Advanced Sciences and Technologies for Security Applications

Mamoun Alazab
MingJian Tang *Editors*

Deep Learning Applications for Cyber Security

Springer

Advanced Sciences and Technologies for Security Applications

The series Advanced Sciences and Technologies for Security Applications comprises interdisciplinary research covering the theory, foundations and domain-specific topics pertaining to security. Publications within the series are peer-reviewed monographs and edited works in the areas of:

- biological and chemical threat recognition and detection (e.g., biosensors, aerosols, forensics)
- crisis and disaster management
- terrorism
- cyber security and secure information systems (e.g., encryption, optical and photonic systems)
- traditional and non-traditional security
- energy, food and resource security
- economic security and securitization (including associated infrastructures)
- transnational crime
- human security and health security
- social, political and psychological aspects of security
- recognition and identification (e.g., optical imaging, biometrics, authentication and verification)
- smart surveillance systems
- applications of theoretical frameworks and methodologies (e.g., grounded theory, complexity, network sciences, modelling and simulation)

Together, the high-quality contributions to this series provide a cross-disciplinary overview of forefront research endeavours aiming to make the world a safer place.

The editors encourage prospective authors to correspond with them in advance of submitting a manuscript. Submission of manuscripts should be made to the Editor-in-Chief or one of the Editors.

More information about this series at http://www.springer.com/series/5540

Mamoun Alazab • MingJian Tang
Editors

Deep Learning Applications for Cyber Security

Springer

Editors
Mamoun Alazab
Charles Darwin University
Casuarina, NT, Australia

MingJian Tang
Singtel Optus
Sydney, NSW, Australia

ISSN 1613-5113 ISSN 2363-9466 (electronic)
Advanced Sciences and Technologies for Security Applications
ISBN 978-3-030-13056-5 ISBN 978-3-030-13057-2 (eBook)
https://doi.org/10.1007/978-3-030-13057-2

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

ICT is arguably the greatest general purpose tool currently invented by mankind. It has turned the carts into autonomous vehicles, the printing press into the Internet, the telephone into smart phones, the light bulb into LiFi, electricity into megabytes and penicillin into endolysins. History is likely to show ICT will surpass the importance of the printing press, electricity, telephone, light bulb and penicillin.

As society continue to depend on technology, more we are presented with increased risk exposure, just as technology brings ever greater benefits, it also brings ever greater threats. It becomes a focal point for cybercrime, industrial espionage, cyberattacks and cyberwarfare. Therefore, cybersecurity is not optional and becomes a high priority.

Threats in cybersecurity evolve over time as attackers change their behaviour. Static, rule-based cybersecurity systems to identify behaviour that indicates a threat only goes so far in the current environment. Artificial intelligence (AI) applications appear to lift many barriers on the capacity of ICT to disrupt, change and create quantum-level transformations. The use of AI, especially deep learning, provides means to proactively advance cybersecurity objectives, detecting, modelling, monitoring, learning, preventing, analysing and augmenting the resources to defend against various threats to sensitive data and systems in dynamic manner.

Right now, technology is a force multiplier without equal. Tomorrow, technology may become the force. Humanity must find ways to leverage and control technology for good; or we shall succumb to our own invention. Continuous research and development in AI and cybersecurity are important for sustainable growth of technology.

I thank and congratulate all the authors for this timely publication.

President, Australian Computer Society
Secretary General SEARCC
Sydney, NSW, Australia

Yohan Ramasundara, FACS

Acknowledgements

We are grateful to many individuals and organisations for assisting with the preparation of this book. We would like to acknowledge the support from Charles Darwin University, Department of Corporate and Information Services, Northern Territory Government of Australia and Singtel Optus. We appreciate the support made by our respective employers who allowed us time in which to edit the chapters and attend to the many tasks that had to be undertaken to finalise the publication. We are grateful to all the authors for their insights and excellent contributions to this book. We greatly appreciate the professional assistance of the editorial staff of Springer. Finally, we would like to thank our families for their support and patience during the months we spent developing this book.

Mamoun Alazab
MingJian Tang

Contents

About the Authors

Mamoun Alazab is an Associate Professor at the College of Engineering, IT and Environment at Charles Darwin University, Australia. He received his Ph.D. degree in Computer Science from the School of Science, Information Technology and Engineering at the Federation University of Australia. He is a cyber security researcher and practitioner with industry and academic experience. Alazab's research is multidisciplinary that focuses on cyber security and digital forensics of computer systems with a focus on cybercrime detection and prevention including cyber terrorism and cyber warfare. He has more than 100 research papers. He delivered many invited and keynote speeches, 22 events in 2018 alone. He convened and chaired more than 50 conferences and workshops. He works closely with government and industry on many projects, including Northern Territory (NT) Department of Information and Corporate Services, IBM, Trend Micro, the Australian Federal Police (AFP), the Australian Communications and Media Authority (ACMA), Westpac, United Nations Office on Drugs and Crime (UNODC), and the Attorney General's Department. He is a Senior Member of the IEEE. He is the Founder and Chair of the IEEE Northern Territory (NT) Subsection.

Sanjay Chawla is a Research Director in the Qatar Computing Research Institute (QCRI), Hamad Bin Khalifa University (HBKU), Doha, Qatar. Prior to joining QCRI, he was a Professor in the School of Information Technologies, University of Sydney, Australia. His main area of research is data mining and machine learning. More specifically, he has been working on three problems of contemporary interest: outlier detection, imbalanced classification, and adversarial learning. His research has been published in leading conferences and journals and has been recognized by several best paper awards. He serves on the editorial board of the IEEE Transactions on Knowledge and Data Engineering and the *Data Mining and Knowledge Discovery*. He served as the Head of school from 2008 to 2011 and was an Academic Visitor at Yahoo! Labs, Bangalore, in 2012. He served as PC Cochair for PAKDD 2012 and SDM 2016.

Adriel Cheng is a Research Scientist in the Cyber and Electronic Warfare Division of Australia's Defence Science and Technology Group. He holds a B.Eng. and Ph.D.

from the University of Adelaide, South Australia, and was awarded the University of Adelaide Medals for both degrees. His research interests include machine learning, and characterization, modelling, and analysis of telecommunication IP networks.

Hong Gunn Chew is an Associate Lecturer in the School of Electrical and Electronic Engineering at the University of Adelaide. He is the Electronics Practical Coordinator and final year projects coordinator in the school. His research interests are in machine learning, autonomous systems, cybersecurity, and energy management.

Mounir Ghogho received his Ph.D. from the National Polytechnic Institute of Toulouse, France, in 1997. In 2001, he joined the University of Leeds, where he was promoted to a Full Professor in 2008. While still affiliated with the University of Leeds, he joined the Université Internationale de Rabat in 2010, where he is currently the Director of the TICLab (ICT Research Laboratory) and a Scientific Advisor to the President. He is a member of the steering committee of the IEEE Transactions on Signal and Information Processing. He was a Recipient of the 2013 IBM Faculty Award and the 2000 UK Royal Academy of the Engineering Research Fellowship. He chaired many conferences and workshops. He served as an Associate Editor for the *IEEE Transactions on Signal Processing* and the *IEEE Signal Processing Letters*. He is currently an Associate Editor of IEEE Signal Processing Magazine.

Richard Harang is a Principal Data Scientist at Sophos with over 7 years of research experience at the intersection of computer security, machine learning, and privacy. Prior to joining Sophos, he served as a Scientist at the US Army Research Laboratory, where he led the research group investigating the applications of machine learning and statistical analysis to problems in network security. He received his Ph.D. in statistics from the University of California, Santa Barbara. Research interests include randomized methods in machine learning, adversarial machine learning, and ways to use machine learning to support human analysis. By day, he uses bad guys to catch math. By night, he teaches killer robots to protect his garden from squirrels.

Mofakharul Islam received his Ph.D. in computer science from the Federation University of Australia, School of Science, Information Technology and Engineering. He is a Lecturer in cybersecurity at the Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia. Currently, he is engaged in advanced research in deep learning, machine learning, computer vision, digital forensics, forensic imaging, and image processing (i.e., modelling, analysis, design, and verification) with a particular focus on the prevention of cybercrime (e.g., cyberbullying/cyber harassment, scam, financial fraudulence, botnet, DDoS attack, malware, phishing, identity theft, child pornography, child abuse, online abuse, XSS, MITM, etc.) and the analysis/modelling/simulation of biological and physical phenomena including medical/industrial imaging. He has published widely in the field.

Muhammad Ismail (S'10–M'13–SM'17) received B.Sc. (with Hons.) and M.Sc. degrees in electrical engineering (electronics and communications) from Ain Shams University, Cairo, Egypt, in 2007 and 2009, respectively, and Ph.D. in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2013. He is currently an Assistant Research Scientist with the Department of Electrical and Computer Engineering, Texas A&M University at Qatar, Doha, Qatar. His research interests include smart grids, wireless networks, and cyber-physical security. He was the Co-recipient of the Best Paper Awards in the IEEE ICC 2014, the IEEE GLOBECOM 2014, the SGRE 2015, and the GREEN 2016. He is an Associate Editor for the IEEE Transactions on Green Communications and Networking and IET Communications. He was an Editorial Assistant for the IEEE Transactions on Vehicular Technology from January 2011 to July 2013. He was a Workshop Cochair in IEEE GreenCom 2018, TPC Cochair in IEEE VTC 2017 and 2016, Publicity and Publication Cochair in CROWNCOM 2015, and Web-Chair in IEEE INFOCOM 2014. He has been a Technical Reviewer for several IEEE conferences and journals.

Alireza Jolfaei is a Lecturer (Assistant Professor) of Cyber Security at the Department of Computing, Macquarie University, Sydney, NSW, Australia. He received his Ph.D. degree in Applied Cryptography from Griffith University, Gold Coast, QLD, Australia. Before he joined Macquarie University, he worked as a Lecturer (Assistant Professor) at Federation University Australia, Ballarat, VIC, Australia, and as an Assistant Professor of Computer Science at Temple University, Philadelphia, Pennsylvania, USA. His current research areas include cryptology, cyber-physical systems security, IoT security, and AI and machine learning for cyber security. He has authored over 40 peer-reviewed journal and conference articles on topics related to cyber security. He received the prestigious IEEE Australian Council award for his research paper published in the *IEEE Transactions on Information Forensics and Security*. He has received multiple awards for academic excellence, university contribution, and inclusion and diversity support. He is currently serving as the Chair of Computational Intelligence Society in IEEE Victoria Section.

Mohasinina Binte Kamal received her M.Sc.in electrical engineering from The University of Akron, Akron, OH, USA, in 2018, and B.Sc. in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2014. She is working toward her Ph.D. in electrical and computer engineering from the University of California, Riverside, CA, USA, to be completed in Fall 2018. Her research interests include the cyber-physical energy management system, cybersecurity in power distribution, and applications of machine learning techniques.

Amara Dinesh Kumar is a Master's student from Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India, since July 2017. He received his B.Tech. in electronics and communication from TKR College of Engineering, Hyderabad, in 2014. His research areas are cybersecurity, network traffic analysis, vehicle security, adversarial cybersecurity, security for cyber-physical systems,

IOT security, machine learning, deep learning, and reinforcement learning. He has participated in several international shared tasks and hackathons. His research profile is available at https://sites.google.com/view/dineshkumaramara/home/.

William Younghoo Lee is a Data Scientist at Sophos. Prior to joining Sophos, he developed malware classification systems at Symantec and mobile platforms at Samsung Electronics. He has presented on mobile malware topics at security conferences. His research interests include deep learning models for malware classification and categorization in web content.

