance for each class of the underlying dataset. Thus, this section shares the experimental analyses conducted with the class-specific performance metrics. Three experiments were conducted using the UDSDAG method with the **RoboDA** dataset. Each experiment used 1,000 samples from the simulated domain and 1,000 samples of the training sub-dataset of the real-world domain for training. The model was then tested on the 1,000 samples of the test sub-dataset of the **RoboDA** dataset. For each experiment, the accuracy, the confusion matrix, the precision, the recall and the F1-score were found as shown in the Figure 6.11. The accuracies corresponding to all the experiments are high, due to the usage of the Siamese Neural Network in the UDSDAG approach that expands the dataset. However, the class '2' and '4' lag behind in the recall metric, due to them being wrongly predicted as the class '3', '5' and '9' respectively. Consequently, the classes '3', '5' and '9' have low precision due to incorrectly being mapped to other classes. The holistic metric, F1-score is thus low for the classes '2', '3', '4' '5' and '9'. The confusion matrix for each of the experiments details the predictions made for all the classes (100 predictions for corresponding samples of each class), which can be referred to understand in detail, the correct and incorrect predictions. The Figure 6.12 visualises the impact of using the UDSDAG method based DA on the **RoboDA** dataset in the aforementioned experiments. As can be observed from the figure, for each experiment, the distribution of the samples before adaptation (upper half) stays the same. However, after using the UDSDAG method, the feature distribution of the penultimate layer of the classifier varies across the three experiments (different relative locations of the class-wise clusters across experiments). This can be attributed to the difference in the parameter values for the trained model of the three experiments. Nevertheless, within each experiment, the relative location of clusters across the three maps stays similar with clear isolation of each cluster. After the UDSDAG method is trained for DA, its constituent layers are able to extract high-level features for the given input samples. Thus, the features extracted by the penultimate layer of the UDSDAG method for the given dataset are more refined in terms of decreased inter-class overlap and can be used to obtain a version of the dataset with clearly isolated class-wise datapoints. It may be noted that although the UDSDAG method used in this contribution sets a high accuracy benchmark for the **RoboDA** dataset, it consumes a substantially high amount of data (2,00,000 pairs of source-target domain images when provided with 1,000 images each from the source and the target domains), computation and time (seven hours of GPU access per experiment) to achieve this. We encourage the potential users of this dataset to come up with methods that can achieve similar performance but with substantially lower resources.



(a) Experiment 1

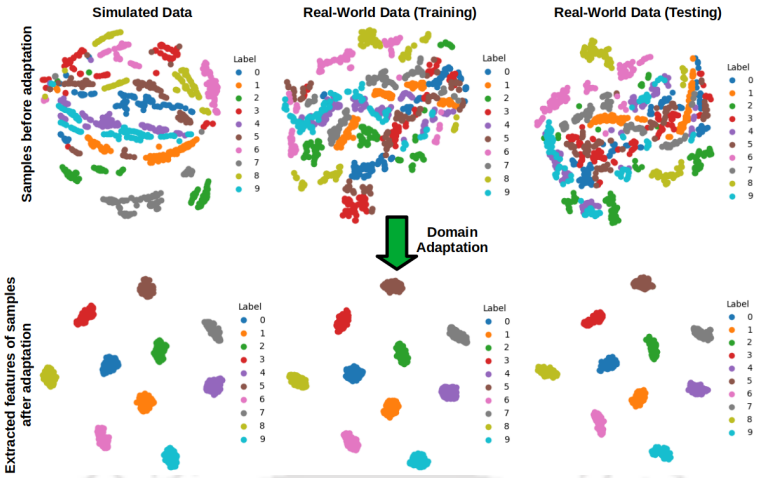


(b) Experiment 2

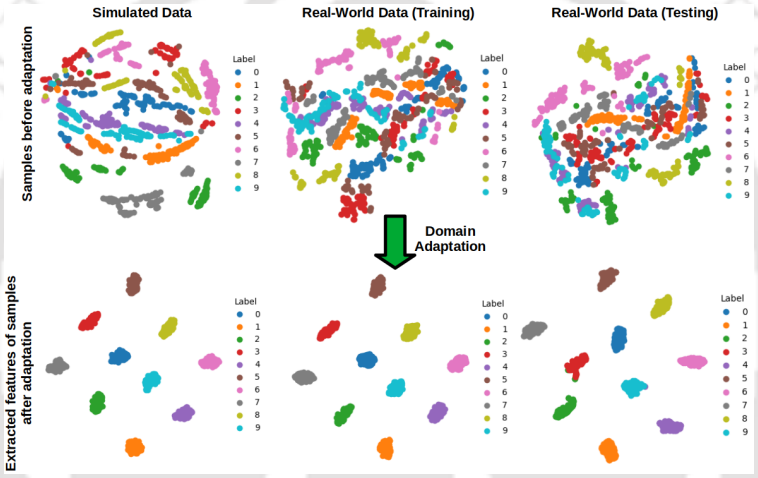


(c) Experiment 3

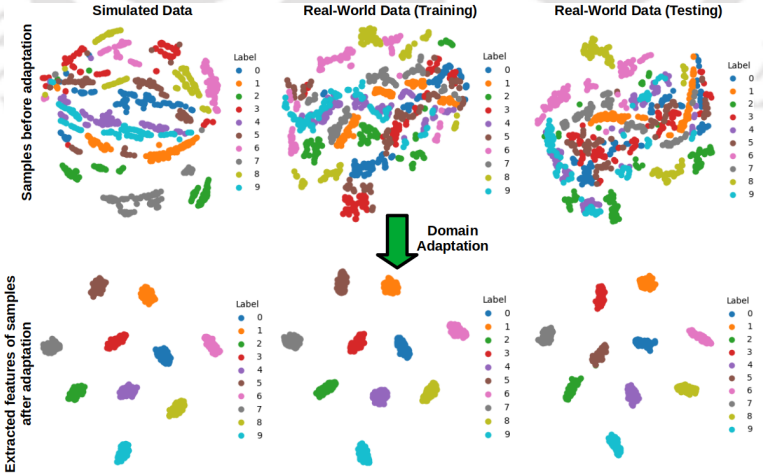
Figure 6.11: Confusion matrix and related metrics for experiments performed with UDSDAG using the RoboDA dataset



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 6.12: t-SNE based visualisation of the impact of DA across three experiments performed using the RoboDA dataset

6.3.9 Prepocessing the raw image captured by the mobile robot

The robot used in the experiments to create the **RoboDA** dataset uses a generic camera [134] to capture the images. The captured images are colored and thus, have three channels. However, in order to make the dataset simpler, each image is saved as single channel grayscale image with a size of 28x28 pixels. To achieve this, the raw image captured by the robot is subject to the preprocessing steps shown in the Figure 6.13. The raw image is first resized to a size of 28x28 pixels. The resulting image is then converted to a single channel grayscale image. The image is further refined by using the bitwise not and the Otsu thresholding [201] methods. It may be noted from Figure 6.2 that even after preprocessing, in addition to the occlusions, the images in the real-world domain also vary in the morphology and size of the numbers. This is due to the imperfect motion of the robot and the sensor noise in the real world due to which the robot stops at varying distances from the pallets for capturing the samples. Such variations are not observed in the simulated domain. The availability of both the domains within the **RoboDA** dataset makes it an ideal candidate to evaluate real-world robot vision DA methods.

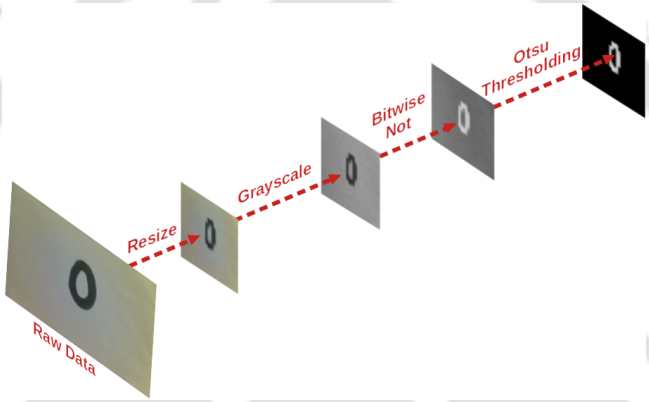


Figure 6.13: Preprocessing steps used for refining the raw image captured by the mobile robot

6.4 Chapter Summary

This contribution presents **RoboDA**, a dataset for evaluating DA methods for robot vision. The apparatus used for acquiring the dataset involves standard and open-source resources, allowing reproducibility of the experiments. It provides data for both the simulated and the real-world domain. Despite the ease of use, the dataset has a challenging feature landscape as highlighted in the experimental section, wherein the naive knowledge transfer methods and some of the DA methods, struggle to adapt to the target domain. This is attributed to the usage of an autonomous mobile robot for data acquisition, which exposes the dataset to noisy scenarios. However, given an appropriate DA method, the **RoboDA** dataset can deliver good performance, as discussed later in the section. Thus, this dataset acts as an appropriate benchmark to filter out good DA methods. The simulated domain of the dataset has also been shown to be a better choice than the seemingly close MNIST dataset for DA. This dataset is envisioned to allow easy evaluation of robot vision DA methods, even with devices having limited computational power. The content of the images is kept simple, allowing more attention to the DA part.

7

Onboard Class Incremental Learning for Resource-Constrained scenarios using Genetic Algorithm and TinyML

7.1 Introduction

The work described so far in the thesis has tried to address issues related to real-world deployments of ML algorithms, specifically in resource-constrained devices. Practical implementation of the solutions to each of these issues were also described and investigated. Most of these proposed solutions pertained to the field of computer/robot vision with a fair amount of constraints in the computing resources. However, the challenge posed by the very first contribution described in chapter two, still remains unaddressed. The *Smart Patch* required to use an extremely resource-constrained microcontroller with a small form factor that can perform the task of gesture recognition using ML. The work described in this chapter addresses this challenge by making onboard ML operations run on an extremely resource-constrained microcontroller. This work also extends the scope of the thesis by dealing with time series domain data to perform gesture recognition. From an algorithmic point of view, this contribution caters to the issue of *Class Incremental Learning*, which is a more common but complex issue to deal with. With DL as its primary driving force, AI has accelerated a variety of domains, including computer vision, natural language processing, finance, healthcare, etc., [202]. DNN models, in general, need to be trained on computationally rich systems. While such pre-trained models, generally deployed on *Cloud* servers, deliver an impressive performance, the training overheads involved with them limit their capabilities when it comes to real-world low-resource scenarios. A recent survey by *Gartner* [31] suggests that by the year 2025, more than 55 per cent of all the data analysis carried out using DNNs will be accomplished at the point of capture in edge systems. While this acts as an indicator of the increasing adoption of DL approaches, it also highlights a strong need for onboard learning capabilities on EDs. TinyML, still in its nascent stages, is one such tool for onboard deployments of ML models [203]. However, most of the existing TinyML approaches only facilitate onboard inference, without any onboard learning capabilities [204]. Such static pre-trained model deployments struggle to deliver good performance when faced with dynamically changing data or data from new classes. A sustainable way to deal with such non-stationary data is to accommodate unseen data, incrementally. CIL is one such technique that allows the addition of the knowledge of new data into an existing model [205]. However, the confluence of TinyML and CIL has not been explored to its full extent. Pursuing such work for domains involving multi-dimensional and sequential data, such as time-series data,

becomes even more challenging due to the complex underlying optimisation space. GAs allow easy simultaneous control over multiple optimisation landscapes [206], and can thus be leveraged to handle optimisation for complex data. This contribution proposes a method to club TinyML and GA to allow CIL on resource-constrained EDs. The method is generic in nature and has been used in conjunction with time-series data, for a gesture recognition task. The method allows learning and classification, in tandem with the newly provided gesture class onboard an ED. The performance comparison with existing CIL strategies and its deployment as a real-world CPS, establish the efficacy of the proposed method. The primary contributions of the proposed work are:

- Exploring and establishing the usability of GAs for TinyML
- Highlighting the utility of GAs for CIL with time-series data
- Analysing the usage of a GA-based TinyML method on real-world low-resource deployments

7.2 Related Work

This contribution covers multiple areas to analyse the target problem, identify the research gaps and combine techniques from multiple domains to devise the solution. This section briefly describes each of the referred areas.