Cheng-Chew Lim received the B.Sc. (Hons.) in electronic and electrical engineering and Ph.D. in electronic and electrical engineering from Loughborough University, Leicestershire, UK. He is currently a Professor with the University of Adelaide, Adelaide, SA, Australia. His research interests include control and systems theory, autonomous systems, cyber-physical systems, and optimization techniques and applications. Dr. Lim serves as an Associate Editor for the *IEEE Transactions on Systems, Man, and Cybernetics: Systems* and an Editorial Board Member for the *Journal of Industrial and Management Optimization* and has served as a Guest Editor for a number of journals, including *Discrete and Continuous Dynamical Systems-Series B*.

Wei Liu is a Senior Lecturer at the Advanced Analytics Institute, School of Software, Faculty of Engineering and IT, University of Technology, Sydney. Before joining UTS, he was a Research Fellow at the University of Melbourne and then a Machine Learning Researcher at NICTA working with the transportation industry. He obtained his Ph.D. from the University of Sydney. He works in the areas of machine learning and data mining and has been publishing papers in the research topics of game theory, tensor factorization, graph mining, causal inference, data imbalance handling, spatiotemporal data mining, and anomaly detection.

Brian C. Lovell is a Professor with the School of ITEE, The University of Queensland, Australia. He was Research Leader in National ICT Australia (2006–2012) and Research Director of the Security and Surveillance Research Group in the School of ITEE, UQ. He was President of the Australian Pattern Recognition Society (1995–2005), Senior Member of the IEEE, Fellow of the World Innovation Forum, Fellow of the IEAust, and Voting Member for Australia on the governing board of the International Association for Pattern Recognition since 1998. Professor Lovell served on the Editorial Board of *Pattern Recognition Letters* and reviews for many of the major journals in the fields of computer vision and pattern recognition as well as CVPR, ICCV, and ECCV.

Abdun Nur Mahmood received his Ph.D. from the University of Melbourne, Australia, in 2008, M.Sc. (Research) in computer science, and B.Sc. in applied physics and electronics from the University of Dhaka, Bangladesh, in 1999 and 1997, respectively. He is currently in the Department of Computer Science, School of Engineering and Mathematical Sciences, La Trobe University. Previously, he worked at UNSW, RMIT, Melbourne University, and the University of Dhaka.

He has been working as an Academic in Computer Science since 1999. In 2011, he joined UNSW as a Lecturer in Computer Science until he joined La Trobe University, Melbourne, in 2017, where he is currently working as a Senior Lecturer in Cybersecurity. His research interests include data mining techniques for scalable network traffic analysis, anomaly detection, and industrial SCADA security. He has published his work in various IEEE Transactions and A-tier international journals and conferences.

Mohamed Mahmoud received his Ph.D. from the University of Waterloo in April 2011. From May 2011 to May 2012, he worked as a Postdoctoral Fellow in the Broadband Communications Research Group, University of Waterloo. From August 2012 to July 2013, he worked as a Visiting Scholar in the University of Waterloo and a Postdoctoral Fellow in Ryerson University. Currently, Dr. Mahmoud is an Associate Professor in Department Electrical and Computer Engineering, Tennessee Tech University, USA. The research interests of Dr. Mahmoud include security and privacy preserving schemes for smart grid communication network, mobile ad hoc network, sensor network, and delay-tolerant network. Dr. Mahmoud has received NSERC PDF Award. He won the Best Paper Award from IEEE International Conference on Communications (ICC 2009), Dresden, Germany, 2009. Dr. Mahmoud is the Author for more than 23 papers published in major IEEE conferences and journals, such as INFOCOM conference and IEEE Transactions on Vehicular Technology, Mobile Computing, and Parallel and Distributed Systems. He serves as an Associate Editor in Springer journal of *Peer-to-Peer Networking and Applications*. He served as a Technical Program Committee Member for several IEEE conferences and as a Reviewer for several journals and conferences such as IEEE Transactions on Vehicular Technology, IEEE Transactions on Parallel and Distributed Systems, and the journal of *Peer-to-Peer Networking and Applications*.

Des McLernon received his B.Sc. in electronic and electrical engineering and M.Sc. in electronics, both from the Queen's University of Belfast, Northern Ireland. He then worked on radar systems research and development with Ferranti Ltd in Edinburgh, Scotland, and later joined Imperial College, University of London, where he took his Ph.D. in signal processing. His research interests are broadly within the domain of signal processing for wireless communications (in which area he has published over 320 journal and conference papers). He has supervised over 45 Ph.D. students and is Associate Editor of IET journal *Signal Processing*. His current research projects include PHY layer security, M2M communications/caching in heterogeneous networks, energy harvesting for communications, robotic communications, SDNs, distributed sensing, stochastic geometry, multi-packet reception through the wall radar, drone small cell communications, and spectrum sensing for cognitive radio.

Gihan J. Mendis received his B.Sc. Eng.(Hons.) in electronics and telecommunication engineering from the University of Moratuwa, Moratuwa, Sri Lanka, in 2012, and M.Sc. in electrical engineering from The University of Akron, Akron, OH, USA, in 2016. He is currently working toward his Ph.D. in electrical and computer

engineering from The University of Akron, Akron, OH, USA. He has worked at Ocius Technologies LLC as the Principal Scientist from August 2016 to December 2016. His research interests include the deep learning algorithms, decentralized deep learning, cybersecurity, cognitive radar sensors, and cognitive radio.

Lotfi Mhamdi received his Master of Philosophy (MPhil.) in computer science from the Hong Kong University of Science and Technology (HKUST) in 2002 and Ph.D. in computer engineering from Delft University of Technology (TU Delft), the Netherlands, in 2007. He continued his work at TU Delft as Postdoctoral Researcher, working on high-performance networking topics. Since July 2011, he has been a Lecturer with the School of Electronic and Electrical Engineering at the University of Leeds, UK. Dr. Mhamdi is/was a Technical Program Committee Member in various conferences, including the IEEE International Conference on Communications (ICC), the IEEE GLOBECOM, the IEEE Workshop on High Performance Switching and Routing (HPSR), and the ACM/IEEE International Symposium on Networks-on-Chip (NOCS). His research work spans the area of high-performance networks including the architecture, design, analysis, scheduling, and management of high-performance switches and Internet routers. He is a Member of the IEEE.

Kyle Millar is a first year Ph.D. candidate from the University of Adelaide's School of Electrical and Electronic Engineering. In his final year of undergraduate studies, Kyle was introduced to the topic of deep learning techniques for malicious traffic detection. For his work on this project, he achieved first class honors as well as DST Group's Surveillance Systems Undergraduate Prize. Extending this research into his Ph.D., Kyle's main research topic now focuses on device inference through network traffic forensics.

Mahmoud Nabil is currently a Graduate Research Assistant in the Department of Electrical Computer Engineering, Tennessee Tech University, USA, and pursuing his Ph.D. in the same department. He received B.S. and M.S. degrees in computer engineering from Cairo University, Cairo, Egypt, in 2012 and 2016, respectively. His research interests include machine learning, cryptography and network security, smart grid and AMI networks, and vehicular ad hoc networks.

Prabaharan Poornachandran is a Professor at Amrita Vishwa Vidyapeetham. He has more than two decades of experience in computer science and security areas. His areas of interests are malware, critical infrastructure security, complex binary analysis, AI, and machine Learning.

Khalid Qaraqe (M'97–SM'00) was born in Bethlehem. He received his B.S. (with Hons.) in electrical engineering from the University of Technology, Baghdad, Iraq, in 1986; M.S. in electrical engineering from the University of Jordan, Amman, Jordan, in 1989; and Ph.D. in electrical engineering from Texas A&M University, College Station, TX, USA, in 1997. From 1989 to 2004, he has held a variety positions in many companies and has more than 12 years of experience in the telecommunication industry. He has worked on numerous GSM, CDMA, and

WCDMA projects and has experience in product development, design, deployments, testing, and integration. In July 2004, he joined the Department of Electrical and Computer Engineering at Texas A&M University at Qatar, Doha, Qatar, where he is currently a Professor. He has been awarded 15 research projects consisting of more than USD 9.0 M from local industries in Qatar and the Qatar National Research Fund (QNRF). He has authored 90 journal papers in top IEEE journals and published and presented 194 papers at prestigious international conferences. He has 13 book chapters published, 2 books slated to appear in 2015, and 4 patents and presented 5 tutorials and talks. His research interests include communication theory and its application to design and performance, analysis of cellular systems and indoor communication systems, mobile networks, broadband wireless access, cooperative networks, cognitive radio, diversity techniques, and beyond 4G systems. He was the Recipient of the Itochu Professorship Award (2013–2015), the Best Researcher Award from QNRF 2013, the Best Paper Award from the IEEE First Workshop on Smart Grid And Renewable Energy (March 2015), the Best Paper Award from the IEEE GLOBECOM 2014, the Best Poster Award from IEEE DySPAN Conference (October 2012), the TAMUQ Research Excellence Award (April 2010), the Best Paper Award from ComNet 2010, the Best Paper Award from CROWNCOM 2009, and the Best Paper Award from ICSPC 2000 and 2007.

Joshua Saxe is Chief Data Scientist at major security vendor, Sophos, where he leads a security data science research team. He leads the data science team with a particular focus on inventing, evaluating, and deploying deep learning detection models in support of a next-gen endpoint security solutions. He's also a Principal Inventor of Sophos' neural network-based malware detector, which defends tens of millions of Sophos customers from malware infections. Before joining Sophos, Joshua spent 5 years leading DARPA-funded security data research projects for the US government.

Erchin Serpedin (F'13) received the specialization degree in signal processing and transmission of information from École supérieure d'électricité (SUPELEC), Paris, France, in 1992; M.Sc. from the Georgia Institute of Technology, Atlanta, GA, USA, in 1992; and Ph.D. in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in January 1999. He is currently a Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. He has authored 2 research monographs, 1 textbook, 9 book chapters, 110 journal papers, and 180 conference papers. His research interests include signal processing, biomedical engineering, bioinformatics, and machine learning. He is currently an Associate Editor for the IEEE Signal Processing Magazine and the Editor-in-Chief of the journal *EURASIP Journal on Bioinformatics and Systems Biology* (Springer). He was an Associate Editor for a dozen journals, such as the *IEEE Transactions on Information Theory*, the *IEEE Transactions on Signal Processing*, the *IEEE Transactions on Communications*, the *IEEE Signal Processing Letters*, the *IEEE Communications Letters*, the *IEEE Transactions on Wireless Communications*, *Signal Processing* (Elsevier), *Physical Communication* (Elsevier), and *EURASIP Journal on Advances in Signal Processing* and a Technical

Chair for five major conferences. He was the Recipient of numerous awards and research grants.

Mostafa Shahin graduated from Electrical Engineering Department at Ain Shams University (2003) and received his M.Sc. in electronics and communications engineering from Cairo University (2015). From 2004, he worked in the speech research group at RDI Company, Egypt. In 2011, he joined Texas A&M University at Qatar as a Research Associate. He is currently a Ph.D. candidate in the Electrical Engineering and Telecommunications Department at the University of New South Wales (UNSW), Sydney, Australia. His research focuses on applying machine learning techniques for speech and biomedical signal processing.