7.2.1 TinyML

TinyML is a budding field of ML that focuses on deploying ML models on resource-constrained EDs such as battery-powered microcontrollers, and embedded systems [203]. Shifting ML model deployments from the *Cloud* to EDs is advantageous in many scenarios due to reduced network bandwidth requirements, enhanced security and privacy, lower latency, energy and cost efficiency, etc. [207], in the latter case. The usage of TinyML spans multiple areas including environmental monitoring, healthcare, agriculture, anomaly detection, etc. [203]. Despite its widespread acceptance, TinyML still struggles with multiple issues; offline learning being one of them. Most of the contemporary TinyML models are trained offline before being deployed in the real world. Their onboard activities are limited to inference without any training capabilities [204]. Under a continuously changing real-world data distribution, the performance of the static model drops, a phenomenon often known as *concept drift* [82]. This calls for the integration of incremental learning (IL) approaches for TinyML. CIL can be used to handle concept drift by accommodating data from new classes in an online manner, as discussed in the following section.

7.2.2 Class Incremental Learning

DL techniques, in general, involve training a model using large datasets in an offline manner and deploying it for inference. However, the real world is rife with non-stationary data, where new data samples are available over time [208]. While retraining the pre-trained DL model on such new data seems intuitive, this practice often leads to catastrophic forgetting [208], leading to the knowledge of the old data being omitted from the model. CIL is an approach that tackles both the issues of concept drift and catastrophic forgetting by incrementally accommodating the knowledge of the data from the new class(es) while retaining those of the older classes. CIL has been realised in multiple ways including architectural, regularization and memory-based techniques [209, 205, 210]. However, the memory-based CIL techniques are known to outperform the others [209]. Latent Replays (LR), embodying a memory-based technique for CIL, use intermediate feature

maps, logits or activations of a pre-trained model to encode data into a low-dimensional space that can then be used to combine both old and new data in a computationally tractable manner [209]. This involves freezing the model at an intermediate layer and adapting the rest of the model for CIL [209]. Replays often comprise exemplars that are either samples, logits or activations stored from the previously observed data [205]. iCaRL [211] is one such technique that uses exemplars as a set of samples from each class. During the inference of the target data from a new class, iCaRL finds a prototype vector for each existing class using the exemplars. The class having its prototype vector closest to the vector of the target data is identified as the inferred class. While this method offers a simple IL approach, it suffers from multiple issues. It requires gradient-based retraining of the complete model for IL which is computationally expensive when deployed on EDs [212]. This also necessitates the usage of storing raw images as exemplars, as the feature vectors keep on changing due to retraining. Such an exemplar choice puts too much load on the memory requirements and makes it directly proportional to the input size. This makes it difficult to work with data of large sizes. Disabato and Roveri [204] specifically addressed IL for TinyML by utilising TL and the kNN method [213]. They utilised a DL-based feature extractor and a dimensionality reduction mechanism followed by the usage of a kNN classifier for allowing incremental learning. The usage of kNN hampers the speed of inference due to its *lazy* nature which is detrimental to real-time systems. Similarly, Support Vector Machines (SVM) [214] are often used as classifiers or intermediate components of the CIL pipeline used in conjunction with a DL-based feature extractor to accommodate CIL on EDs [204, 215]. Most of the existing CIL approaches deploy a classifier on top of a DL-based extractor without explicitly controlling the underlying optimisation space where the knowledge of the old and new data resides. Moreover, the kNN, the SVM and the ANN based classifiers either increase the inference latency or are computationally heavy for the EDs [212, 204]. Another issue is that the bias of the CIL model towards the newly available class data demands greater control over the optimisation route of CIL to avoid such biases [216]. With limited computational overhead, a GA, on the contrary, allows better control over the optimisation space and thus forms an ideal candidate in the domain of TinyML and CIL. A brief description of GA follows.

7.2.3 Genetic Algorithms

GAs are metaheuristic algorithms that leverage a population of solutions to traverse an optimisation space [217]. GAs have been applied across a wide array of fields including operation management, multimedia, wireless networking, agriculture, etc. [217]. GAs have also been used for facilitating DL model deployment on resource-constrained devices by optimising the topology of DL models as a Neural Architecture Search technique [157]. GAs are known to reach better ANN optimisation spaces, faster than the gradient-based methods [43] with better chances of evading the local optimum [44]. In addition, GAs offer unique optimisation features such as control over diversity, topology evolution, resistance to vanishing gradient (crucial for tasks involving sequential data), etc. [218]. Most importantly, the fitness functions employed in GAs allow explicit steering control across the optimisation space. This makes GA a good choice for enabling onboard CIL by employing them on top of DL models.

7.2.4 Gesture Recognition

Gesture recognition is one of the most widely investigated areas in the field of Human-Computer Interaction [219]. With technological advancements in sensors and embedded technologies, gesture recognition has rightfully become one of the most active topics. This may in part be credited to its

widespread applications across fields including sign-language communication, digital interaction, military coordination, virtual gaming, elderly/disabled support, etc. [219, 220]. Inertial Measurement Units (IMU) are one of the most widely accepted non-intrusive gesture recognition tools due to their tolerance to environmental dynamics, namely lighting conditions, line of sight, and detector proximity [219]. IMUs track the acceleration data of the subject wearing the IMU [219]. Faisal *et al.* [219] created a data glove comprising an MPU6050 IMU sensor and a microcontroller among other circuitry to recognise hand gestures. However, their approach does not involve IL. Naosekpam and Sharma [221] utilised an MPU6050 IMU sensor and a microcontroller to recognise hand gestures. However, the recognition task was outsourced to a personal computer and did not involve IL. Similar efforts have been made by multiple researchers to use IMU data for a variety of recognition tasks [222, 223, 224, 225]. However, very few of these allow IL to cater to new data with fewer works combining CIL and TinyML for onboard learning. For the IMU-based tasks dealing with sensitive applications such as monitoring and surveillance [226, 227], the requirement of onboard CIL becomes indispensable.

7.3 Methodology

The challenges observed across the discussed survey highlight the requirement of the following features in the intended solution:

1. It should be deployable on a resource-constrained ED with limited storage and computational requirements,
2. It should have the capability to handle time-series data,
3. It should allow onboard CIL.

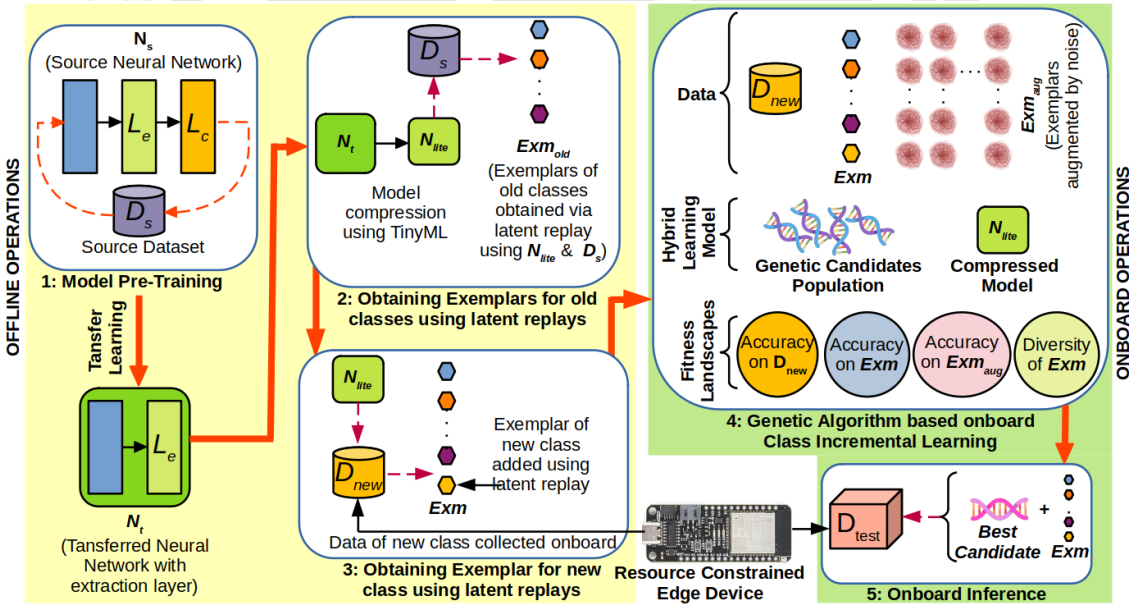


Figure 7.1: A schematic of the proposed CIL method using GA and TinyML

Adhering to the aforementioned requirements, we propose a method, as depicted in Figure. 7.1, whose steps are discussed as follows:

7.3.1 Model Pre-training and Transfer

This step involves training a DNN model as the source network, N_s, using an existing multi-class source dataset, D_s, comprising the training samples and the corresponding labels. The penultimate and the last layer, L_e and L_c, respectively, of the N_s, are identified as the extraction and the classification layers, respectively. The raw data, D_s, undergoes data wrangling to make it consumable for training the N_s using the general gradient-based DL method. All the trained layers of N_s, except for the L_c layer, are transferred to another model, N_t. The N_t, with all the parameters frozen, acts as an encoder network that converts input samples to their equivalent low-dimensional feature vectors, to be used for producing the latent replays.

7.3.2 Obtaining Exemplars for old classes using latent replays

This step is used to obtain exemplars for each class of the old data. The N_t is firstly compressed, using TinyML techniques, to obtain the N_{lite}, a lightweight model. For each class, old_i, of the D_s, the N_{lite} is used to obtain the corresponding exemplar, Exm_{old_i}, as per the equation (7.1) below.

$$Exm_{old\_i} = \frac{1}{num_i} \sum_{j=1}^{num_i} N_{lite}(S_j) \tag{7.1}$$

where S_j is a sample belonging to the class old_i whose exemplar, Exm_{old_i}, is calculated as the average of the feature vectors of all the samples of the same class, obtained using the N_{lite}. This step concludes by yielding the set Exm_{old} that comprises exemplars of each class of the D_s, with the corresponding labels.

7.3.3 Obtaining Exemplar for new class using latent replays

To incrementally learn data from a new class, a set of training samples, D_{new}, comprising the training samples and the corresponding labels, is obtained from the resource-constrained ED. The exemplar of the new class is obtained using equation (7.1) in conjunction with the D_{new} and the N_{lite}. This exemplar is appended to the Exm_{old_i}, with the corresponding label, to obtain the set Exm, which acts as the latent replay for CIL.

7.3.4 Genetic Algorithm based onboard Class Incremental Learning

This step commences the onboard CIL process. The data available for onboard Learning includes the D_{new} and the Exm. However, such a data corpus is dominated by the data of the new class which increases the chances of the model being biased towards the new class while using CIL. To curb this issue, an augmented dataset based on the Exm is obtained by adding noise to the Exm, as per the equation(7.2).

$$Exm_{aug\_i} = \bigcup_{i=1}^{num\_aug} Exm_i * \varphi \tag{7.2}$$

where num_aug number of augmented samples are obtained for the exemplar of each class i by element-wise multiplication of its exemplar, Exm_i, with a noise vector φ. φ is a uniform random distribution vector having the same size as Exm_i. The distribution range of φ is kept close to 1 so as to maintain the essence of Exm_i in Exm_{aug_i}. Augmented exemplars, with the corresponding labels, of all the classes are combined to form the augmented set, Exm_{aug}.

With these datasets in place, a GA-based optimisation routine is employed, in conjunction with N_{lite}, by using a genetic population of candidate solutions, Φ. Each candidate, Φ_i, is a matrix of size $|L_e|$*num_{exm} where $|L_e|$ is the size of the extraction layer and num_{exm} is the number of exemplars in Exm. Owing to the complex nature of the CIL task, the GA routine proceeds by optimising the fitness of the Φ across four fitness landscapes. The optimisation process spans across multiple generations of Φ. In each generation, for each Φ_i, fitness is recorded and evaluated for the following four objectives:

1. **Accuracy on** D_{new}: As D_{new} comprises data from the new class, encouraging Φ for achieving good accuracy on D_{new} is set as one of the optimisation goals. Given a sample S_{new} from the D_{new}, its corresponding embedding, $E_{S_{new}}$ is obtained by feeding it to the N_{lite}. A masked embedding is obtained from the $E_{S_{new}}$ by augmenting it with the Φ_i as per equation (7.3)

$$E_{masked} = E_{S_{new}} * \Phi_i \tag{7.3}$$

   where $*$ denotes the element-wise product.

   Similar masked embeddings are obtained for each exemplar of the Exm as Exm_{masked_i}. The class i of the masked exemplar Exm_{masked_i} having the least Euclidean distance from E_{masked} is inferred as the class of S_{new}. This is compared with the actual class of the S_{new} which is available as the training label. If the comparison yields a match, the inference is deemed accurate. Such an inference is done for each sample of D_{new} and the resulting accuracy percentage across all of its samples is recorded as the *novelty_fitness* for Φ_i.

2. **Accuracy on** Exm: This objective is used to pivot the fitness of Φ_i on exemplars by assessing the inference accuracy for all the exemplars. To achieve this, for each exemplar Exm_i, the Euclidean distance of its masked embedding Exm_{masked_i}, is found with the masked embedding, Exm_{masked_j}, of every other exemplar. The class of Exm_i is inferred as the class of closest Exm_{masked_j}. The accuracy obtained across all the exemplars is recorded as the *exemplar_fitness* for Φ_i.