K. P. Soman has 25 years of research and teaching experience at Amrita School of Engineering, Coimbatore. He has around 150 publications in national and international journals and conference proceedings. He has organized a series of workshops and summer schools in advanced signal processing using wavelets, kernel methods for pattern classification, deep learning, and big data analytics for industry and academia. He authored books on *Insight Into Wavelets: From Theory to Practice*, *Insight into Data Mining: Theory and Practice*, *Support Vector Machines and Other Kernel Methods*, and *Signal and Image Processing: The Sparse Way*, published by Prentice Hall, New Delhi, and Elsevier. More details available at https://nlp.amrita.edu/somankp/

Tuan Anh Tang received his B.E. in electronic and communication from Danang University of Science and Technology, Vietnam, in 2013. He is currently pursuing his Ph.D. at the University of Leeds, UK. His research interests include software-defined networking (SDNs), security, intrusion detection, machine learning, and deep learning.

Harish Thodupunoori is a Master's student from Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India, since July 2017. He received his B.Tech. in electronics and communication from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, in 2011. His research areas are cybersecurity, network traffic analysis, vehicle security, machine learning, and deep learning.

Sitalakshmi Venkatraman earned her Ph.D. in computer science, with a doctoral thesis titled "Efficient Parallel Algorithms for Pattern Recognition," from National Institute of Industrial Engineering in 1993. Prior to that, she was awarded M.Sc. (mathematics) in 1985 and M.Tech. (computer science) in 1987, both from Indian Institute of Technology (Madras), and subsequently M.Ed. from the University of Sheffield in 2001. Sita has more than 30 years of work experience both in industry and academics – developing turnkey projects for IT industry and teaching a variety of IT courses for tertiary institutions, in India, Singapore, New Zealand, and Australia, since 2007. She is currently the Discipline Leader and Senior Lecturer in information technology at Melbourne Polytechnic. She specializes in applying efficient computing models and data mining techniques for various

industry problems and recently in the e-health, e-security, and e-business domains through collaborations with industry and universities in Australia. She has published 7 book chapters and more than 130 research papers in internationally well-known refereed journals and conferences. She is a Senior Member of professional societies and editorial boards of international journals and serves as Program Committee Member of several international conferences every year.

R. Vinayakumar is a Ph.D. student in the Computational Engineering & Networking, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India, since July 2015. He has received B.C.A from JSS college of Arts, Commerce and Science, Ooty Road, Mysore, in 2011 and M.C.A from Amrita Vishwa Vidyapeetham, Mysore, in 2014. He has several papers in machine learning applied to cybersecurity. His Ph.D. work centers on application of machine learning (sometimes deep learning) for cybersecurity and discusses the importance of natural language processing, image processing, and big data analytics for cybersecurity. He has participated in several international shared tasks and organized a shared task on detecting malicious domain names (DMD 2018) as part of SSCC 2018 and ICACCI 2018. More details available at https://vinayakumarr.github.io/

Yang Wang earned his Ph.D. from School of Computer Science & Engineering, the University of New South Wales, Kensington, Australia. He is currently a Huangshan Professor at media computing lab at Hefei University of Technology, China. Before that, he has been an Associate Professor at the Faculty of Electronic Engineering, Dalian University of Technology, China, and a Research Fellow at the University of New South Wales, Australia. So far, he has published 50 research papers, most of which appeared in the major venues, such as IEEE TIP, IEEE TNNLS, IEEE TCYB, IEEE TKDE, IEEE TMM, Neural Networks, Pattern Recognition, *VLDB Journal*, IJCAI, ACM Multimedia, ACM SIGIR, IEEE ICDM, ACM CIKM, etc. Yang was the winner of Best Research Paper Runner-up Award for PAKDD 2014 and was a Program Committee Member for various leading conferences meanwhile serving as Guest Editor for *Pattern Recognition Letters, Multimedia Tools and Applications*.

Paul Watters is a leading expert in cybersecurity. He is Professor of cybersecurity at La Trobe University in Melbourne. He began his first R&D role in security in 2002, joining the CSIRO's Networking Applications and Technologies (NAT) Group and leading a program in secure, distributed storage. He is a Chartered IT Professional and Cybersecurity Researcher with 16 years of experience in cyber R&D. He is a Distinguished Public Speaker and Published Author, with 3,196 citations to his work (h-index=26, i10-index=73). His work has been funded by numerous research partners, including Westpac, IBM, the Australian Federal Police, the Attorney General's Department, the Motion Picture Association of America, End Child Prostitution and Trafficking, and the Oceania Cyber Security Centre. He is a Fellow of the British Computer Society, a Senior Member of the IEEE, a Member of the Australian Psychological Society, and a Chartered IT Professional. He currently holds an ARC Discovery grant in the area of child exploitation demand reduction and an OCSC grant investigating malicious advertising.

Jin Wei received her B.E. from the Beihang University, Beijing, China, in 2004; M.S. in electrical engineering from the University of Hawaii at Manoa, Honolulu, HI, USA, in 2008; and Ph.D. in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2014. She worked as a Postdoctoral Fellow in National Renewable Energy Laboratory from April 2014 to July 2014. She is currently an Assistant Professor in electrical and computer engineering with The University of Akron, Akron, OH, USA, and the Director of the Cyber-Physical-Social Systems Design Lab. Her research interests include the smart energy systems, cyber-physical-social systems security and privacy, renewable energy integration, social networks, and cognitive wired/wireless communication networks.

Lin Wu was awarded a Ph.D. from the University of New South Wales, Sydney, Australia, in 2014. She has intensively published academic papers in CVPR, ACM Multimedia, IJCAI, ACM SIGIR, IEEE TIP, IEEE TNNLS, IEEE TCYB, IEEE TMM, Neural Networks, and Pattern Recognition. Her research interests include deep representation learning, computer vision, pattern recognition, and machine intelligence.

Zhizhou Yin is a Ph.D. student from the University of Sydney. His research focuses are on data mining and machine learning. He has published journal paper in the research topic of adversarial learning. Besides his study, he is a Data Scientist at the Universities Admissions Centre (UAC). He works on providing university admission services to high school students based on machine learning techniques.

Syed Ali Raza Zaidi received his Ph.D. from the School of Electronic and Electrical Engineering, University of Leeds, Leeds. From 2011 to 2013, he was associated with the International University of Rabat as a Lecturer. From 2013 to 2015, he was associated with the SPCOM Research Group, US Army Research Laboratory. He is currently an Assistant Professor in wireless communication and sensing systems with the University of Leeds. He has published over 90 papers in leading IEEE conferences and journals. He is also an Active Member of the EPSRC Peer Review College. He is a EURASIP Local Liaison for UK and also a General Secretary of the IEEE Technical Subcommittee on Backhaul and Fronthaul Networks. He served as an Editor for the IEEE Communication Letters. He is currently an Associate Technical Editor of IEEE Communication Magazine

Adversarial Attack, Defense, and Applications with Deep Learning Frameworks

Zhizhou Yin, Wei Liu, and Sanjay Chawla

Abstract In recent years, deep learning frameworks have been applied in many domains and achieved promising performance. However, recent work have demonstrated that deep learning frameworks are vulnerable to adversarial attacks. A trained neural network can be manipulated by small perturbations added to legitimate samples. In computer vision domain, these small perturbations could be imperceptible to human. As deep learning techniques have become the core part for many security-critical applications including identity recognition camera, malware detection software, self-driving cars, adversarial attacks have become one crucial security threat to many deep learning applications in real world. In this chapter, we first review some state-of-the-art adversarial attack techniques for deep learning frameworks in both white-box and black-box settings. We then discuss recent methods to defend against adversarial attacks on deep learning frameworks. Finally, we explore recent work applying adversarial attack techniques to some popular commercial deep learning applications, such as image classification, speech recognition and malware detection. These projects demonstrate that many commercial deep learning frameworks are vulnerable to malicious cyber security attacks.

Keywords Adversarial learning · Deep learning · Cyber security

Z. Yin (✉)
School of Information Technologies, University of Sydney, Sydney, NSW, Australia
e-mail: zyin4256@uni.sydney.edu.au

W. Liu
Advanced Analytics Institute, University of Technology Sydney, Sydney, NSW, Australia
e-mail: wei.liu@uts.edu.au

S. Chawla
Qatar Computing Research Institute, Doha, Qatar
e-mail: schawla@qf.org.qa

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_1

1 Introduction

Deep learning techniques have become one of most popular sub-field of machine learning in recent years, and have been successfully applied in many domains including computer vision, speech recognition, natural language processing [15]. Despite the promising performance in those domains, recent research have demonstrated that deep learning frameworks are vulnerable to adversarial attacks which manipulate a small subset of the original data to "fool" a trained neural network. The adversarial learning work with deep learning frameworks are mostly applied in computer vision domain, as deep learning frameworks, such as Convolutional Neural Network (CNN), have become the key technique of this domain. Adversarial attacks in computer vision domain are more meaningful as adversaries tend to manipulate human-indiscernible changes to a normal image in order to mislead the deep learning model to make mistakes. However, adversarial learning is also applied in other domains e.g. speech recognition, natural language processing, etc. This chapter will introduce recent work on adversarial attacks with deep learning frameworks, and then discuss common approaches for defending against adversarial attacks. In addition, we will also explore recent adversarial learning applications in real world deep learning frameworks which might result in cyber security related vulnerabilities.

2 Methods for Generating Adversarial Samples

Since Szegedy et al. [30] demonstrated that adding certain hardly perceptible perturbation can cause trained neural network to mis-classify an image, various of methods of generating adversarial samples have been proposed in recent years. According to Papernot et al. [24], the adversarial tasks can be grouped by the adversarial goals and information available for the attack, which is illustrated in Fig. 1.

As shown in Fig. 1, there are four different adversarial goals as listed in the horizontal axis, and the complexity of them are increasing along the axis:

1. **Confidence reduction**: The adversary tries to reduce the confidence of the classification to the correct label.
2. **Misclassification**: The adversary tries to lead the classification to any class different from the original class
3. **Targeted misclassification**: The adversary aims to force the classification to a specific target class different from the original class.
4. **Source and target misclassification**: The adversary forces the classification of a certain type of input to a specific target class different from the original class.

The vertical axis in Fig. 1 indicates the different types of availability of the model and data knowledge for the adversarial attack. The adversarial attacks become more difficult as the available information decreasing:

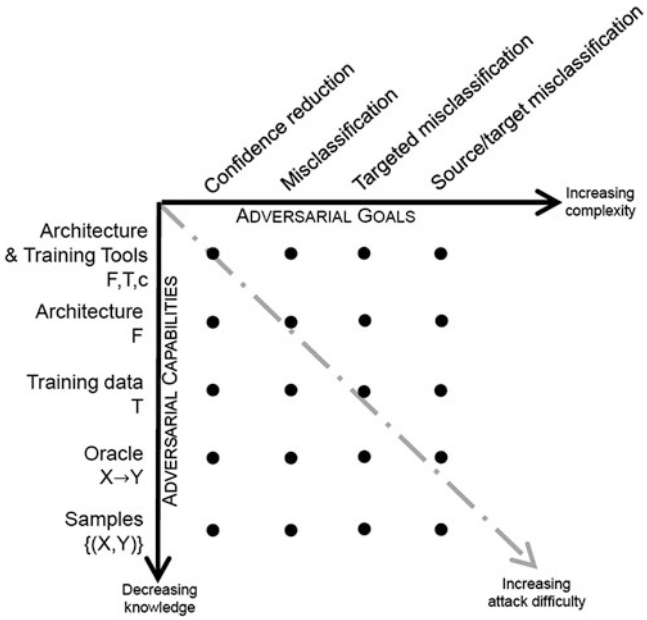


Fig. 1 Different types of attacks in adversarial learning [24]

1. **Network architecture and training data**: The adversary has the information of the neural network's architecture F which includes number and type of layers, neurons, activation functions, weight, bias matrices from the training phase, and the associated loss function c. Besides, the adversary also has the access to the entire training dataset T. It could be the perfect setting for adversarial learning but unrealistic for most of practical project.
2. **Network architecture**: This adversary has the knowledge of the network architecture F as described above but nothing else such as training data.
3. **Training data**: This adversary has the access to the training dataset T or a surrogate dataset sampled from the same distribution of the training dataset. However, the adversary has no information about the neural network architecture. This type of attack might involve the process of training another neural network with the available training dataset to approximate the model to attack.
4. **Oracle**: This adversary can only use the neural network as an "oracle". The attacker can get the output classifications for a given input, but has no access to any information of the neural network or training data. The attacker might need to analyze the relationship between changes in inputs and outputs to generate adversarial samples.
5. **Samples**: This adversary is able to collect some input and output pairs of the neural network, but unable to give inputs and observe the difference of output.

Table 1 List of adversarial attack methods introduced in this section and their types

	White/Black-box	Targeted/Not targeted
Box-constrained L-BFGS approach [30]	White-box	Targeted
Fast gradient sign method [7]	White-box	Not targeted
Basic iterative method [13]	White-box	Not targeted
Iterative least-likely class method [13]	White-box	Targeted
Jacobian-based saliency map attack [24]	White-box	Targeted
DeepFool [21]	White-box	Not targeted
Carlini & Wagner attack [3]	White-box	Targeted
Transferability based approach [23]	Black-box	Targeted
Houdini [4]	Black-box	Targeted

The adversarial attacks with neural network architecture knowledge are also known as **white-box attack**. By contrast, adversary performs attacks in a **black-box attack** setting when they have no access to the architecture or parameters of the model.

In this section, we will introduce nine adversarial attack methods which represent different types of adversarial attack technologies. Table 1 lists the reviewed methods as long as their types.

2.1 Box-Constrained L-BFGS Approach

The first known work demonstrating that adversarial samples can fool deep learning models was reported by Szegedy et al. [30]. In this work, a Box-constrained L-BFGS approach is used to compute imperceptibly small perturbation to the original image, so that it is no longer classified correctly by the network.

To formally describe the process, let $f : \mathbb{R}^m \rightarrow \{1 \ldots k\}$ denote the model we used to classify an image consists of pixel value vectors to one of the k labels. The loss function of f is $loss_f : \mathbb{R}^m \times \{1 \ldots k\} \rightarrow \mathbb{R}^+$. To let the classifier f to classify image $x \in \mathbb{R}^m$ to a target label $l \in \{1 \ldots k\}$, we want to solve the following optimization problem:

$$\min_r \|r\|_2 \;\; s.t. \;\; f(x+r) = l; x + r \in [0,1]^m \tag{1}$$

The exact computation is a hard problem for $f(x) \neq l$, which means the target label is different from the original label of x. So the author propose an approximate solution of it by using a box-constrained L-BFGS approach. A line-search is performed to find the minimum $c > 0$ for which the minimizer r of the following problem satisfies $f(x + r) = l$.

$$\min_r c|r| + loss_f(x + r, l) \;\; s.t. \;\; x + r \in [0,1]^m \tag{2}$$

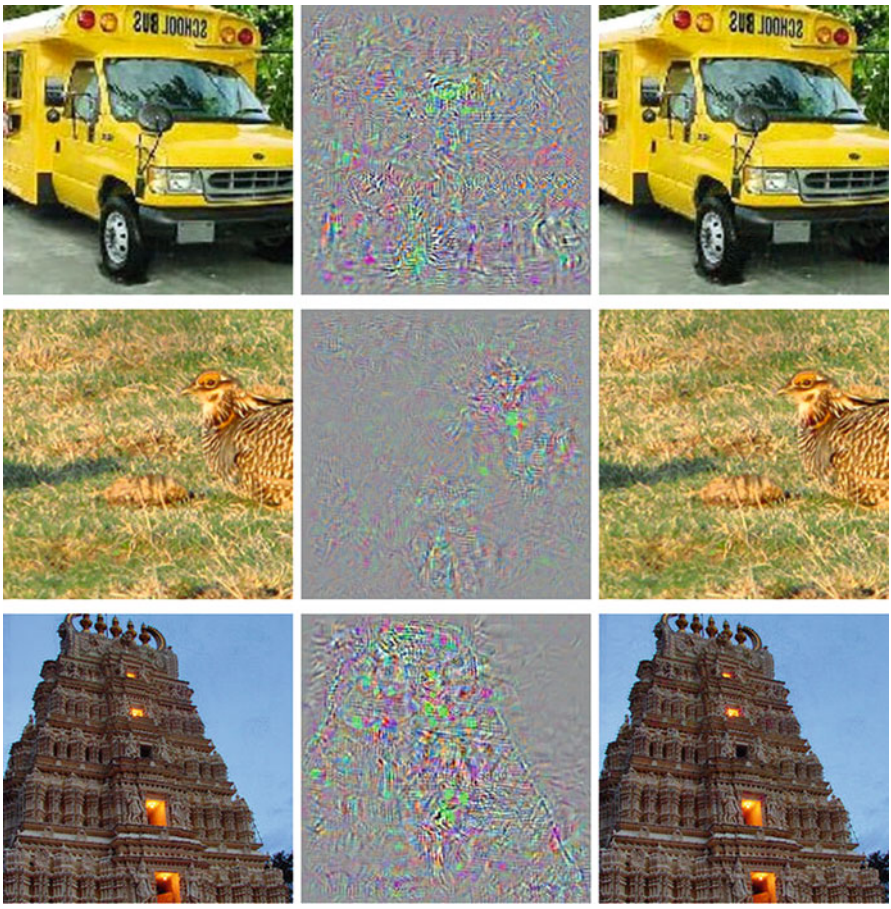


Fig. 2 Adversarial samples generated on AlexNet dataset. Left column is the original image, middle column is the computed perturbations, and right column is the adversarial sample. The images in the right column are predicted incorrectly to be an ostrich, Struthio, and camelus separately [30]

This function would have the exact solution if the classifier have a convex loss function. However, neural networks are generally non-convex, so they end up with an approximation.

This approach has been tested on three different datasets: MNIST [17], AlexNet [12], and QuocNet [14]. It is able to generate adversarial samples with small perturbations on all three datasets to fool the neural network classifier. Figures 2 and 3 show a few adversarial samples generated by this approach in multi-classification and binary classification scenarios separately. As the figures shown, the difference between generated images and original images are all indistinguishable to human but enough to fool the system.

Fig. 3 Adversarial samples generated on QuoNet dataset. The image on the left is classified as a car, while the one in the middle is classified as not a car. The difference between these two images are showed on the right [30]

2.2 Fast Gradient Sign Method

A more popular method in early works is the Fast Gradient Sign Method (FGSM) which was introduced by Goodfellow et al. [7]. This approach assumes linear behavior in high-dimensional spaces is sufficient to generate adversarial examples. The linear view of adversarial examples suggests a fast and efficient way of generating them as the perturbation is computed with a single step.

Formally, we define θ is the weights of the neural network, x is the input to the model, y is the target labels for x, and $J(\theta, x, y)$ is the cost function for the neural network. We will linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained perturbation of

$$\eta = \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \tag{3}$$

In the equation, ∇_x is the gradient of the cost function with respect to the input x, which can be computed efficiently using backpropagation. The magnitude of the perturbation η is controlled by the parameter ϵ. The perturbation with a lager ϵ value would be more likely to fool the neural network, but will also be more discernible to human. Figure 4 demonstrates an adversarial sample generated with FGSM.

2.3 Iterative Gradient Sign Methods

FGSM is a simple one-step method to generate adversarial samples. One idea to extend and improve FGSM is modifying it to an iterative methods instead of one-step method. Basic Iterative Method (BIM) and Iterative Least-Likely Class Method (ILCM) are two different methods developed by Kurakin et al. [13] to iteratively generate adversarial samples.

Basic Iterative Method (BIM) is a straightforward extension of FGSM to apply it multiple times in small step size. The clip pixel values are checked after each step to make sure they are in ϵ-range to the original image:

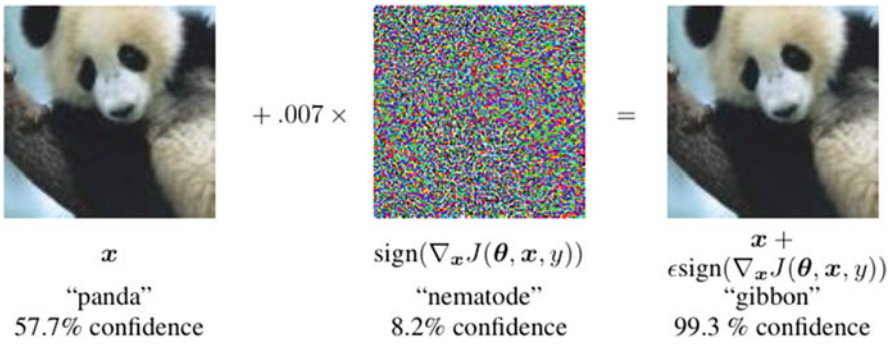


Fig. 4 A demonstration of fast adversarial example generation applied to GoogLeNet [28] on ImageNet [5]. By adding an imperceptibly small vector to the input, GoogLeNet's classification of the image is changed [7]

$$X_0^{adv} = X;\ X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \alpha sign(\nabla_X J(X_N^{adv}, y_{true}))\} \tag{4}$$

X_{N+1}^{adv} denotes the adversarial sample generated at the $N+1$ iteration. It is the $Clip$ pixel value of the X_N^{adv} plus the perturbation computed with FGSM. In the author's experiment, $\alpha = 1$ and $N = min(\epsilon + 4, 1.25\epsilon)$ which is sufficient to reach the edge of the ϵ max-norm ball within reasonable computational complexity.

Both FGSM and BIM are only trying to increase the cost of the correct class, but not care about which incorrect classes the model should mis-classify to. **Iterative Least-Likely Class Method** (ILCM) aims to generate adversarial samples that can lead the neural network to mis-classify the image X to the least-likely class y_{LL} according to the trained model:

$$y_{LL} = \underset{y}{\text{argmin}}\{p(y|X)\} \tag{5}$$

According to this definition, the least-likely class would be the class highly different from the true class. To generate the adversarial image to be classified as y_{LL}, we add perturbation to the original image iteratively with the direction of $sign(\nabla_X \log p(y_{LL}|X))$ which equals to $sign(\nabla_X J(X, y_{LL}))$ for neural networks with cross-entropy loss. So the iterative steps can be represented as following:

$$X_0^{adv} = X;\ X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} - \alpha sign(\nabla_X J(X_N^{adv}, y_{true}))\} \tag{6}$$

Again, α is used to control the step size and equals to 1 in the author's experiment.