3. **Accuracy on** Exm_{aug}: This objective helps Φ_i decrease the bias towards the D_{new} by introducing Exm_{aug} for the Φ_i to cater to. The accuracy for Exm_{aug} is obtained in the same manner as done in the case of the first objective. However, the Exm_{aug} is used in place of the D_{new} to provide the optimisation landscape for Φ_i. The accuracy obtained across Exm_{aug} is recorded as the *augmented_exemplar_fitness* for Φ_i.

4. **Diversity of** Exm: Chasing the accuracy over data samples is an intuitive practice for guiding the Φ optimisation routine. However, in exemplar-based CIL, maintaining the diversity of the exemplars by keeping them distinct from each other may push the CIL performance further by decreasing the overlap between exemplars of different classes. This last objective achieves this task by finding the discrimination factor, $Disc_{Exm_i}$, for each exemplar using equation (7.4)

$$Disc_{Exm_i} = \sum_{j=1, j\neq i}^{num_{exm}} d(Exm_{masked_i}, Exm_{masked_j}) \tag{7.4}$$

   where the $Disc_{Exm_i}$ for the exemplar Exm_i is found by summing up the Euclidean distance, $d()$, of its masked embedding, Exm_{masked_i} with the masked embedding, Exm_{masked_j}, of every other exemplar j. $Disc_{Exm_i}$ for all the exemplars are added together and recorded as the *discrimination_fitness* for Φ_i.

The final fitness, F_{Φ_i}, for Φ_i is obtained using equation (7.5)

$$F_{\Phi_i} = h1 * novelty\_fitness + h2 * exemplar\_fitness \\ +h3 * augmented\_exemplar\_fitness + h4 * discrimination\_fitness \tag{7.5}$$

where $h1$, $h2$, $h3$ and $h4$ are the hyperparameters used to control the impact of each type of fitness.

The F_{Φ_i} fitness calculation for all the candidates of Φ is followed by the *selection* step of GA wherein the best half of the Φ population, based on the fitness, F_{Φ_i}, is chosen as the parent population, Φ_{parent}.

Φ_{parent} is further chosen for the *crossover* operation wherein each child, $Child_i$, is obtained by using two parents, $Par1$ and $Par2$, from Φ_{parent}. $Par1$ and $Par2$ are oppositely positioned in Φ_{parent} sorted by fitness. The former half content of $Child_i$ is copied from $Par1$ and the latter half content is copied from $Par2$. As many $Child_i$s are generated as the members in Φ_{parent}. All the child and parent Φ_is are merged into a single set, $\Phi_{interim}$. Lastly, a fixed number of mutations of randomly picked candidates from $\Phi_{interim}$ is conducted. The mutation process involves the addition of uniform random noise to the selected candidates. The whole process discussed so far under the GA optimisation routine is repeated for the desired number of generations which completes the onboard CIL process. The Φ_i having the best fitness across all generations is chosen as Φ_{best} along with the N_{lite}, to be used for onboard inference.

7.3.5 Onboard Inference

To conduct the onboard inference for a test sample, D_{test}, N_{lite} is used to obtain the embedding E_{test} for the test sample. The embedding is masked as, E_{test_masked}, as per the equation (7.3) by using Φ_{best} in place of Φ_i. The Euclidean distance of E_{test_masked} is found with Exm_{masked_i} of each exemplar. The class whose Exm_{masked_i} is least distant from E_{test_masked} is inferred as the class of D_{test}. Algorithm 7 briefly describes the complete proposed CIL method.

The scalability of the method to new classes is ensured by a simple procedure. To accommodate a new class, an exemplar of the same may be calculated. This exemplar may be added to the Exm with the corresponding augmented data added to the Exm_{aug}, without disturbing any of the existing model parameters or the architecture. Unlike in [211], the GA-based optimization routine bears the load of IL and allows the feature extractor part of the method to be frozen. This facilitates the usage of feature vectors as exemplars, that remain the same across training iterations, waiving off the need to store large raw input data as exemplars, as done in [211]. This substantially lowers the memory footprint of the proposed method and also makes the memory footprint independent of the input sample size. During the CIL, the *novelty_fitness* keeps a check on the concept drift while the *exemplar_fitness* and the *augmented_exemplar_fitness* keep a check on the catastrophic forgetting. The candidates are used to mask the embeddings to enable the GA optimisation routine on top of the N_{lite} model, without disturbing its parameters/architecture during the GA optimisation process.

7.4 Experiments and Results

This section details the experiments conducted to analyse the validity and efficacy of the proposed method. The experiments involved a comparison of the proposed method with other methods based on the existing strategies widely used for CIL. This was followed by analysing the practical usability of the proposed method by deploying it on a resource-constrained ED, ESP32 [228]. The WISDM [45] dataset, a well-accepted dataset for IMU-based research, was used as the basis for

Algorithm 7 Onboard CIL using Genetic Algorithm and TinyML

Input: D_{new} (New class Dataset), Exm (Exemplars of all classes), Exm_{aug} (Augmented exemplars of all classes), D_{test} (Test data), P (Population size of candidates), Gen (Number of Generations), M (Number of Mutations)
Output: Φ_{best} (Best candidate), N_{lite} (TinyML model) &

$\Phi \leftarrow random.uniform(P, sizeOf(L_e), sizeOf(Exm))$
`/* create a population of candidates with randomly initialised values spread uniformly across 0 and 1. */`
while $Gen > 0$ **do**
  $F_\Phi \leftarrow []$
  for Φ_i *in* Φ **do**
    correct $\leftarrow 0$
    total $\leftarrow 0$
    for X, Y *in* D_{new} **do**
      pred $\leftarrow f(X, Y, \Phi_i, Exm)$
      `/* f returns the index of the class whose masked exemplar embedding has the least Euclidean distance from the masked embedding of X */`
      if $Pred = Y$ **then**
        correct $\leftarrow correct + 1$
      $total \leftarrow total + 1$
    accuracy $\leftarrow (correct/total) * 100$
    $novelty\_fitness \leftarrow accuracy$
    for X, Y *in* Exm **do**
      pred $\leftarrow f(X, Y, \Phi_i, Exm)$
      if $Pred = Y$ **then**
        correct $\leftarrow correct + 1$
      $total \leftarrow total + 1$
    accuracy $\leftarrow (correct/total) * 100$
    $exemplar\_fitness \leftarrow accuracy$
    for X, Y *in* Exm_{aug} **do**
      pred $\leftarrow f(X, Y, \Phi_i, Exm)$
      if $Pred = Y$ **then**
        correct $\leftarrow correct + 1$
      $total \leftarrow total + 1$
    accuracy $\leftarrow (correct/total) * 100$
    $augmented\_exemplar\_fitness \leftarrow accuracy$
    for X *in* Exm **do**
      $\text{Disc}_X \leftarrow g(X, \Phi_i, Exm)$
      `/* g returns the sum of the Euclidean distance of the masked embedding of X from the masked embedding of every exemplar */`
    $discrimination_fitness \leftarrow$discrimination_fitness + Disc_X
  $F_{\Phi i} = h1 * novelty\_fitness + h2 * exemplar\_fitness + h3 * augmented\_exemplar\_fitness + h4 * discrimination\_fitness$
  F_Φ.append($F_{\Phi i}$)
  $F \leftarrow sortByFitness(\Phi, F_\Phi)$ `/* rearrange candidates in increasing order of fitness */`
  $\Phi_{best} = \Phi[—\Phi—1)]$
  $\Phi_{parent} \leftarrow latterHalf(\Phi)$
  $\Phi_{children} \leftarrow []$
  for Par_1 *in* Φ_{parent} **do**
    Par_2 = pickParent(Φ_{parent}) $Child$ = concat(formerHalf(Par_1),latterHalf(Par_2))
    $\Phi_{children}$.append($Child$)
  $\Phi_{interim} \leftarrow \Phi_{parent} + \Phi_{children}$
  $mutation\_indices \leftarrow random(|\Phi|))$
  for j *in* $mutation\_indices$ **do**
    $\Phi[j]$ = uniformRandomNoise($\Phi[j]$)
    $M \leftarrow M - 1$
  Gen $\leftarrow Gen - 1$
return Φ_{best}, N_{lite}
`/* best candidate and the` N_{lite} `to be used for inference */`

D_s. It comprises raw accelerometer (a_x, a_y, a_z) and gyroscope (g_x, g_y, g_z) sensor data, collected from a smartphone and a smartwatch at a rate of 20Hz. The data was collected from 51 subjects across 18 different activities. In order to cater to resource-constrained devices, for our experiments, the WISDM dataset was wrangled by restructuring the data to contain samples of four classes (activities): *Walking*, *Jogging*, *Standing* and *Dribbling*. The gyroscope values were removed and a window size of forty was kept with a stride of ten, to prepare each sample sequence. A 1D CNN, as shown in Figure. 7.4, was used as the N_s. The onboard training dataset, D_{new}, was generated locally by using an MPU6050 sensor [46] connected to an ESP32-Wroom-32 (ESP32) microcontroller [228], as depicted in the Figure 7.2. The MPU6050 is an IMU that has a 3-axis accelerometer, a 3-axis gyroscope, a temperature sensor and a digital motion processor, packaged as a small chip. It provides robust sensing capabilities, has wide software support and is very affordable, priced at around three USD. The ESP32 microcontroller is a low-power and low-resource system on chip microcontroller with Wi-Fi and Bluetooth modules. It has a series of onboard pins that can be used to interface multiple sensors, including the MPU6050 sensor. However, it has low onboard Flash memory (4 MB) which makes the onboard usage of ML algorithms a challenging task. The D_{new} collection process involved connecting the MPU6050 sensor to the ESP32 microcontroller and powering it with a power bank (20000 mAh, 5V, 2A). A blinking LED onboard the microcontroller was used to indicate the beginning of the data collection process. The MPU6050 sensor was worn, using a band, on the wrist with its X-axis perpendicular to the ground and the Y-axis parallel to the ground. This involved keeping the hand parallel to the ground. This was followed by making a twisting motion of the fist in a clockwise and then an anti-clockwise fashion. The complete twisting action was completed within a second. 20 samples of the acceleration triplets (a_x, a_y, a_z), each along the X, the Y and the Z axis, were captured for each gesture (per second) yielding a sampling rate of 20Hz. A gesture (sample) involved making two such twisting actions. A total of 510 samples, along with the respective labels (*Twisting*), were collected out of which 408 samples were reserved for training and 102 samples were used for testing. Figure 7.3 shows the different types of samples used in the experiments.

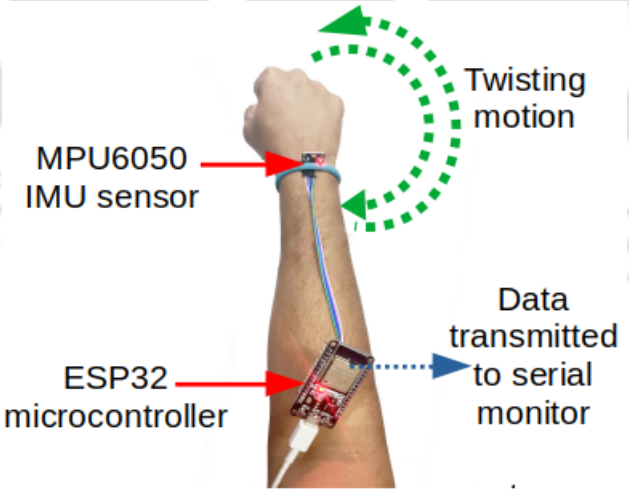


Figure 7.2: The setup used for collecting onboard data using an MPU6050 sensor and an ESP32 microcontroller

The Tensorflow [67] and the Keras [66] frameworks were used to design and implement the DNN models, wherever applicable. The Tensorflow Lite framework [162] was used to compress the model and handle ancillary TinyML operations. The Arduino IDE [137] was used to host and use