The performance of FGSM, BIM and ILCM are compared on 50,000 validation samples from the ImageNet dataset using the Inception v3 model [29]. Adversarial samples have been generated with these three methods with different ϵ values. Top-1 and Top-5 classification accuracy on clean image and adversarial images are compared in Fig. 5. As the figure shown, FGSM (fast adv.) can decrease both

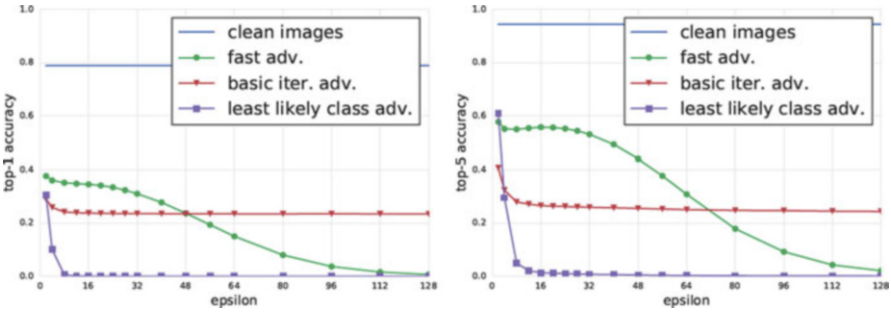


Fig. 5 Top-1 and top-5 accuracy of Inception v3 under attack by different adversarial methods and ϵ values. The accuracy was computed on 50,000 validation images from the ImageNet dataset [13]

top-1 and top-5 accuracy by about 40% with a relatively small ϵ value, but cannot continue to decrease the accuracy until $\epsilon \geq 32$. BIM (basic iter. adv.) can generate better adversarial samples with small ϵ, but unable to improve when increasing ϵ. By contrast, ILCM (least likely class adv.) shows much better performance. It makes the neural network mis-classify most of images with very small ϵ. $\epsilon \leq 16$ is used in the author's experiments as it produces small perturbation but is enough to lead most images mis-classified.

2.4 Jacobian-Based Saliency Map Attack

Jacobian-based Saliency Map Attack (JSMA) [24] generates adversarial samples through building adversarial saliency maps. In this method, given an original sample X which is classified as Y by the neural network F, we want to generate an adversarial sample X^* very similar to X, but misclassified as $Y^* \neq Y$ by F:

$$\underset{\delta_X}{\operatorname{argmin}} ||\delta_X|| \;\; s.t. \;\; F(X + \delta_X) = Y^* \tag{7}$$

where δ_X is the small perturbation added to X to generate adversarial sample X^*, and $|| \cdot ||$ is an appropriate norm.

To generate the adversarial sample X^*, JSMA firstly compute the forward derivative for the original sample X, which is the Jacobian matrix of the neural network F learned during training process:

$$\nabla F(X) = \frac{\partial F(X)}{\partial X} = \left[\frac{\partial F_j(X)}{\partial X_i}\right]_{i \in 1...M, j \in 1...N} \tag{8}$$

The second step of the algorithm is constructing an adversarial saliency map indicating which feature we should perturb to most efficiently generate the adversarial sample. Because the neural network predict a class label by $label(X) = \arg\max_j F_j(X)$, we want to increase the probability of $F_t(X)$ for a target class t and decrease the probability $F_j(X)$ for all other classes $j \neq t$ to make $F_t(x)$ as the largest probability. This process is done by increasing the input features indicated by the saliency map $S(X, t)$:

$$S(X,t)[i] = \begin{cases} 0 \; if \; \frac{\partial F_t(X)}{\partial X_t} < 0 \; or \; \sum_{j \neq t} \frac{\partial F_t(X)}{\partial X_t} > 0 \\ (\frac{\partial F_t(X)}{\partial X_t}) | \sum_{j \neq t} \frac{\partial F_t(X)}{\partial X_t} | \; otherwise \end{cases} \tag{9}$$

The identified feature will be perturbed by a small amount to generate a new temporary sample X^*. The previous steps will be iterated with updated X^* many times until the cumulated perturbation reach the maximum allowed distortion.

The JSMA method has been validated using a well studied neural network for hand-written digit recognition, namely LeNet [16]. The author claims 97.10% success rate for mis-classifying to any target class on any input sample with only 4.02% perturbation of input features in average using the MNIST dataset. Figure 6 shows 10 original samples on the diagonal and 90 adversarial samples generated for the original samples for different target classes.

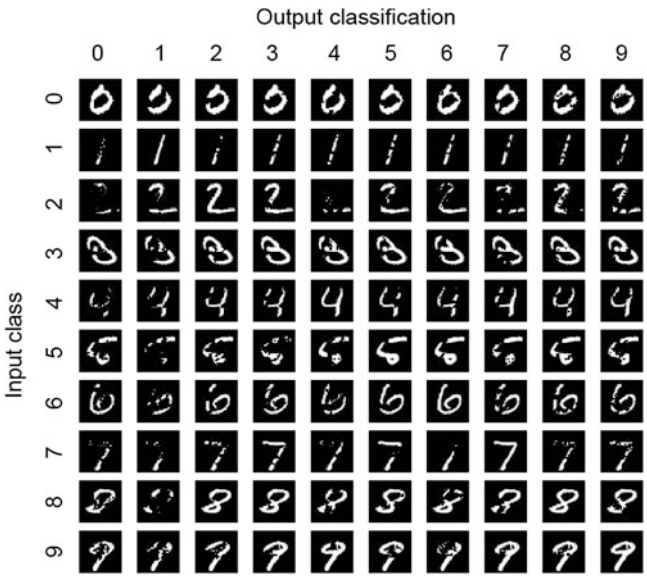


Fig. 6 Original samples and generated Adversarial samples on the MNIST dataset. The rows indicate original input classes and columns indicate target classes [24]

2.5 *DeepFool*

DeepFool [21] computes the minimal norm adversarial perturbation for the original image based on an iterative linearization of the classifier. Here we introduce the core DeepFool algorithm for binary classifiers. It can be easily extended to multiclass classifiers as they could be treated as consisting with several binary classifiers with the one-vs-all scheme. We define $f : \mathbb{R}_n \rightarrow \mathbb{R}$ is the image classification function, and $\hat{k}(x) = sign(f(x))$ is the classified label for the given image x. Besides, we denote the hyperplane $\mathscr{F} = \{x : f(x) = 0\}$ at zero of f. For a point x_0, the minimal perturbation makes the least movement of x_0 towards the hyperplane $\mathscr{F}$ to change a classifier's decision. It is given by the closed-form formula:

$$r_*(x_0) := \text{argmin}\, ||r||_2 \;\; s.t. \;\; sign(f(x_0 + r)) \neq sign(f(x_0)) \tag{10}$$

The author also call the $\Delta(x_0; f) = r_*(x_0)$ as the robustness of the function f at point x_0. At each iteration i, the algorithm estimates the robustness of f at x_i by linearizing f around the point x_i, which is computed as

$$\underset{r_i}{\text{argmin}}\, ||r_i||_2 \;\; s.t. \;\; f(x_i) + \nabla f(x_i)^T r_i = 0 \tag{11}$$

For the linearized f, the minimal perturbation is the orthogonal projection of x_i onto $\mathscr{F}$ as Fig. 7 illustrated. It is given by the closed-form formula:

$$r_*(x_i) = -\frac{f(x_i)}{||\nabla f(x_i)||_2^2} \nabla f(x_i) \tag{12}$$

The algorithm terminates the iteration when the sign of the classifier is changed.

The algorithm have been evaluated on eight state-of-the-art classifiers such as LeNet, GoogLeNet, CaffeNet [10] etc. with three datasets: MNIST, CIFAR-10 [11] and ILSVRC2012 [26]. The results show that Deepfool can accurately and efficiently generate adversarial samples. Furthermore, the authors claim it can generate smaller perturbations compared to FGSM with same attacking performance.

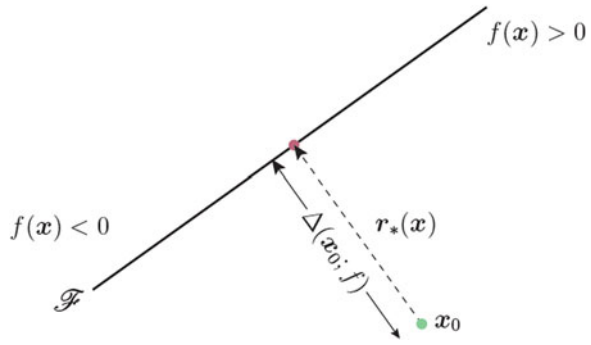


Fig. 7 Computation of the minimal perturbation for linearized binary classier [21]

2.6 Carlini & Wagner Attack

Carlini & Wagner Attack [3] aims to generate adversarial samples against the neural networks with defensive distillation. **Defensive distillation** is a recent technique applied to defend adversarial attacks on neural networks [25], which will be introduced in detail in next section of this chapter. The authors demonstrate that the defensive distillation model fails on Carlini & Wagner Attack while defeats other attacks.

The L_p norms distance metric is used for measuring the similarity of a given input sample x and an adversarial sample of it x'. It is written as $||x - x'||_p$, where the $p - norm$, $||\cdot||_p$ is

$$||v||_p = (\sum_{i=1}^{n} |v_i|^p)^{\frac{1}{p}} \tag{13}$$

Three widely used L_p norms distance metrics have been applied separately in the Carlini & Wagner Attack:

1. L_0 **distance**: measures the coordinates i such that $x_i \neq x_i'$. It checks the number of the pixels have been perturbed for an image.
2. L_2 **distance**: measures the Euclidean distance between x and x'
3. L_∞ **distance**: measures the maximum change to any of the coordinates.

To formally define the approach of Carlini & Wagner Attack, given a trained classifier C, an image x, and the target label t, we want to find some small perturbation δ

$$\underset{\delta}{\text{argmin}} ||\delta||_p \;\; s.t. \;\; C(X + \delta) = t, \;\; x + \delta \in [0, 1]^n \tag{14}$$

To solve the optimization problem, we choose a suitable objective function f such that $C(x + \delta) = t$ only if $f(x + \delta) \leq 0$. Beside, we use a constant $c > 0$ to control the sensitivity of the objective function. The equation above could be converted to

$$\underset{\delta}{\text{argmin}} ||\delta||_p + c \cdot f(x + \delta) \;\; s.t. \;\; x + \delta \in [0, 1]^n \tag{15}$$

There are many options for the function f, the one the authors used is

$$f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t, -k) \tag{16}$$

where Z is the logits function and k is used to control the confidence of the misclassification.

In the authors' experiments, adversarial samples are generated with the L_0, L_2, and L_∞ distance metrics separately. These types of attack have been evaluated

on ImageNet, CIFAR and MNSIT datasets. The results show that L_2 has the best performance which is able to find ten times less distortion than other methods, e.g. FGSM, JSMA, Deepfool, etc., while the L_0 can generate two times less distortion samples and L_∞ have similar performance as other methods.

The authors also exploited one important feature of adversarial sample: **transferability** which is one of the core concepts for black-box attack. They have shown that adversarial samples generated for one model can be used to attack another model when the value of k is big enough. We will discuss this concept and black-box attack techniques in the following sections.

2.7 Transferability Based Approach

Different from white-box attack which requires the knowledge of neural network architecture and parameters, black-box attack has no access to them. Many work on black-box attack are the transferability based approaches. Transferability refers to the property that adversarial samples generated for one model may also be misclassified by another model. Thus, in the black-box attack setting that target model architecture is unavailable, we can construct a substitute of the target model, then generate adversarial samples based on the substitute model, and attack the target model with the generated samples.

The first black-box attack against neural networks is demonstrated by Papernot et al. [23]. This work shows that adversarial attacks can be done without knowing the architecture and parameter of the model. The attacking approach can be divided into two stages: substitute model training and adversarial sample crafting.

The first stage of the work is to train a substitute model F for the target model oracle O. To build the substitute model, we need to firstly choose the architecture of it. The choice depends on the objective of model, e.g. image classification, text processing, etc. We want to choose the model suitable for the task, such as choosing a Convolutional Neural Network(CNN) for classifying images. The authors claim that the type, number and size of neural network layers have relatively small impact on the task.