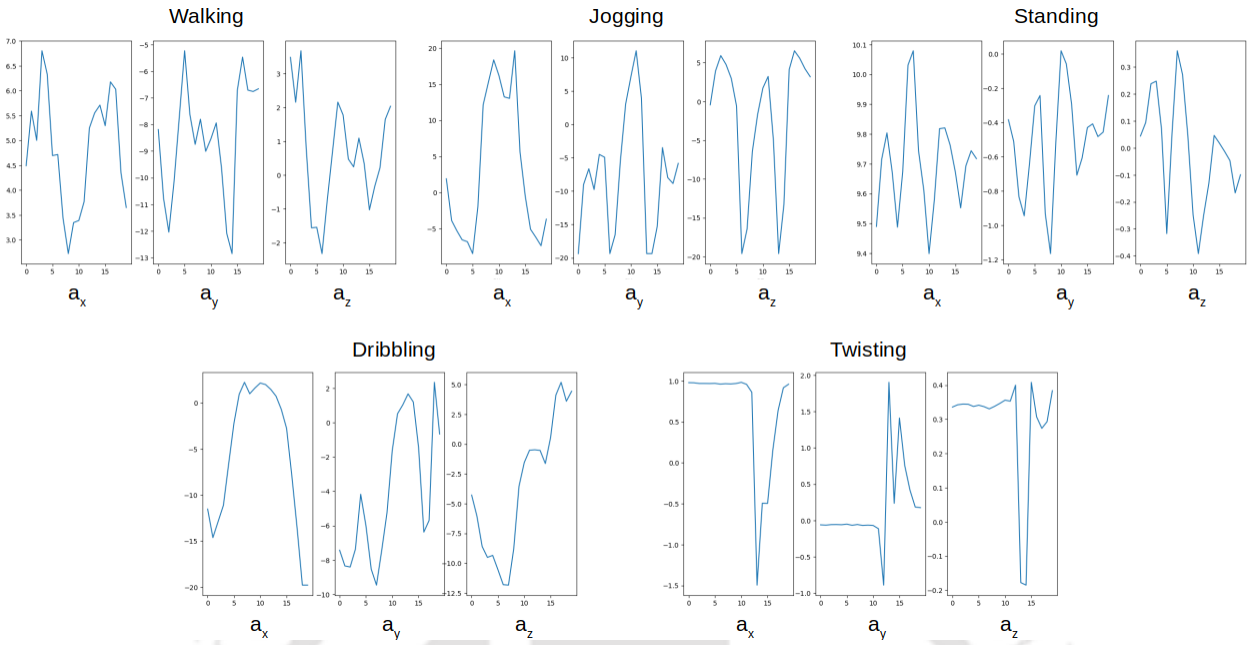


Figure 7.3: Time-series sequences of the data used in the experiments

the N_{lite} model and the proposed method on the ESP32. Figure 7.4 shows the architecture of the D_s used to extract the N_{lite} in all the experiments. As can be observed, the D_s is a 1D CNN with a penultimate *Dense layer* with five neurons used as the extraction layer, L_e. 1D CNNs are known to surpass many of the existing time-series models in terms of performance [229] and have a simple linear structure that allows easy model compression and deployment for the EDs.

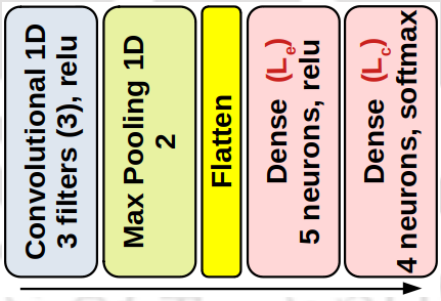


Figure 7.4: Architecture of the source network D_s used to create the N_{lite}

7.4.1 Comparison with other methods

Before deploying the proposed method onboard a microcontroller, it was compared with methods based on some of the existing CIL strategies, as discussed earlier, to assess the efficacy of the approach. Each of the methods is enumerated as follows:

1. N_{lite}: This method used N_{lite} to extract feature vectors from the presented samples which were then compared with the feature vectors of the exemplars, similar to a clustering approach.

The class of the closest feature vectors was recorded as the inference. This method did not involve any onboard IL.

2. N_{lite}_NN: This method used a neural network layer on top of the N_{lite}. N_{lite} was used to obtain the feature vectors of the training data. This was followed by creating a simple neural network with an input layer followed by a *Dense* layer having as many neurons as the number of exemplars. This neural network was trained, for 50 epochs, on the extracted feature vectors from the training data. The trained model was used for making the inference.

3. N_{lite}_GA: This is the proposed method that used both the N_{lite} and the GA for CIL. It involved a candidate population size of 10 trained across 15 generations with 4 mutations performed in each generation. Each candidate was represented by a matrix of size 5x5. The values of $h1, h2, h3$ and $h4$ were kept as 1, 1, 2.85 and 0.45, respectively for using the corresponding fitness functions as explained in the section 7.3.4. These were found after hyperparameter tuning.

4. $N_{lite}\_GA\_voting$: This method was an off-shoot variant of N_{lite}_GA. However, instead of using a single candidate (Φ_{best}), this method used the top 5 candidates for inference. The inferencing part involved recording the class to be the one that was voted the most among all the chosen candidates.

5. N_{lite}_SVM: This method used an SVM on top of the N_{lite}. The method commenced by extracting features of the training data using N_{lite}. The feature vectors were flattened and then fed to an SVM classifier having a linear kernel with a C value of one. The trained resulting model was used for inference.

6. N_{lite}_kNN: This method used a kNN classifier on top of the N_{lite}. N_{lite} was used for extracting the features from the training data which were flattened and fed to a kNN classifier having a neighbour size of three. The trained resulting model was used to make the inference.

For each method, the training datasets comprised the D_{new}, the Exm and the Exm_{aug}. The D_{new}, contained 408 samples, the Exm contained five samples and the Exm_{aug} contained 50 samples, each having the corresponding labels. The test data used for each method comprised 102 samples of type *Twisting* from the D_{new} with the corresponding labels. It also included 16,571 samples from the WISDM based D_s dataset with the corresponding labels. 10 experiments were conducted for each method and the accuracies obtained over the test data were averaged to a single value for each method. The inference involved the classification of the test data among one of the five classes (four old and one new class). Further, to analyse the impact of the number of samples of the new class, for each experiment, the training dataset size of the D_{new} was varied as having 10, 100, 250 and 408 samples. This amounted to a total of 240 experiments. Figure 7.5 shows the classification performances for all the methods.

The overall average accuracies for all the methods were found to be below 75 per cent. This can be attributed to the complex feature space of the test data, as shown in Figure 7.6. There exist substantial overlaps between the features of different classes of data. However, the conversion of the feature space to a better-segregated exemplar space, as shown in Figure 7.7, restricts the accuracies from falling too low. The N_{lite}_NN method gives the least accuracy due to the poor adaptation capability of the dense layer to the new data. As can be seen in the case with 10 samples, the N_{lite}_GA method is able to deliver better performance than the N_{lite} for scenarios having fewer samples of new class. The $N_{lite}\_GA\_voting$ method is unable to perform better than the N_{lite}_GA method which may be attributed to the lack of a positive consensus amongst the

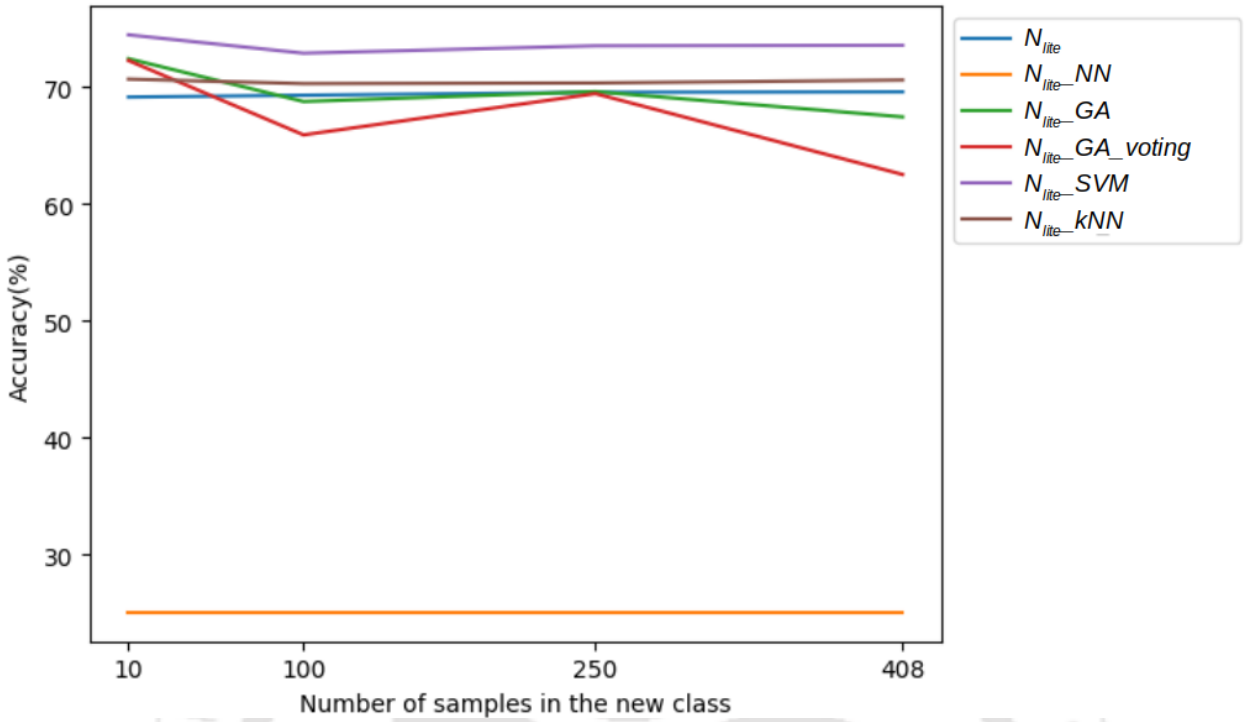


Figure 7.5: Comparing the proposed method with existing CIL methods

multiple Φ_is chosen for inference. The $discrimination\_fitness$ hyperparameter may be the reason behind this lack of consensus which, on the contrary, helps push the Φ_{best} to better optimisation landscapes in the N_{lite}_GA method. The N_{lite}_kNN method delivers a similar performance to that of the N_{lite}_GA method. However, owing to its *lazy* execution approach, it consumes more time in inferencing. In general, the N_{lite}_SVM method seems to deliver better performance than the other methods. However, this method does not allow the freedom to tweak different optimisation routes as can be done in the case of the N_{lite}_GA method. The candidate population size and the number of generations in the N_{lite}_GA method can be increased further, to strive for even better performances, based on the computing resources available at the target system.

7.4.2 Deployment on a Resource-Constrained Edge Device for Gesture Recognition

To assess the real-world efficacy of the N_{lite}_GA, it was deployed on an ESP32 device for a gesture recognition task, with the same setup as shown in Figure 7.2. The N_{lite} model was deployed on the ESP32 and the rest of the proposed method, along with the Exm, was coded onto the ESP32 using the Arduino IDE. The memory footprints of the code and the N_{lite} model were 58.2kB and 28.7kB, respectively. Keeping all the GA hyperparameters same as the ones mentioned earlier, experiments were conducted onboard the ESP32. A training dataset of 10 D_{new} samples was provided to the ESP32. To facilitate bulk testing, while adhering to the memory limitations of the microcontroller, separate test batches of the D_{test} were provided to the ESP32, wherein each batch had 10 samples from either the D_{new} or the D_s. This boiled down to providing the data of all the classes across the test batches. For each batch, 10 iterations of onboard training, followed by inference were

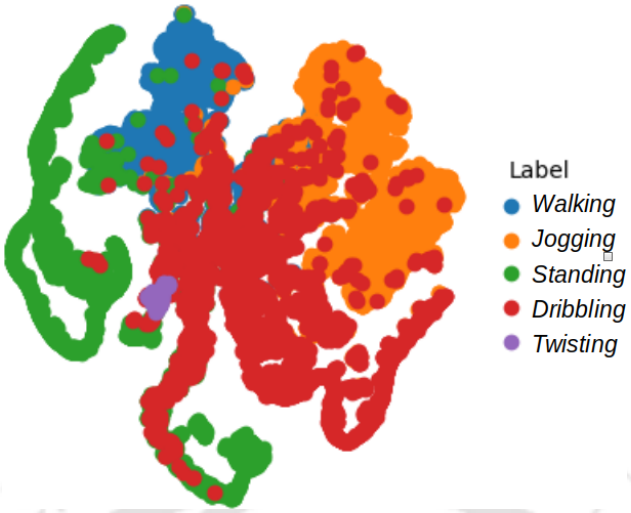


Figure 7.6: A t-SNE map of the feature vectors extracted from the test dataset

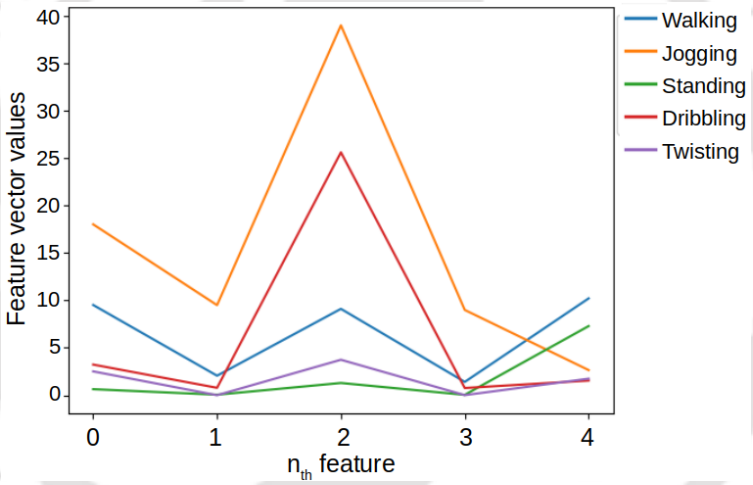


Figure 7.7: Feature vectors of the exemplars

executed. Eight such batches were tested across the experimental analyses. This amounted to testing 800 inferences. The proposed method achieved an onboard accuracy of 48 per cent for the 800 inferences. It may be noted that the onboard TinyML accuracies are known to be affected by the compressions performed, causing further propagation of error. Figure 7.8 shares a pertinent insight into the performance of the method. The confusion matrix shows a strong performance consistency for the *Twisting* class with 90 per cent accuracy for the corresponding predictions. This highlights the fact that the method can perform well on the data of the new class, encouraging further investigation into the hyperparameter space of the method. Further investigations into the hyperparameter space of the method could definitely result in an increase in the overall accuracy. The deployment of the proposed method as a CPS, leveraging physical interactions to update the computational model, all within an extremely resource-constrained microcontroller, establishes a promising research direction to explore further.