To generate training dataset for the substitute model, we firstly collect a small set S_0 of representative inputs, and query target model O to get the label $\tilde{O}(\overrightarrow{x})$ for each input $\overrightarrow{x} \in S_0$. After training an initial substitute model F, we can iteratively augment the dataset by computing the sign of the F's Jacobian matrix dimension corresponding to the label of $\overrightarrow{x}$. The augmented dataset S_ρ at iteration ρ is

$$S_{\rho+1} = \overrightarrow{x} + \lambda \cdot sgn(J_F[\tilde{O}(\overrightarrow{x})]) : \overrightarrow{x} \in S_\rho \cup S_\rho \tag{17}$$

where λ is a parameter to control the step size for augmentation. Figure 8 illustrates the training process with augmentation of dataset.

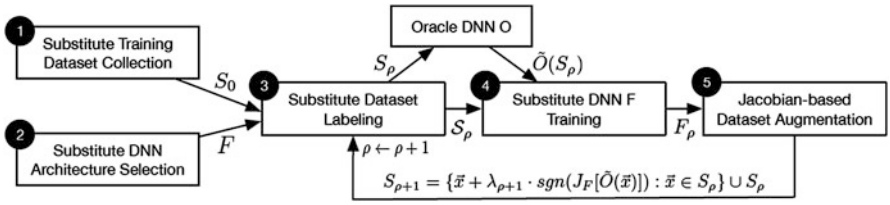


Fig. 8 Process of training substitute DNN and dataset augmentation [23]

Once we have a substitute neural network, we can generate adversarial samples based on it with a suitable white-box attack technique. The authors applied FGSM and JSMA approach to generate adversarial samples in their work.

This black-box attack approach has been evaluated on several real world classifiers held by MetaMind, Amazon, Google etc., and archived promising success rates. We will discuss these work in Sect. 4.1 as practical attacking examples in real world.

2.8 Houdini

Houdini [4] is an approach proposed for fooling gradient-based models by generating adversarial samples tailored for the task loss. Formally, given a neural network g_θ with parameter θ and the loss function is $l(\cdot)$, the adversarial sample $\tilde{x}$ for an input (x, y) can be defined as

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p \leq \epsilon}{\text{argmax}} \; l(y_\theta(\tilde{x}), y) \tag{18}$$

where $||\cdot||_p$ is a p-norm and ϵ controls the strength of the attack. To be able to optimize the task loss, a surrogate loss $\bar{l}_H$ namely Houdini has been proposed:

$$\bar{l}_H(\theta, x, y) = \mathbb{P}_{\gamma \sim \mathcal{N}(0,1)}[g_\theta(x, y) - g_\theta(x, \hat{y}) < \gamma] \cdot l(\hat{y}, y) \tag{19}$$

As the equation shows, Houdini is a product with two parts. The first part is a stochastic margin that calculate the probability that difference between the score of actual label and predicted label is less than γ. This part indicates the confidence of the prediction. The second part is the original loss we want to maximize. When the probability in the first part goes to 1, the surrogate loss converges to the original task loss. This property ensures Houdini is a good surrogate of the task loss for generating adversarial samples. The derivation process of computation of the gradient of Houdini is omitted here due to its complexity, but we eventually can compute it with the following formula:

$$\Delta_g[\bar{l}_H(\hat{y}, y)] = \begin{cases} -C \cdot e^{-|\delta g(y,\hat{y})|^2/2} l(y, \hat{y}), \ if \ g = g_\theta(x, y) \\ C \cdot e^{-|\delta g(y,\hat{y})|^2/2} l(y, \hat{y}), \ if \ g = g_\theta(x, \hat{y}) \\ 0, \ otherwise \end{cases} \tag{20}$$

where $C = \frac{1}{\sqrt{2\pi}}$, and $\delta g = g_\theta(x, y) - g_\theta(x, \hat{y})$.

The performance of Houdini has been evaluated in different tasks: human pose estimation, semantic segmentation and speech recognition. We will introduce the experiment of applying Houdini to attack speech recognition systems in Sect. 4.2.

3 Methods for Defending Adversarial Attacks

Along with the development of adversarial attack techniques, various models also have been developed for defending these attacks and archived good results. These work can be roughly grouped into three categories:

1. **Modifying training samples**: These approaches incorporate knowledge of adversarial attacks into the training dataset. Majority of these methods add adversarial samples generated by some attacking methods into the legitimate training set to train a robust model.
2. **Modifying model structure or training process**: These approaches train robust models against adversarial attacks by modifying certain parts of neural network, such as number of layers, format of output layer, loss functions, etc., or altering the normal steps for training neural network, such as adding pre-steps, adding post-steps, repeating normal process multiple times, etc.
3. **Combining with other models**: These approaches train one or multiple other models apart from the original model to assist the neural network for classification in adversarial environment.

Some defending methods might combine multiple approaches. For instance, the Game Theory based approach which will be introduced in this section both adds adversarial training samples and modifies the training process.

In this section, we introduce five representative methods for defending adversarial attacks as listed in Table 2.

Table 2 List of defending methods introduced in this section and their categories

	Category
Adversarial training [7, 21]	Adding adversarial training samples
Defensive distillation [25]	Modifying model structure or training process
Game theory based approach [32]	Adding adversarial training samples & Modifying model structure or training process
MagNet [20]	Combining with other models
Defense-GAN [27]	Combining with other models

3.1 Adversarial Training

One popular approach to defend adversarial attacks is the so called adversarial training. In this approach, adversarial examples generated from one or multiple attack models are added into the training set to increase the robustness of the trained model. Actually, many work on adversarial attacks are both generating adversarial samples and applying the generated samples to train models more robust to the attacks, such as [7, 21] etc.

Here we firstly discuss an adversarial training approach done by Goodfellow et al. [7]. They applied FGSM to generate the adversarial samples as we discussed in Sect. 2.2. With the notions used in Sect. 2.2, the adversarial objective function is

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y))) \tag{21}$$

where α is used to control the proportion of adversarial samples added to the original training set and was set to 0.5 in the authors' experiments. As the reported by the authors, the error rate on adversarial examples generated by FGSM significantly dropped from 89.4% to 17.9% with adversarial training. The adversarial training procedure can be viewed as a special type of active learning which is able to request labels on new points. For adversarial learning, the labels of new points are generated by copying from neighbor points rather than given by human.

As another example, the adversarial training approach done by Moosavi-Dezfooli et at. [21] trains neural networks on adversarial samples generated by Deepfool method which is also proposed by themselves. This approach includes five additional epochs to fine-tune the network with 50% decreased learning rate on perturbed training set. The performance of this adversarial training approach is evaluated with the model robustness metrics. Formally, the robustness $\hat{\rho}_{adv}$ of a model f on all input x in the test set $\mathbb{T}$ is defined as

$$\hat{\rho}_{adv}(f) = \frac{1}{|\mathbb{T}|} \sum_{i \in \mathbb{T}} \frac{||\hat{r}(x)||_2}{||x||_2} \tag{22}$$

where $\hat{r}$ is the perturbation obtained using Deepfool. Figure 9 shows the evolution of the model robustness for LeNet on CIFAR-10 dataset through the epochs. The performance of using FGSM generated adversarial samples, Deepfool generated adversarial samples and clean samples for fine-tuning are compared in the figure. The fine-tuned model on Deepfool generated adversarial samples performs best in term of robustness and is significantly improved compare to clean samples. Interestingly, the adversarial samples generated by GFSM lead to weaker robustness in later epochs. The authors suspect that is due to the perturbations generated by FGSM is much large than Deepfool.

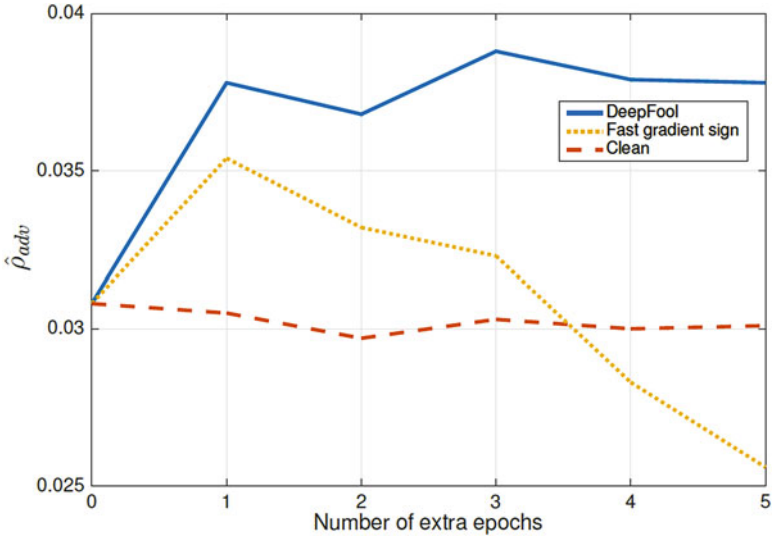


Fig. 9 Evolution of the model robustness for LeNet on CIFAR-10 dataset through the epochs [21]

3.2 Defensive Distillation

Some approaches modify the training process of neural network to make the model robust to adversarial attacks. One important approach among them is Defensive Distillation [25] which is inspired by the Distillation technique introduced by Hinton et al. [9].

Distillation is a technique proposed for transferring knowledge from a relatively complex network to a simpler network. This transferability is based on the fact that knowledge of a trained neural network is encoded in not only weight parameters but also probability vectors. Therefore, we can extract class knowledge from probability vectors and transfer it to another neural network. Distillation was initially developed for reducing complex neural networks' architecture so they can be deployed on devices with limited resources such as smart phones.

The intuition of applying Distillation as defensive technique is that transferring knowledge from a larger neural network to a smaller neural network can improve the generalization capability of the network while maintain reasonable accuracy. Therefore the distilled network will be more robust to perturbations.

In [25]'s approach, the authors explored the Defensive distillation technique on the neural networks with softmax layer as output layer. Formally, we define the input X from the training set χ with hard label $Y(X)$ in one-hot encoding, then we train the model through the following steps:

1. Training a deep neural network F with a softmax output layer at temperature T with the given training set. The probability vector $F(X) = p(\cdot|X, \theta_F)$, where θ_F is the parameter of F.

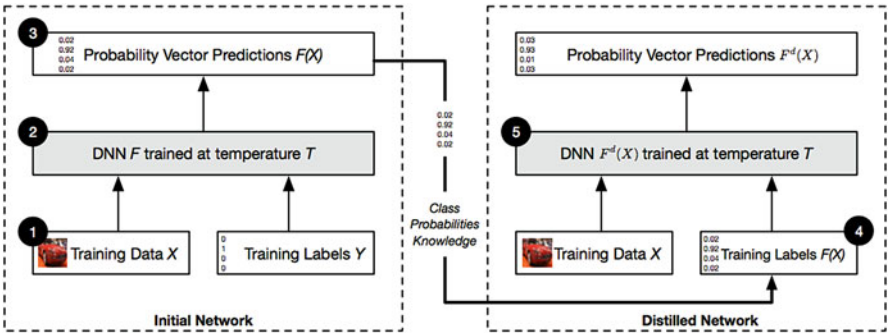


Fig. 10 The process of tanning a distilled neural network [25]

2. Generating a new training set $\{(X, F(X)) : X \in \chi\}$, in which we use the soft-target $F(X)$ instead of the original hard label $Y(X)$.
3. Using the new training set to train the distilled network F^d with same architecture and temperature as F.

The core part of this process is using the soft-target $F(X)$ instead of hard label $Y(X)$ which enable us to utilize the knowledge in the probability vectors. Figure 10 illustrates the whole process of training distilled neural network.

The defensive distillation technique has been evaluated on MNIST and CIFAR-10 dataset with DNN neural networks. The experiment results show that the approach is able to defend many popular attacks such as L-BFGS, FGSM. However, as we discussed in Sect. 2.6, defensive distillation failed on Carlini & Wagner Attack.

3.3 Game Theory Based Approach

Game Theory is a strategy framework to be applied by competing players to optimize the payoff for all parties. Game Theory has been applied for developing robust models against adversarial learnings in recent years [18, 19, 31, 32]. Here we introduce a Game Theory based approach both adds adversarial training samples and modifies training process to train a robust classifier against adversarial attacks [32]. In this approach, the process of training a classier and attacking the trained classifier are formulated as a Non-zero-sum Game between data miner and adversary.

In a binary classification setting, for a given training dataset $(x_i, y_i)_{i=1}^n$, where x_i is the input and $y_i \in \{-1, 1\}$ is the label, the data miner's objective is finding optimized weight w^* of the classifier

$$w^* = \underset{w}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w, x_i) + \lambda_w \|w\|_p \tag{23}$$

Table 3 Loss functions used for the two players [32]

	$\ell(y_i, w, x_i)$
Square	$\frac{1}{2}\|y_i - w^T x_i\|_2$
Logistic	$\log(1 + \exp(-y_i w^T x_i))$
Hinge	$(1 - y_i(w^T x_i))_+$

where l is a suitable (convex) loss function, λ_w is a regularization parameter and $||\cdot||_p$ is L_p norm. In the author's experiments, three different loss functions in Table 3 are applied with L_1 and L_2 norm separately.

To formalize the objective of adversary, we denote all samples with positive label as $(x_i, 1)_{i=1}^{npos}$, and all samples with negative label as $(x_i, -1)_{i=npos+1}^{n}$. The adversary aims to compute a small perturbation α^*:

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \frac{1}{npos} \sum_{i=1}^{npos} \ell(-1, w, x_i + \alpha) + \lambda_\alpha \|\alpha\|_p \tag{24}$$

Once the α^* is generated by the adversary, the data miner will find the new weight w^* with the updated training set $(x^*{}_i, y_i)_{i=1}^{n}$, where x^* is the updated input with perturbation α^*. These two steps are repeated sequentially until reach the Nash Equilibrium state of the game or the accumulated perturbation reaches a predefined maximum budget MB. The detailed steps are described in Algorithm 1.

Algorithm 1 Non-zero-sum Game [32]

Input: Training data $D = \{x_i, y_i\}_{i=1}^{n}$, maximum budget MB, λ_w, λ_α and Norm p
Output: w and α

1: // Build the initial classifier using original training data:
2: $w = \text{argmin}_w \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w, x_i) + \lambda_w \|w\|_p$
3: $Cost \leftarrow 0, \alpha_{Sum} \leftarrow 0$
4: **while** $Cost <= MB$ **do**
5: // Step 1: Adversary attacks
6: // Learn α by assigning negative label to positive samples.
7: $\alpha = \text{argmin}_\alpha \frac{1}{npos} \sum_{i=1}^{npos} \ell(-1, w, (x_i + \alpha)) + \lambda_\alpha \|\alpha\|_p$
8: for positive data : $x_i^{*npos} = x_i^{npos} + \alpha$
9: // Step 2: Data miner responds
10: $w = \text{argmin}_w \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w, x_i^*) + \lambda_w \|w\|_p$
11: // Calculate accumulated cost
12: $Cost += \|\alpha\|_1$
13: **end while**
14: return w

Apart from the experimenting on popular digit image dataset MNIST and USPS, the authors also performed the experiment for detecting spam email. The experiment results show that their game-theoretic classifier deteriorate much slower on new types of junk spam in the future comparing to regular classifiers.

3.4 MagNet

The MagNet framework [20] is proposed to defend neural network against adversarial attacks by adding one or more detector network and a reformer network. The mis-classification problems are divided into two scenarios in this framework:

1. The adversarial example is far from the boundary of the normal sample group. In this scenario, the classifier mis-classifies the input only because it has to predict one label anyway even with very low probability.
2. The adversarial example is close to the boundary of the normal sample group. The classifier mis-classifies the input to the class that is close to adversarial samples in this case.

For any given input x, MagNet will firstly check if the input is for the first scenario by checking the difference of input sample to normal samples. If the input is detected as adversarial sample by all detectors, we are in the first scenario and finish the process. Otherwise, we are in the second scenario, and will apply a reformer network to reform the input to x' before passing it to the classifier. Figure 11 illustrates the MagNet work flow for these two scenarios.

There are two types of detectors applied in this approach. The first type of detector is based on reconstruction error. It uses an AutoEncoder as the detector and use reconstruction error to estimate the distance between the input and normal sample group. **AutoEncoder**(AE) is a type of neural network that output values to be similar to input. It is able to learn simpler hidden representation of the input for regularization purpose. An AutoEncoder consists of two parts: an encoder $e : \mathbb{S} \rightarrow \mathbb{H}$ and a decoder $d : \mathbb{H} \rightarrow \mathbb{S}$, where $\mathbb{S}$ is the input space and $\mathbb{H}$ is the hidden representation space. The reconstruction error on a input x with AutoEncoder is

$$E(x) = ||x - AE(x)||_p \tag{25}$$

where $AE(x)$ is the output of the AutoEncoder and $||\cdot||_p$ is $L_p - norm$.

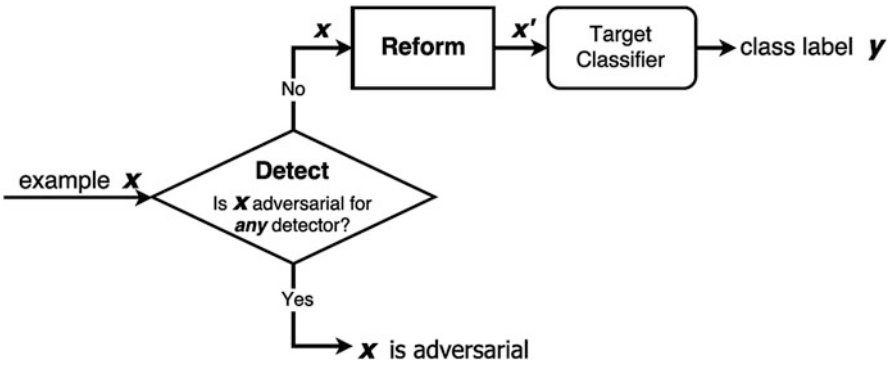


Fig. 11 The illustration of the MagNet work flow [20]

Another type of detector is based on the probability divergence which is for detecting adversarial samples with small reconstruction error. We define $f(x)$ as the output of a neural network's last layer which using a softmax function. We want to compute the divergence between $f(x)$ and $f(AE(x))$ using Jensen-Shannon Divergence(JSD):

$$\begin{gathered} JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M), \\ where \;\; D_{KL} = (P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}, \\ and \;\; M = \frac{1}{2}(P + Q) \end{gathered} \tag{26}$$

There are also two types of reformers. The first type is a naive noise-based reformer which add random Gaussian noise to the input. The other type of reformer is an AutoEncoder-base reformer. The AutoEncoder trained to minimize the reconstruction error when developing detectors is used here again. It should generate almost same output for normal samples, while output approximation of adversarial samples which are similar to neighbor normal samples.

The MagNet framework has been evaluated on protecting neural networks against FGSM, iterative gradient sign method, DeepFool and C&W attack on MNIST dataset and CIFAR-10 dataset. MagNet achieved above 75% accuracy on all the attacks and above 90% accuracy for more than half attacks, which shows the effectiveness to different types of attacks.

3.5 Defense-GAN

As another example framework combining multiple models for defending adversarial attacks, Defense-GAN [27] leverages the capability the representative power of a popular generative model Generative Adversarial Networks to defend adversarial attacks.

Generative Adversarial Networks (GANs) is a deep learning framework for creating synthetic data similar to given input data [6]. The framework consists of two networks: a generator G and a discriminator D. Intuitively, The generator generates data similar to the input data, while the discriminator determines whether a given data is from the input dataset or is the output of the generator. Formally, we denote generator G generates output $G(z)$ based on a random input vector z following a distribution $p_z(z)$. On the other hand, $D(x)$ is the probability that x from the input data rather than generated by G. G and D are trained simultaneously in an adversarial setting as we want to train D to maximize the probability of assigning correct label but train G to generate $G(z)$ to fool D. G and D are trained with a combined value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \dim p_data(x)}[\log D(x)] + \mathbb{E}_{z \dim p_z(z)}[\log(1 - D(G(z)))] \tag{27}$$

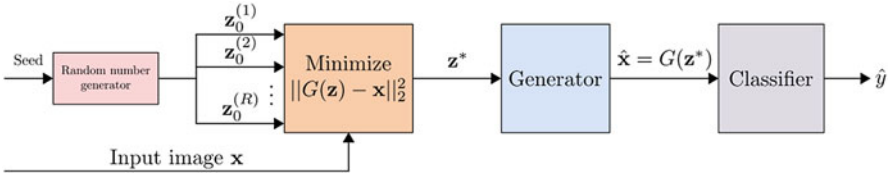


Fig. 12 Overview of the Defense-GAN algorithm [27]

Defense-GAN is developed based on a variation of GANs namely Wasserstein GANs (WGANs) [1]. When we have an image x to be classified by the model, we will firstly generate a substitute image $G(z)$ with the trained generator of WGANs with the minimum error

$$\min_z \|G(z) - x\|_2^2 \tag{28}$$

The generated image $G(z)$ is then fed into the classier instead of using image x. The intuition of this process is that the image generated by the generator G will contain reduced adversarial noise of image x, and thus can be classified more precisely. Figure 12 illustrates the algorithm of Defense-GAN framework.

Defense-GAN has been tested on both white-box attacks and black-box attacks, such as FGSM, C&W attacks, etc., and compared the performance with adversarial training and MagNet. The results show that Defense-GAN is a feasible defense framework. However, as noted by the authors, the performance of model heavily relies on the proper training of GANs which is often a challenging task.

4 Adversarial Learning Applications for Cyber Security

As more and more adversarial learning techniques, especially black-box attack techniques well developed, many adversarial learning techniques have been applied to deep learning frameworks in real world applications and cause cyber security issues. Here we introduce several recent work that attempt to attack commercial products and web services in real world, and achieved high success rate.

4.1 Attack Commercial Web Services

A practical black-box attack to commercial web services hosted on MetaMind, Amazon and Google have been done by Papernot et al. [23].

The transferability based attack which is discussed in Sect. 2.7 has been applied in this approach. The authors firstly upload MNIST dataset with 50,000 samples to

train a DNN classifier through the web service at https://metamind.io/., and trained a classifier with 94.97% accuracy after 36 h training. To attack the trained classifier, the authors prepared a substitute training set which augmented from 150 samples in original training set and 100 new samples. Then they trained two substitute DNNs which achieve 81.2% and 67% accuracy separately. Adversarial samples are generated on the substitute DNN model by applying FGSM attack with different variations of ϵ values. With $\epsilon = 0.3$, the mis-classification rate reach 84.24% and 78.72% separately for adversarial samples generated based on two substitute models. As a result, the model trained through MetaMind web service is fooled successfully.