The deployment of the proposed GA-based CIL method on EDs, such as the ESP32, could thus be looked upon as a significant step towards augmenting the TinyML domain with GAs.

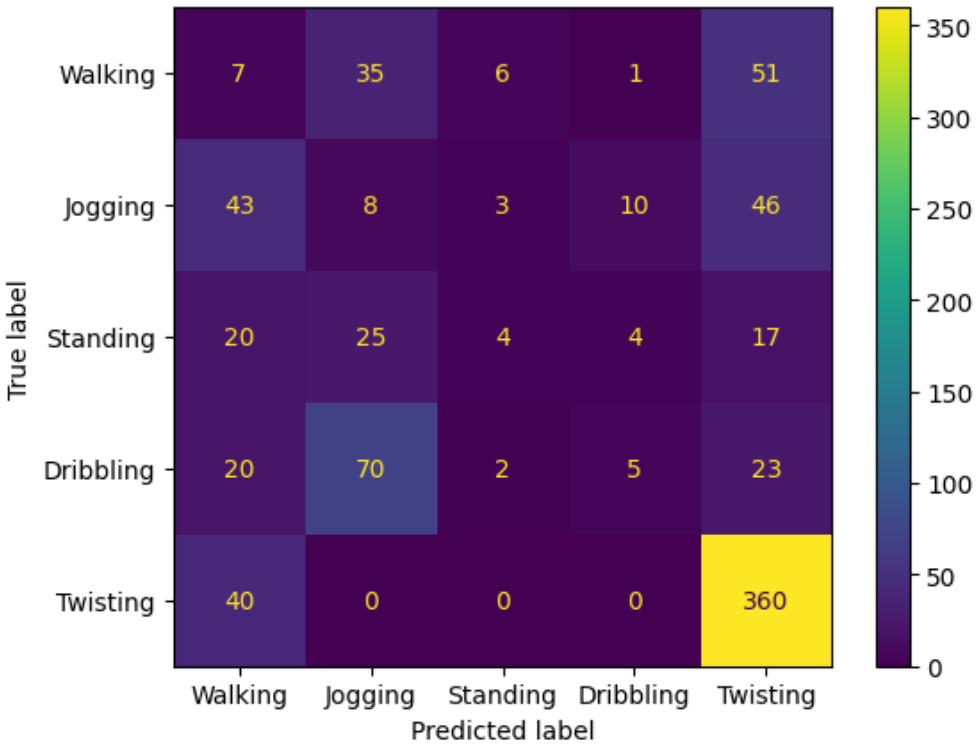


Figure 7.8: The confusion matrix depicting the onboard class-wise performance

7.5 Chapter Summary

The work covered in this chapter attempts to bridge the fields of Genetic Algorithms (GA) and TinyML so as to achieve Class Incremental Learning (CIL). Deploying ML models in real-world settings, especially over resource-constrained EDs, has always been a challenging task. While TinyML tackles this issue to an extent, by mostly using pre-trained DL models, the static nature of such models renders them ineffective for non-stationary data. A model having a low memory and computational footprint that could allow onboard CIL so as to accommodate data from new classes and also avoid catastrophic forgetting, is thus, the need of the day. GAs have the ability to span vast search spaces quickly and can be used to control multiple optimisation objectives together. Such an ability is crucial in scenarios involving low computation resources and real-time complex data. TinyML often faces such scenarios. The work conducted under this contribution endeavours to meet this need by providing a method that utilises TinyML to accommodate a DL model onboard a resource-constrained device. To enable onboard CIL over the DL model, the method leverages Latent Replays (as exemplars) and a GA to create a multi-fitness landscape that facilitates the inclusion of new class data, suppresses catastrophic forgetting and keeps a check on the quality of exemplars. The proposed method achieves a fair performance in comparison to the existing CIL approaches and maintains a low memory footprint, allowing it to be accommodated in resource-constrained EDs. A time-series domain application for gesture recognition was chosen to provide a challenging evaluation landscape for the proposed method. Results from real-world deployment of the proposed method establish its practical usability. This contribution provides a closure to the research endeavour made under this thesis. It connects back to the challenge of automating gesture recognition using robust ML techniques that was left open in the first contribution described in

chapter two. From an algorithmic point of view, it explores and provides a solution to the issue of CIL, which is a more prevalent and practical issue. It also successfully attempts to combine the fields of TinyML and GA to inspire further exploration in this direction. This contribution also extends the scope of the thesis by involving time series domain data, validating the usability of bio-inspired techniques there as well. Lastly, this contribution involves the application of the proposed method to an extremely resource-constrained, noisy and dynamic real-world deployment. Real-world deployments have been a cornerstone of this thesis and this last contribution abides by that practice to an altogether newer level by squeezing everything into a memory size of around four MB within a device of the size of a thumb.

8

Conclusions and Future Work

8.1 Conclusions

This thesis focuses on devising bio-inspired algorithms for resource-constrained CPSs. The contributions made under the thesis explore a wide range of challenges faced by the existing AI algorithms and also provide practical implementations of the corresponding solutions, in the form of CPSs in most cases. This validates the robustness and the pragmatic value of the research conducted under the thesis.

The first contribution realises a CPS and explores its utility for a proactive surveillance and response task. It proposes a *Smart Patch*, an IoT-enabled wearable that can help in tackling the issue of child trafficking. Using its sensory circuitry and a network of WiFi hotspots, *Smart Patch* is able to facilitate voluntary and involuntary responses to relay any message of concern or distress of the child to the guardian. The *Smart Patch* has a low form factor and cost, making it widely usable. It also involves very little manual intervention by the child/guardian in the surveillance and response mechanism. Having explored a resource-constrained CPS, the subsequent contribution is dedicated to the pursuit of a scalable data and compute-efficient ML method to cater to noisy and dynamic data distributions, commonly witnessed in CPSs. The second contribution (described in chapter three) explores a synergistic bridge between a connectionist and an evolutionary ML method. It successfully leverages an AIS to enhance the multi-class classification performance of an SNN. The experimental results testify to the increment in the accuracy and the speedup achieved by the proposed method. This contribution lays the groundwork for pursuing the combination of connectionist and evolutionary ML methods for resource-constrained devices. The third contribution indulges in the challenging task of enabling AI on an ED as part of a CPS. It proposes *ChaoticImmuneNet*, an EAI method for resource-constrained devices. *ChaoticImmuneNet* fairs well against the existing multi-class classification methods. Due to the usage of embeddings, it is able to perform onboard ML using EAI. The experimental results obtained by deploying *ChaoticImmuneNet* on a mobile robot, under a prototype warehouse, validate its competence in supporting real-time ML operations in the real world. While *ChaoticImmuneNet* minimises the computational load from EDs, it still involves a computationally heavy gradient-based process onboard the EDs. This limitation is removed in the fourth contribution that proposes *GADANN*. *GADANN* follows an altogether different approach by relying on a simulated dataset and a DA technique to deal with the inconsistency of the real-world data. The experimental results, conducted over a CPS, show the ability of *GADANN* to consistently adapt from the simulated to the real world in scenarios involving low data and computational resources, a feat tough to achieve by the existing data and

resource-hungry methods. *GADANN* is also able to support onboard ML on EDs without relying on any computationally heavy gradient-based approach. Instead, *GADANN* combines TinyML with GAs to support onboard ML. The successful usage of *GADANN* on a mobile robot for the task of pallet identification in a prototype warehouse testifies to its pragmatic value. The fifth contribution, described in chapter six, caters to the robot vision domain wherein, **RoboDA**, a dataset for evaluating DA methods for robot vision has been analysed. **RoboDA** comprises both simulated and real-world data collected under standard open-source setups. This makes **RoboDA** an easy-to-use dataset with better reproducibility and scalability. The experimental results discussed in this contribution highlight the role of the **RoboDA** dataset in filtering out good DA methods. The role of the simulated portion of **RoboDA** has also been highlighted by comparing it with a standard computer vision dataset that performs poorly for the DA task when used in the real world. The sixth and last contribution (chapter seven) wraps around the thesis by leveraging the experience gained so far to enable onboard learning and using it to cater to the issue left open in the first contribution (chapter two), viz. enabling ML-based gesture recognition onboard a microcontroller. The contribution proposes a GA and TinyML based method to tackle the issue of CIL. The proposed method maintains a low footprint system suitable for supporting onboard learning on an extremely resource-constrained device. The real-world deployment of the proposed method as a CPS proves the ability of GAs to extend the scope of TinyML towards onboard learning. This contribution puts a closure to the research pursuit of this thesis by catering to the gesture recognition task, as performed in the first contribution.

8.2 Future Work

This thesis mostly involves real-world CPS deployments that pose multiple challenges such as noisy data distributions, resource constraints, complex feature landscapes, etc. While the contributions under the thesis have attempted to cater to each of these challenges, there still exists an ample amount of opportunities to contribute towards in the future.

The first contribution embodies a generic mechanism to proactively conduct the surveillance and response task. However, the expanse of the mechanism is limited due to the involvement of WiFi hot-spots and the corresponding module for connectivity. Thus, the integration of a GSM-based network into the *Smart Patch* would expand its coverage substantially.

In the second contribution, the proposed method of combining the antigenic and IN contributions uses a simple multiplication-based strategy. Considering the impact that such a combination process has on the classification performance of the overall method, further investigations into the process of combining the two contributions would be a promising step to push the performance further.

The third contribution proposes *ChaoticImmunenNet*, an EAI based method to enable onboard ML for EDs. *ChaoticImmunenNet* is able to deliver good performance in terms of accuracy and inference time when compared to existing onboard ML methods. However, *ChaoticImmunenNet* still relies on compute-intensive gradient-based operations onboard the ED. Considering the usage of multiple GSs in *ChaoticImmunenNet*, an exploration of parallel hardware is also a promising direction to follow in the future.

The fourth contribution introduces *GADANN* that facilitates the transfer of knowledge across domains. *GADANN* has been tested for scenarios involving low availability of data and compute power. Exploring the usage of *GADANN* in data-rich settings is still open for research. The usage of better generative techniques to adapt data across domains is another direction to explore in the case of *GADANN*.

The fifth contribution attempts to cater to the demand of a robot vision dataset for DA. The focus of this contribution was to allow quick evaluation of DA tasks with cost-effective open-source setups. Having established the demand and utility of a robot vision data, the next step is to incorporate a more complex dataset involving multiple objects with an additional set of tasks to perform such as segmentation, detection, tracking, etc.

The last contribution targets an ambitious task of allowing onboard CIL over an extremely resource-constrained device. While the proposed method is able to incorporate the knowledge of new data points, it does not perform well for old data points, post CIL. This encourages further research on the hyperparameters and fitness landscape definitions used in the experiments.

Appendix

Method	Description	Hyperlink
ChaoticImmuneNet	Schematic animation and recording of the associated experiment	Link
GADANN	*Recording of the virtual experiment*: The virtual screen at the top half shows the activity of the virtual robot across different rounds. The console at the bottom half shows the predicted values and ground truth associated with the pallets encountered by the virtual robot. It also shows the prediction accuracy at the end of each round, that follows an increasing trend.	Link
GADANN	*Recording of the world view of the real world experiment*: This recording shows the activity of the real robot across different rounds. The robot, within each round, stops upon encountering a pallet, takes its snapshots, makes and stores the predictions and moves to the next pallet. Unlike the virtual robot, the real robot faces real world challenges, for instance, friction due to the floor and non-uniform motion due to draining of batteries. Such challenges add to the noise observed in the captured pallet snapshots, making the problem at hand more complex. This necessitate the role of onboard learning with real-world data in addition to the knowledge learnt from the simulated data.	Link
GADANN	*Recording of the robot's internal operations in the real world experiment*: This recording shows the predictions and the corresponding ground truth of the pallet encountered by the real robot in the real world. In addition, the prediction accuracy of each round is also shown at the end of the round, that follows an increasing trend.	Link

Bibliography

[1] Cyber-Physical Systems (CPS). *NSF - National Science Foundation*, March 2024. [Online; accessed 17. Mar. 2024].

[2] Felix O. Olowononi, Danda B Rawat, and Chunmei Liu. Resilient machine learning for networked cyber physical systems: A survey for machine learning security to securing machine learning for cps. *IEEE Communications Surveys Tutorials*, 23(1):524–552, 2021.

[3] Jiafu Wan, Hehua Yan, Hui Suo, and Fang Li. Advances in Cyber-Physical Systems Research. *KSII Transactions on Internet and Information Systems (TIIS)*, 5(11):1891–1908, 2011.

[4] Mahbuba Afrin, Jiong Jin, Ashfaqur Rahman, Andrea Gasparri, Yu-Chu Tian, and Ambarish Kulkarni. Robotic edge resource allocation for agricultural cyber-physical system. *IEEE Transactions on Network Science and Engineering*, 9(6):3979–3990, 2022.

[5] Kun Cao, Shiyan Hu, Yang Shi, Armando Walter Colombo, Stamatis Karnouskos, and Xin Li. A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 17(11):7806–7819, 2021.

[6] Md. Maruf Hossain Shuvo, Syed Kamrul Islam, Jianlin Cheng, and Bashir I. Morshed. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, 111(1):42–91, 2023.

[7] Edge Device Overview and Technology Solutions – Intel, May 2024. [Online; accessed 19. May 2024].

[8] Luigi Capogrosso, Federico Cunico, Dong Seon Cheng, Franco Fummi, and Marco Cristani. A machine learning-oriented survey on tiny machine learning. *IEEE Access*, 12:23406–23426, 2024.

[9] Artificial Intelligence market size/revenue comparisons 2022 | Statista, April 2024. [Online; accessed 20. Apr. 2024].

[10] Topic: Big data, April 2024. [Online; accessed 20. Apr. 2024].

[11] Data center chip architecture for AI training 2025 | Statista, April 2024. [Online; accessed 20. Apr. 2024].

[12] Machine Learning (ML) on the Edge: 5 Companies Showing Us What It Looks Like, April 2024. [Online; accessed 20. Apr. 2024].

[13] Gaudenz Boesch. Edge Intelligence: Edge Computing and ML (2024 Guide) - viso.ai. *Viso*, April 2024.

[14] John Soldatos. 2023 Edge AI Technology Report. Chapter II: Advantages of Edge AI. *Wevolver*, July 2023.

[15] Syntiant. *Syntiant*, April 2024. [Online; accessed 21. Apr. 2024].

[16] Arpan Kumar Kar. Bio inspired computing – a review of algorithms and scope of applications. *Expert Systems with Applications*, 59:20–32, 2016.

[17] BING LI WEISUN JIANG. Optimizing complex functions by chaos search. *Cybernetics and Systems*, 29(4):409–419, 1998.

[18] Hong-Ji Meng, Peng Zheng, Rong-Yang Wu, Xiao-Jing Hao, and Zhi Xie. A hybrid particle swarm algorithm with embedded chaotic search. In *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, volume 1, pages 367–371 vol.1, 2004.

[19] T. Yamada, K. Aihara, and M. Kotani. Chaotic neural networks and the traveling salesman problem. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pages 1549–1552 vol.2, 1993.

[20] Haitao Yuan, Jing Bi, MengChu Zhou, and Ahmed Chiheb Ammari. Time-aware multi-application task scheduling with guaranteed delay constraints in green data center. *IEEE Transactions on Automation Science and Engineering*, 15(3):1138–1151, 2018.

[21] Shreshth Tuli, Giuliano Casale, and Nicholas Jennings. Simtune: bridging the simulator reality gap for resource management in edge-cloud computing. *Scientific Reports*, 12, 11 2022.

[22] Meelis Kull and Peter A. Flach. Patterns of dataset shift. 2014.

[23] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale. Are we done with object recognition? the icub robot's perspective. *Robotics and Autonomous Systems*, 112:260–281, 2019.

[24] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999.

[25] Linda B. Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial Life*, 11:13–29, 2005.

[26] Gabriele Angeletti, Barbara Caputo, and Tatiana Tommasi. Adaptive deep learning through visual domain localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7135–7142, 2018.

[27] Yang Liu, Xin Tao, Xin Li, Armando Walter Colombo, and Shiyan Hu. Artificial intelligence in smart logistics cyber-physical systems: State-of-the-arts and potential applications. *IEEE Transactions on Industrial Cyber-Physical Systems*, 1:1–20, 2023.

[28] Franklin Oliveira, Daniel G. Costa, Flávio Assis, and Ivanovitch Silva. Internet of intelligent things: A convergence of embedded systems, edge computing and machine learning. *Internet of Things*, 26:101153, 2024.

[29] Rakhee Kallimani, Krishna Pai, Prasoon Raghuwanshi, Sridhar Iyer, and Onel Alcaraz López. Tinyml: Tools, applications, challenges, and future research directions, 03 2023.

[30] TinyChat: Large Language Model on the Edge, April 2024. [Online; accessed 9. Apr. 2024].

[31] Gartner Identifies Top Trends Shaping the Future of Data Science and Machine Learning, January 2024. [Online; accessed 31. Jan. 2024].

[32] Ashraf Darwish. Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. *Future Computing and Informatics Journal*, 3(2):231–246, 2018.

[33] Juan Isern, Francisco Barranco, Daniel Deniz, Juho Lesonen, Jari Hannuksela, and Richard R. Carrillo. Reconfigurable cyber-physical system for critical infrastructure protection in smart cities via smart video-surveillance. *Pattern Recognition Letters*, 140:303–309, 2020.

[34] Shao-Bo Lin. Generalization and expressivity for deep nets. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1392–1406, 2019.

[35] Jane Bromley, James Bentz, Leon Bottou, Isabelle Guyon, Yann Lecun, Cliff Moore, Eduard Sackinger, and Rookpak Shah. Signature Verification using a "Siamese" Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:25, 08 1993.

[36] Gerardo Quintana López, Luis Antonio Morales, and Luis Fernando Niño. Immunological computation. In *Autoimmunity: From Bench to Bedside [Internet]*. El Rosario University Press, July 2013.

[37] Akash Moodbidri and Hamid Shahnasser. Child safety wearable device. In *2017 International Conference on Information Networking (ICOIN)*, pages 438–444, 2017.

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.

[39] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep Learning for Classical Japanese Literature, 2018.

[40] N. K. Jerne. Towards a Network Theory of the Immune System. *Ann. Immunol. (Paris).*, 125C(1-2):373–389, January 1974.

[41] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, 2010.

[42] Qinghai Miao, Yisheng Lv, Min Huang, Xiao Wang, and Fei-Yue Wang. Parallel learning: Overview and perspective for computational learning across syn2real and sim2real. *IEEE/CAA Journal of Automatica Sinica*, 10(3):603–631, 2023.

[43] Jatinder N.D Gupta and Randall S Sexton. Comparing backpropagation with a genetic algorithm for neural network training. *Omega*, 27(6):679–684, 1999.

[44] Shifei Ding, Li Xu, Chunyang Su, and Hong Zhu. Using genetic algorithms to optimize artificial neural networks. *JCIT*, 5:54–62, 10 2010.

[45] Gary M. Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202, 2019.

[46] MPU-6050 | TDK InvenSense, November 2023. [Online; accessed 31. Jan. 2024].

[47] Suraj Kumar Pandey, Sonia Sonia, Tushar Semwal, and Shivashankar B. Nair. Smart patch: An iot based anti child-trafficking solutio. In *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, pages 194–200, 2021.

[48] Suraj Kumar Pandey and Shivashankar B. Nair. Immuno-inspired augmentation of siamese neural network for multi-class classification. In Wei Qi Yan, Minh Nguyen, and Martin Stommel, editors, *Image and Vision Computing*, pages 486–500, Cham, 2023. Springer Nature Switzerland.

[49] Suraj Kumar Pandey and Shivashankar B Nair. Enhancing siamese neural networks for multi-class classification: An immuno-inspired approach. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 39–40, New York, NY, USA, 2023. Association for Computing Machinery.

[50] Suraj Kumar Pandey and Shivashankar B Nair. Onboard class incremental learning for resource-constrained scenarios using genetic algorithm and tinyml. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '24 Companion, page 299–302, New York, NY, USA, 2024. Association for Computing Machinery.

[51] Alessandro Quarto, Domenico Soldo, Simona Gemmano, Rita Dario, Vincenzo Di Lecce, Cataldo Guaragnella, Angelo Cardellicchio, and Angela Lombardi. Iot and cps applications based on wearable devices. a case study: Monitoring of elderly and infirm patients. In *2017 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, pages 1–6, 2017.

[52] S. Rama Gokula Krishnan. Reducing Child Trafficking in India: The Role of Human Rights Education and Social Work Practice. *J. Hum. Rights Soc. Work*, 8(2):156–166, June 2023.

[53] Forced labour, modern slavery and human trafficking. Available: https://www.ilo.org/global/topics/forced-labour/lang--en/index.htm.

[54] Child trafficking in india. Available: https://www.theguardian.com/global-development/2015/apr/28/child-trafficking-india-domestic-labour-chhattisgarh.

[55] Human trafficking. Available: https://migrationdataportal.org/themes/human-trafficking.

[56] HUMAN TRAFFICKING - QUESTIONS & ANSWERS. Available: https://d306pr3pise04h.cloudfront.net/docs/issues_doc%2Flabour%2FForced_labour%2FHUMAN_TRAFFICKING_-_BACKGROUND_BRIEFING_NOTE_-_final.pdf.

[57] Global slavery index. Available: https://www.globalslaveryindex.org/2018/findings/country-studies/india/.

[58] Soo-Cheol Kim, Young-Sik Jeong, and Sang-Oh Park. Rfid-based indoor location tracking to ensure the safety of the elderly in smart home environments. *Personal Ubiquitous Comput.*, 17(8):1699–1707, December 2013.

[59] Ke-Yu Chen, Mark Harniss, Justin Haowei Lim, Youngjun Han, Kurt L. Johnson, and Shwetak N. Patel. ulocate: A ubiquitous location tracking system for people aging with disabilities. In *Proceedings of the 8th International Conference on Body Area Networks*, BodyNets '13, pages 173–176, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[60] Cong-Thinh Huynh, Huu-Quoc Nguyen, Xuan-Qui Pham, Tien-Dung Nguyen, and Eui-Nam Huh. Cloud-based real-time location tracking and messaging system: A child-care case study. In *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*, IMCOM '15, pages 23:1–23:7, New York, NY, USA, 2015. ACM.

[61] The New Apple Watch Doubles As One Dope Kids' GPS Tracker, March 2022. [Online; accessed 23. Nov. 2023].

[62] 4 geolocation technologies compared - Sensolus: Asset tracking solution to streamline logistics processes, February 2023. [Online; accessed 23. Nov. 2023].

[63] Comparing Location Technologies (GPS, vs RFID vs Bluetooth vs. Wi-Fi) - GTI Tech, February 2023. [Online; accessed 23. Nov. 2023].

[64] Jayun Patel and Ragib Hasan. Smart bracelets: Towards automating personal safety using wearable smart jewelry. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–2, 2018.

[65] Sarifah Putri Raflesia, Firdaus, and Dinda Lestarini. An integrated child safety using geofencing information on mobile devices. In *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pages 379–384, 2018.

[66] Gabriel Deak, Kevin Curran, and Joan Condell. Review: A survey of active and passive indoor localisation systems. *Comput. Commun.*, 35(16):1939–1954, September 2012.

[67] M Regus, R Talar, and R Labudzki. Indoor positioning and navigation system for autonomous vehicles based on RFID technology. *IOP Conference Series: Materials Science and Engineering*, 659:012059, oct 2019.

[68] Xinrong Li and Nayef Alsindi. Recent advances in indoor geolocation techniques. *International Journal of Wireless Information Networks*, 20(4):243–245, Dec 2013.

[69] Y. Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys Tutorials*, 11(1):13–32, 2009.

[70] Tushar Semwal and Shivashankar Nair. Agpi: Agents on raspberry pi. *Electronics*, 5:72, 10 2016.

[71] V. Varshney, R. K. Goel, and M. A. Qadeer. Indoor positioning system using wi-fi bluetooth low energy technology. In *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, pages 1–6, 2016.

[72] Andrei Popleteev, Venet Osmani, and Oscar Mayora. Investigation of indoor localization with ambient fm radio stations. 08 2013.

[73] X. Xie, H. Xu, G. Yang, Z. Mao, W. Jia, and M. Sun. Reuse of wifi information for indoor monitoring of the elderly. In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, pages 261–264, 2016.

[74] Ahmed Azeez, Saba Jabbar, Dr.Mohammed Sulttan, Desheng Wang, and Medo Toty. Wireless indoor localization systems and techniques: Survey and comparative study. *Indonesian Journal of Electrical Engineering and Computer Science*, 3:392–409, 08 2016.

[75] themonkeyapp.com. Available: https://themonkeyapp.com/.

[76] MQTT Dash. Available: https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=en_IN.

[77] Blynk. Available: https://play.google.com/store/apps/details?id=cc.blynk&hl=en_IN.

[78] CloudMQTT. Available: https://www.cloudmqtt.com/.

[79] Charlie Veal, Jeffrey Schulz, Andrew Buck, Derek Anderson, James Keller, Mihail Popescu, Grant Scott, Dominic Ho, and Timothy Wilkin. Doing more with less: Similarity Neural Nets and metrics for small Class Imbalanced data sets. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV*, volume 11418 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 1141802, April 2020.

[80] Thomas Dean and Mark Boddy. An Analysis of Time-Dependent Planning. In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, AAAI'88, page 49–54. AAAI Press, 1988.

[81] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[82] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2016.

[83] Sabrina M. Neuman, Brian Plancher, Bardienus P. Duisterhof, Srivatsan Krishnan, Colby Banbury, Mark Mazumder, Shvetank Prakash, Jason Jabbour, Aleksandra Faust, Guido C.H.E. de Croon, and Vijay Janapa Reddi. Tiny robot learning: Challenges and directions for machine learning in resource-constrained robots. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 296–299, 2022.

[84] Surbhi Gupta, Sangeeta R, Ravi Shankar Mishra, Gaurav Singal, Tapas Badal, and Deepak Garg. Corridor segmentation for automatic robot navigation in indoor environment using edge devices. *Computer Networks*, 178:107374, 2020.

[85] Harry A. Pierson and Michael S. Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821–835, 2017.

[86] L. Barreto, A. Amaral, and T. Pereira. Industry 4.0 implications in logistics: an overview. *Procedia Manufacturing*, 13:1245–1252, 2017. Manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain.

[87] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[88] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning deep neural networks in continuous learning scenarios. In *ACCV Workshops*, 2016.

[89] Nil Llisterri Giménez, Marc Grau, Roger Pueyo Centelles, and F. Freitag. On-device training of machine learning models on microcontrollers with federated learning. *Electronics*, 11:573, 02 2022.

[90] Jane Bromley, James Bentz, Leon Bottou, Isabelle Guyon, Yann Lecun, Cliff Moore, Eduard Sackinger, and Rookpak Shah. Signature Verification using a "Siamese" Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:25, 08 1993.

[91] Gerardo Quintana López, Luis Antonio Morales, and Luis Fernando Niño. Immunological computation. In *Autoimmunity: From Bench to Bedside [Internet]*. El Rosario University Press, July 2013.

[92] Hairong Lin, Quanli deng, Cong Xu, Zekun Deng, and Chao Zhou. Review on chaotic dynamics of memristive neuron and neural network. *Nonlinear Dynamics*, 106, 09 2021.

[93] Josie Hughes, Arsen Abdulali, Ryman Hashem, and Fumiya Iida. Embodied artificial intelligence: Enabling the next intelligence revolution. *IOP Conference Series: Materials Science and Engineering*, 1261:012001, 10 2022.

[94] Nicholas Roy, Ingmar Posner, Tim Barfoot, Philippe Beaudoin, Yoshua Bengio, Jeannette Bohg, Oliver Brock, Isabelle Depatie, Dieter Fox, Dan Koditschek, Tomas Lozano-Perez, Vikash Mansinghka, Christopher Pal, Blake Richards, Dorsa Sadigh, Stefan Schaal, Gaurav Sukhatme, Denis Therien, Marc Toussaint, and Michiel Panne. From machine learning to robotics: Challenges and opportunities for embodied intelligence, 10 2021.

[95] Chiara Bartolozzi, Giacomo Indiveri, and Elisa Donati. Embodied neuromorphic intelligence. *Nature Communications*, 13:1024, 02 2022.

[96] Tianyao Zhang, Xiaoguang Hu, Jin Xiao, and Guofeng Zhang. A survey of visual navigation: From geometry to embodied ai. *Engineering Applications of Artificial Intelligence*, 114:105036, 2022.

[97] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[98] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

[99] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, C. Karen Liu, Jan Peters, Shuran Song, Peter Welinder, and Martha White. Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400, 2021.

[100] B. Chen, A. Bakhshi, G. Batista, B. Ng, and T. Chin. Update compression for deep neural networks on the edge. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3075–3085, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[101] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, PP:1–20, 07 2019.

[102] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, 2018.

[103] Sarah Aguasvivas Manzano, Dana Hughes, Cooper Simpson, Radhen Patel, and Nikolaus Correll. Embedded neural networks for robot autonomy, 11 2019.

[104] Larissa Ferreira Rodrigues Moreira, Rodrigo Moreira, Bruno Augusto Nassif Travençolo, and André Ricardo Backes. An artificial intelligence-as-a-service architecture for deep learning model embodiment on low-cost devices: A case study of covid-19 diagnosis. *Applied Soft Computing*, 134:110014, 2023.

[105] Giulia Pasquale, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Object identification from few examples by improving the invariance of a deep convolutional neural network. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4904–4911, 2016.

[106] Xiaotian Fan, Yubo Yan, Panlong Yang, and Feiyu Han. Cmss: Use low-power iot cameras to monitor store shelves. In *2021 7th International Conference on Big Data Computing and Communications (BigCom)*, pages 309–315, 2021.

[107] M. Bechtel, Q. Weng, and H. Yun. Deeppicarmicro: Applying tinyml to autonomous cyber physical systems. In *2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 120–127, Los Alamitos, CA, USA, aug 2022. IEEE Computer Society.

[108] Michael Garrett Bechtel, Elise McEllhiney, and Heechul Yun. Deeppicar: A low-cost deep neural network-based autonomous car. *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 11–21, 2017.

[109] Brian H. Curtin and Suzanne J. Matthews. Deep learning for inexpensive image classification of wildlife on the raspberry pi. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0082–0087, 2019.

[110] Rung-Ching Chen, Vani Saravanarajan, and Hsiu-Te Hung. Monitoring the behaviours of pet cat based on yolo model and raspberry pi. *International Journal of Applied Science and Engineering*, 18:1–12, 01 2021.

[111] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 06 2016.

[112] El mehdi Raouhi, Mohamed Lachgar, Hamid Hrimech, and Ali Kartit. Transfer learning for plants' disease classification with siamese networks in low data regime. *International Journal of Computer Engineering and Data Science (IJCEDS)*, 1(1):8–13, Jul. 2021.

[113] Mohsen Heidari and Kazim Fouladi-Ghaleh. Using siamese networks with transfer learning for face recognition on small-samples datasets. In *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pages 1–4, 2020.

[114] Bangcai Wu, Xiaohui Li, Feng Yuan, He Li, and Mingming Zhang. Transfer learning and siamese neural network based identification of geochemical anomalies for mineral exploration: A case study from the cuau deposit in the nw junggar area of northern xinjiang province, china. *Journal of Geochemical Exploration*, 232:106904, 2022.

[115] Mohammad Naeem Naseri and Arun Prakash Agrawal. Impact of transfer learning on siamese networks for face recognition with few images per class. In *2021 Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–5, 2021.

[116] Nicole A. Doria-Rose and M. Gordon Joyce. Strategies to guide the antibody affinity maturation process. *Current opinion in virology*, 11:137, April 2015.

[117] Yongbin Zhu, Tao Li, and Xiaolong Lan. Feature selection optimized by the artificial immune algorithm based on genome shuffling and conditional lethal mutation. *Applied Intelligence*, 53(11):13972–13992, oct 2022.

[118] Eric W. Weisstein. Logistic Map. *Wolfram Research, Inc.*, November 2001.

[119] Bram Hunt, Eugene Kwan, Derek Dosdall, Rob S MacLeod, and Ravi Ranjan. Siamese neural networks for small dataset classification of electrograms. In *2021 Computing in Cardiology (CinC)*, volume 48, pages 1–4, 2021.

[120] Chiara Gabellieri, Alessandro Palleschi, Lucia Pallottino, and Manolo Garabini. Autonomous unwrapping of general pallets: A novel robot for logistics exploiting contact-based planning. *IEEE Transactions on Automation Science and Engineering*, 20(2):1194–1211, 2023.

[121] Arso Vukicevic, Marko Mladineo, Nikola Banduka, and Ivan Macuzic. A smart warehouse 4.0 approach for the pallet management using machine vision and internet of things (iot): A real industrial case study. *Advances in Production Engineering Management*, 16:297–306, 09 2021.

[122] mm. admin. Save time - Automatic pallet recognition for warehouse and intralogistics - vision by ifm. *vision by ifm*, June 2020.

[123] Ifadhila Affia and Ammar Aamer. An internet of things-based smart warehouse infrastructure: design and application. *Journal of Science and Technology Policy Management*, ahead-of-print, 04 2021.

[124] Réné Grzeszick, Sascha Feldhorst, Christian Mosblech, Gernot A Fink, and Michael ten Hompel. Camera-assisted pick-by-feel. *Logistics Journal: Proceedings*, 2016(10), 2016.

[125] Yongyao Li, Xiaohe Chen, Guanyu Ding, Chao Li, Sen Wang, Qinglei Zhao, and Qi Song. Pallet Localization Techniques of Forklift Robot: A Review of Recent Progress . *Journal of Robotics and Mechanical Engineering*, 1.

[126] Benjamin Schnieders, Shan Luo, Gregory Palmer, and Karl Tuyls. Fully convolutional one-shot object segmentation for industrial robotics, 2019.

[127] Huwei Liu, Li Zhou, Junhui Zhao, Fan Wang, Jianglong Yang, Kaibo Liang, and Zhaochan Li. Deep-learning-based accurate identification of warehouse goods for robot picking operations. *Sustainability*, 14:7781, 06 2022.

[128] Anubhav D. Patel and Abhra R. Chowdhury. Vision-based object classification using deep learning for inventory tracking in automated warehouse environment. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, pages 145–150, 2020.

[129] Tuomas Jalonen, Firas Laakom, Moncef Gabbouj, and Tuomas Puoskari. Visual product tracking system using siamese neural networks. *IEEE Access*, 9:76796–76805, 2021.

[130] Junhao Xiao, Huimin Lu, Lilian Zhang, and Jianhua Zhang. Pallet recognition and localization using an rgb-d camera. *International Journal of Advanced Robotic Systems*, 14(6):1729881417737799, 2017.

[131] René Keßler, Christian Melching, Ralph Goehrs, and Jorge Marx Gómez. Using camera-drones and artificial intelligence to automate warehouse inventory. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.

[132] Lele Xu, Teng Wang, Wenzhe Cai, and Changyin Sun. Uav target following in complex occluded environments with adaptive multi-modal fusion. *Applied Intelligence*, 53(13):16998–17014, Jul 2023.

[133] Fire Bird V ATMEGA2560 Robotic Research Platform - Nex Robotics, January 2023. [Online; accessed 9. Jan. 2023].

[134] 5MP Raspberry Pi 3/4 Model B Camera Module Rev 1.3 with Cable, January 2023. [Online; accessed 9. Jan. 2023].

[135] MQTT - The Standard for IoT Messaging, December 2022. [Online; accessed 9. Jan. 2023].

[136] NodeMcu – An open-source firmware based on ESP8266 wifi-soc., December 2021. [Online; accessed 9. Jan. 2023].

[137] Arduino - Home, January 2023. [Online; accessed 9. Jan. 2023].

[138] Edge Computing Market Size, Share & Trends Analysis Report By Component (Hardware, Software, Services, Edge-managed Platforms), By Application, By Industry Vertical, By Region, And Segment Forecasts, 2023 - 2030, June 2023. [Online; accessed 5. Jun. 2023].

[139] Edge computing and its role in the smart factory:, June 2023. [Online; accessed 5. Jun. 2023].

[140] Colby R. Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, David Patterson, Danilo Pau, Jae sun Seo, Jeff Sieracki, Urmish Thakker, Marian Verhelst, and Poonam Yadav. Benchmarking tinyml systems : Challenges and direction, March 2020. 6 pages, 1 figure, 3 tables. This is an author-produced version of the published paper. Uploaded in accordance with the publisher's self-archiving policy. Further copying may not be permitted; contact the publisher for details.

[141] TinyML: What is It and Why Does It Matter?, 1678. [Online; accessed 24. Mar. 2023].

[142] Dr. Lachit Dutta and Swapna Bharali. Tinyml meets iot: A comprehensive survey. *Internet of Things*, 16:100461, 2021.

[143] Haoyu Ren, Darko Anicic, and Thomas A. Runkler. Tinyol: Tinyml with online-learning on microcontrollers. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[144] Mahesh Ranaweera and Qusay H. Mahmoud. Virtual to real-world transfer learning: A systematic review. *Electronics*, 10(12), 2021.

[145] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[146] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. 07 2017.

[147] Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *ArXiv*, abs/1707.02267, 2017.

[148] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[149] Mesay Belete Bejiga and Farid Melgani. Gan-based domain adaptation for object classification. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1264–1267, 2018.

[150] P. O. Pinheiro. Unsupervised domain adaptation with similarity learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8004–8013, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[151] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5716–5726, 2017.

[152] Samson Ooko, Marvin Ogore, J. Nsenga, and Marco Zennaro. Tinyml in africa: Opportunities and challenges. pages 1–6, 12 2021.

[153] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems, 10 2020.

[154] Leonardo Ravaglia, Manuele Rusci, Davide Nadalini, Alessandro Capotondi, Francesco Conti, and Luca Benini. A tinyml platform for on-device continual learning with quantized latent replays. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):789–802, 2021.

[155] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[156] Vijay Chahar, Sourabh Katoch, and Sumit Chauhan. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 02 2021.

[157] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 34(2):550–570, 2023.

[158] R. Oullette, M. Browne, and K. Hirasawa. Genetic algorithm optimization of a convolutional neural network for autonomous crack detection. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 516–521 Vol.1, 2004.

[159] S. Chopra, R. Hadsell, and Y. LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005.

[160] Ltd. Buy a Raspberry Pi 3 Model B – Raspberry Pi, January 2023. [Online; accessed 9. Jan. 2023].

[161] Cyberbotics: Robotics simulation with Webots, June 2023. [Online; accessed 8. Jun. 2023].

[162] TensorFlow Lite | ML for Mobile and Edge Devices, May 2023. [Online; accessed 8. Jun. 2023].

[163] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[164] Run Data Science & Machine Learning Code Online | Kaggle, September 2024. [Online; accessed 29. Sep. 2024].

[165] Sergey I. Nikolenko. Synthetic data for deep learning. *CoRR*, abs/1909.11512, 2019.

[166] Luca Ciampi, Nicola Messina, Fabrizio Falchi, Claudio Gennaro, and Giuseppe Amato. Virtual to real adaptation of pedestrian detectors. *Sensors*, 20(18), 2020.

[167] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.

[168] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. 10 2017.

[169] Peter De Roovere, Steven Moonen, Nick Michiels, and Francis Wyffels. Dataset of industrial metal objects, 2022.

[170] Xiaomeng Zhu, Talha Bilal, Pär Mårtensson, Lars Hanson, Mårten Björkman, and Atsuto Maki. Towards sim-to-real industrial parts classification with synthetic dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4453–4462, June 2023.

[171] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[172] Xuan Li, Kunfeng Wang, Yonglin Tian, Lan Yan, Fang Deng, and Fei-Yue Wang. The paralleleye dataset: A large collection of virtual images for traffic vision research. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2072–2084, 2019.

[173] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[174] Pedro Henrique Feijó de Sousa, Jefferson Silva Almeida, Elene Firmeza Ohata, Fabrício Gonzalez Nogueira, Bismark Claure Torrico, Victor Hugo Costa de Albuquerque, Mohammad Mehedi Hassan, Neeraj Kumar, Md. Rafiul Hassan, and Pedro Pedrosa Rebouças Filho. Intelligent 3d objects classification for vehicular ad hoc network based on lidar and deep learning approaches. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19807–19816, 2022.

[175] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[176] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. pages 7464–7473, 07 2017.

[177] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[178] Berkeley DeepDrive, February 2022. [Online; accessed 28. Jul. 2023].

[179] Jordan J. Bird, Diego R. Faria, Anikó Ekárt, and Pedro P. S. Ayrosa. From simulation to reality: Cnn transfer learning for scene classification. In *2020 IEEE 10th International Conference on Intelligent Systems (IS)*, pages 619–625, 2020.

[180] tusimple-benchmark, July 2023. [Online; accessed 26. Jul. 2023].

[181] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *AAAI Conference on Artificial Intelligence*, 2017.

[182] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, 2010.

[183] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[184] Domain adaptation | ImageCLEF / LifeCLEF - Multimedia Retrieval in CLEF, July 2023. [Online; accessed 27. Jul. 2023].

[185] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech 256, Apr 2022.

[186] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[187] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 06 2010.

[188] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. *Domain-Adversarial Training of Neural Networks*, pages 189–209. Springer International Publishing, Cham, 2017.

[189] J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[190] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

[191] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. 09 2014.

[192] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1406–1415, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society.

[193] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[194] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[195] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, 2016.

[196] Andreas Geiger, P Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: the kitti dataset. *The International Journal of Robotics Research*, 32:1231–1237, 09 2013.

[197] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Sridhar, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? 10 2016.

[198] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[199] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5716–5726, 2017.

[200] Phani Nidadavolu, Saurabh Kataria, Jesús Villalba, and Najim Dehak. Low-resource domain adaptation for speaker recognition using cycle-gans. pages 710–717, 12 2019.

[201] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[202] B. Yamini, V. Prasanna, C. Ambhika, Mutturu Anuradha, B. Maheswari, Siva R, and Nalini Manogaran. A comprehensive survey of deep learning: Advancements, applications, and challenges. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11:445–453, 08 2023.

[203] Youssef Abadade, Anas Temouden, Hatim Bamoumen, Nabil Benamar, Yousra Chtouki, and Abdelhakim Senhaji Hafid. A comprehensive survey on tinyml. *IEEE Access*, 11:96892–96922, 2023.

[204] Simone Disabato and Manuel Roveri. Incremental on-device tiny machine learning. In *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, AIChallengeIoT '20, page 7–13, New York, NY, USA, 2020. Association for Computing Machinery.

[205] Q. Jodelet, X. Liu, Y. Phua, and T. Murata. Class-incremental learning using diffusion model for distillation and replay. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3417–3425, Los Alamitos, CA, USA, oct 2023. IEEE Computer Society.

[206] Golnoush Abaei Azam Davahli, Mahboubeh Shamsi and Arash Khosravi. Empirical analyses of genetic algorithm and grey wolf optimiser to improve their efficiency with a new multi-objective weighted fitness function for feature selection in machine learning classification: the roadmap. *Journal of Experimental & Theoretical Artificial Intelligence*, 35(2):171–206, 2023.

[207] Dr. Lachit Dutta and Swapna Bharali. Tinyml meets iot: A comprehensive survey. *Internet of Things*, 16:100461, 2021.

[208] Khadija Shaheen, Muhammad Hanif, Osman Hasan, and Muhammad Shafique. Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks. *Journal of Intelligent Robotic Systems*, 105, 05 2022.

[209] Leonardo Ravaglia, Manuele Rusci, Davide Nadalini, Alessandro Capotondi, Francesco Conti, and Luca Benini. A tinyml platform for on-device continual learning with quantized latent replays. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, PP:1–1, 10 2021.

[210] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209, 2020.

[211] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[212] Simone Disabato and Manuel Roveri. Tiny machine learning for concept drift. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2022.

[213] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[214] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, sep 1995.

[215] Yu Li, Zhongxiao Li, Lizhong Ding, Yuhui Hu, Wei Chen, and Xin Gao. Supportnet: a novel incremental learning framework through deep learning and support data. *bioRxiv*, 2018.

[216] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[217] Vijay Chahar, Sourabh Katoch, and Sumit Chauhan. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 02 2021.

[218] Gregory Morse and Kenneth O. Stanley. Simple evolutionary optimization can rival stochastic gradient descent in neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 477–484, New York, NY, USA, 2016. Association for Computing Machinery.

[219] Md Faisal, Farhan Fuad Abir, Mosabber Ahmed, and Md Atiqur Rahman Ahad. Exploiting domain transformation and deep learning for hand gesture recognition using a low-cost dataglove. *Scientific Reports*, 12:21446, 12 2022.

[220] Amritha Purushothaman and Suja Palaniswamy. Development of smart home using gesture recognition for elderly and disabled. *Journal of Computational and Theoretical Nanoscience*, 17:177–181, 01 2020.

[221] Veronica Naosekpam and Rupam Sharma. Machine learning in 3d space gesture recognition. *Jurnal Kejuruteraan*, 31:243–248, 10 2019.

[222] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, mar 2011.

[223] Gary M. Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202, 2019.

[224] Saurabh Gupta. Deep learning based human activity recognition (har) using wearable sensor data. *International Journal of Information Management Data Insights*, 1(2):100046, 2021.

[225] Walid Gomaa and Mohamed Khamis Omar. A perspective on human activity recognition from inertial motion data. *Neural Computing and Applications*, 35, 07 2023.

[226] Suraj Kumar Pandey, Sonia Sonia, Tushar Semwal, and Shivashankar B. Nair. Smart patch: An iot based anti child-trafficking solutio. In *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, pages 194–200, 2021.

[227] Anubha Parashar, Apoorva Parashar, Andrea F. Abate, Rajveer Singh Shekhawat, and Imad Rida. Real-time gait biometrics for surveillance applications: A review. *Image and Vision Computing*, 138:104784, 2023.

[228] ESP32 Wi-Fi & Bluetooth Modules I Espressif, January 2024. [Online; accessed 31. Jan. 2024].

[229] Juan Felipe Reyes, James Steven Montealegre, Yor Jaggy Castano, Christian Urcuqui, and Andres Navarro. Lstm and convolution networks exploration for parkinson's diagnosis. In *2019 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–4, 2019.

Publications

Publications under the thesis

Journals

- S. K. Pandey and S. B. Nair, "RoboDA: A Dataset for Domain Adaptation in Robot Vision" (**Accepted** at the Taylor and Francis IETE Technical Review journal)
- S. K. Pandey and S. B. Nair, "GADANN: A Virtual-to-Real Knowledge Transfer Adaptation method for Edge Devices" (**Major Revision Communicated** to the Elsevier Internet of Things journal)
- S. K. Pandey and S. B. Nair, "ChaoticImmuneNet: A Chaos-driven Immunity inspired Neural Network paradigm for Embodied Intelligence in Resource-Constrained Devices" (**Communicated** to the ACM Journal on Emerging Technologies in Computing Systems)

Conferences

- S. K. Pandey, Sonia, T. Semwal and S. B. Nair, "Smart Patch: An IoT based Anti Child-Trafficking Solution", 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), Indonesia, 2020, DOI: 10.1109/IoTaIS50849.2021.9359702
- S. K. Pandey and S. B. Nair, "Immuno-inspired Augmentation of Siamese Neural Networks for Multi-Class Classification", IVCNZ 2022 Image and Vision Computing New Zealand, 24 - 25 November 2022, DOI: 10.1007/978-3-031-25825-1_35
- S. K. Pandey and S. B. Nair, "Enhancing Siamese Neural Networks for Multiclass Classification: An Immuno-inspired approach", GECCO 2023, The Genetic and Evolutionary Computation Conference, 2023, Lisbon, Portugal. July 15-19, DOI: https://doi.org/10.1145/3583133.3595827
- S. K. Pandey and S. B. Nair,"Onboard Class Incremental Learning for Resource-Constrained scenarios using Genetic Algorithm and TinyML" The Genetic and Evolutionary Computation Conference, 2024, Melbourne, Australia, July 14-18, DOI: https://doi.org/10.1145/3638530.3654392

Other Publications

Conferences

- S. K. Pandey, "Neuroevolution of a Multi-Generator GAN", The 38th Annual AAAI Conference on Artificial Intelligence (AAAI-24), AAAI 2024, Vancouver, Canada, February 20-27, 2024, DOI: https://doi.org/10.1609/aaai.v38i21.30493

- K. Narayan Mohan and S. K. Pandey, "A Deep-Learning based real-time License Plate Recognition system for resource-constrained scenarios" (**Accepted** for the 27th International Conference on Pattern Recognition)
- H. Bijwe, S. K. Pandey and S. B. Nair, "Decentralized Multi-Robot Foraging using a Cloning Controller" (**Under internal review**)