They also attacked the models trained on Amazon Machine Learning API at https://aws.amazon.com/machine-learning and Google's Cloud Prediction API at https://cloud.google.com/prediction. The attacking approaches are omitted here as they are similar to the process of attacking MetaMind. The best mis-classification rates they achieved are 96.78% for DNN trained on Amazon web service and 97.17% for DNN trained on Google web service. These experiment results indicate that many state-of-the-art machine learning APIs in commercial web services are not robust against adversarial attacks.

4.2 Attack Automatic Speech Recognition System

An approach of attacking Automatic Speech Recognition(ASR) system have been demonstrated by Cisse et al. [4]. ASR is a system consists of multiple components such as acoustic model, language model, pronunciation model, etc. to convert a speech segment input to a transcript as output. The ASR system used in this experiment is a neural network consists of two Convolutional layers, seven Bidirectional-LSTM layers and one fully connected layer. The input of the model is a raw spectrogram and output is a transcript for it.

The Houdini method which is introduced in Sect. 2.8 has been applied to attack the ASR and compared with Connectionist Temporal Classification (CTC) based approach which is specially developed for ASR system. In this approach, adversarial samples are generated for 220 speech segments from Librispeech dataset [22]. The attacking performance is evaluated with Word Error Rate(WER) and Character Error Rate(CER). The results indicate that Houdini attack with $\epsilon = 0.05$ can cause 2.3 times higher WER and 1.8 times higher CER compare to CTC based approach.

Furthermore, the authors have attacked Google Voice in black-box attack setting in the same way. Both clean and adversarial speech audio samples are played in front of an Android device with Google Voice. Figure 13 shows a few examples of the converted transcript from both original and adversarial samples by Google Voice. It can be seen that Google Voice output transcriptions from adversarial samples with much higher WER compare to those produced from original inputs.

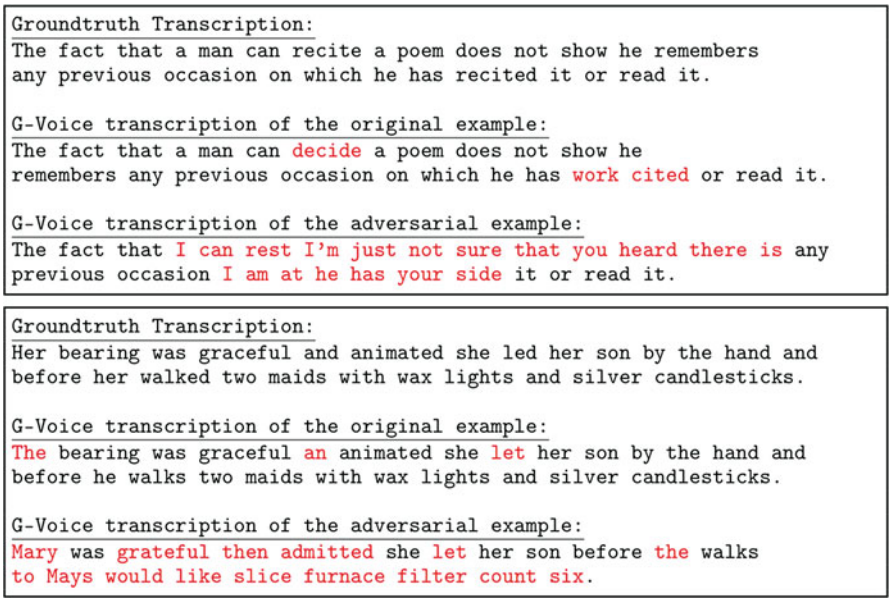


Fig. 13 Examples of transcriptions from Google Voice for original and adversarial input [4]

4.3 Attack Malware Classifier

The experiments on attacking malware classifier has been explored by Grosse et al. [8]. This work demonstrates that adversarial attacks can be threats on security-critical applications such as malware detector.

Several feed forward neural networks with different parameters (e.g. number of layers, number of neurons per layer, etc.) are firstly trained based on the DERBIN Android malware dataset [2]. The DERBIN dataset contains more than 120,000 android applications and 5,000 of them are malwares. The overall prediction accuracy of the classifiers are greater than 90% despite there are small variations across different models.

Adversarial samples are generated carefully using JSMA method which has been discussed in Sect. 2.4. As the functionality of the malware should not be affected when generating adversarial samples, the authors only add features that do not interfere with other features. Some examples of the features used in the process are permissions, intents, API calls, network address, etc. The misclassification rate is between 60% and 80% for adversarial samples on the classifiers with different parameters.

Apart from attacking malware classifier, the authors also exploring three mechanisms to defend the model against adversarial attacks. As a result, they found adversarial leaning with careful selection of parameters is the most effective

mechanism. Besides, the distillation also can improve the robustness of the model, although the improvement is relatively small. Finally, the feature reduction techniques are not suitable for defending adversarial attacks.

5 Conclusion

In this chapter, we introduced a wide range of popular and cutting-edge models in the adversarial learning domain. Both adversarial attacking and defending techniques for deep learning frameworks are explored in detail. Besides, recent work applying these techniques to real world applications and the relevance to cyber security has been discussed. We hope this chapter can benefit not only researchers in the adversarial learning and cyber security communities, but also readers who work in industry and attempt to apply these innovative techniques in real world problems.

References

1. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan. CoRR, abs/1701.07875
2. Arp D, Spreitzenbarth M, Gascon H, Rieck K (2014) Drebin: effective and explainable detection of android malware in your pocket. In: Proceedings of the 21th Annual Network and Distributed System Security Symposium (NDSS'14)
3. Carlini N, Wagner DA (2016) Towards evaluating the robustness of neural networks. CoRR, abs/1608.04644
4. Cisse M, Adi Y, Neverova N, Keshet J (2017) Houdini: fooling deep structured visual and speech recognition models with adversarial examples. In: Neural Information Processing Systems (NIPS 2017)
5. Deng J, Dong W, Socher R, Li LJ, Li K, Li F (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp 248–255
6. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Neural Information Processing Systems (NIPS 2014)
7. Goodfellow I, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: International Conference on Learning Representations
8. Grosse K, Papernot N, Manoharan P, Backes M, McDaniel PD (2016) Adversarial perturbations against deep neural networks for Malware classification. CoRR, abs/1606.04435
9. Hinton G, Vinyals O, Dean J (2014) Distilling the knowledge in a neural network. In: Deep Learning and Representation Learning Workshop at NIPS 2014
10. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia, MM'14, New York. ACM, pp 675–678
11. Krizhevsky A (2009) Learning multiple layers of features from tiny images. Technical report, University of Toronto.
12. Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (NIPS 2012)
13. Kurakin A, Goodfellow IJ, Bengio S (2016) Adversarial examples in the physical world. CoRR, abs/1607.02533
14. Le QV (2013) Building high-level features using large scale unsupervised learning. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp 8595–8598

15. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436 EP –
16. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
17. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE, 86(11):2278–2324
18. Liu W, Chawla S (2009) A game theoretical model for adversarial learning. In: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops. IEEE Computer Society, pp 25–30
19. Liu W, Chawla S (2010) Mining adversarial patterns via regularized loss minimization. Mach Learn 81(1):69–83
20. Meng D, Chen H (2017) Magnet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS'17, New York. ACM, pp 135–147
21. Moosavi-Dezfooli S, Fawziand A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
22. Panayotov V, Chen G, Povey D, Khudanpur S (2015) Librispeech: an ASR corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 5206–5210
23. Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS'17, New York. ACM, pp 506–519
24. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A (2016) The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroSP), pp 372–387
25. Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, Swami A (2015) Distillation as a defense to adversarial perturbations against deep neural networks. CoRR, abs/1511.04508
26. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Li F (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis (IJCV) 115(3):211–252
27. Samangouei P, Kabkab M, Chellappa R (2018) Defense-gan: protecting classifiers against adversarial attacks using generative models. CoRR, abs/1805.06605
28. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1–9
29. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
30. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R (2013) Intriguing properties of neural networks. CoRR, abs/1312.6199
31. Wang F, Liu W, Chawla S (2014) On sparse feature attacks in adversarial learning. In: Proceedings of 2014 IEEE International Conference on Data Mining (ICDM), pp 1013–1018
32. Yin Z, Wang F, Liu W, Chawla S (2018) Sparse feature attacks in adversarial learning. IEEE Trans Knowl Data Eng 30(6):1164–1177

Intelligent Situational-Awareness Architecture for Hybrid Emergency Power Systems in More Electric Aircraft

Gihan J. Mendis, Mohasinina Binte Kamal, and Jin Wei

Abstract In this chapter, we exploit the deep learning and adaptive neuro-fuzzy inference system (ANFIS) techniques to develop an intelligent situational awareness system for energy management systems of the emergency hybrid auxiliary power unit (APU) for more-electric aircrafts. Our proposed security control strategy consists of two main mechanisms: (1) deep learning-based attack detection scheme that explores the techniques of convolutional neural networks, deconvolutional neural networks, and recurrent neural networks and (2) adaptive neuro-fuzzy inference system (ANFIS)-based estimation method to calculate the true values of the compromised data. In this chapter, we also present some simulation results to illustrate the effectiveness of our proposed method in detecting the cyber-attacks, such as false data injection (FDI) attacks, and mitigating the impact of the cyber-attacks in the energy management for the hybrid APUs in more-electric aircrafts.

Keywords More-electrical aircraft · Hybrid emergency power systems · Situational-awareness architecture · Neuro-fuzzy inference system (ANFIS) · Deep learning · Cyber-attack detection

1 Introduction

In recent year, more-electric aircraft (MEA) attracts more and more attention because of its capability to leverage electrical energy to reduce the usage of hydraulic, pneumatic and mechanical energy. By using electrical energy, MEA provides secondary energy in aircraft with higher efficiency, less weight, lower power consumption, and better performance [1]. As the secondary generation systems of MEAs, auxiliary power units (APUs) are mainly used as starters to main power generators and to satisfy other secondary on-board power requirements.

G. J. Mendis (✉) · M. B. Kamal · J. Wei
The University of Akron, Akron, OH, USA
e-mail: ijm11@zips.uakron.edu; mk213@zips.uakron.edu; jwei1@uakron.edu

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_2

Additionally, APUs are critical for the effective energy management and emergency control of power systems. With the growing awareness regarding the importance of reducing pollution and the emission of greenhouse gases, power generation systems are utilizing more environment-friendly new electric power system technologies. As a critical step towards the MEA, the conventional emergency power system of the traditional aircraft can be potentially replaced by the hybrid ones using more efficient and environmental-friendly generating units such as fuel-cell, battery, and supercapacitor hybrid system [2, 3]. Hybrid auxiliary APUs are widely used to achieve this goal [4–7]. Despite the efficiency of MEAs with hybrid APUs, the increased system connectivity and the communication networks result in the higher vulnerability to cyber-attacks on flight operations including electric power system management [8]. In [9], Kumar and Xu identified cybersecurity risks in communication systems in aviation and proposed a vulnerability assessment framework for wireless threats in aviation cyber-physical systems. In [10], Huang and Dong discuss a cyber-attack where attackers monitor and modify measurements on an aero-engine. Considering the lack of outside assistant, systems that are capable of detecting and overcoming cyber-attacks are required for the protection of aircraft. This book chapter considers the vulnerability of hybrid APUs in MEA to cyber-attacks and proposes an attack-resilient energy management architecture.

Recently, some energy management strategies have been developed for fuel-cell hybrid power systems. In [11], Njoya et al. have developed an optimization based energy management strategy (EMS) called external energy maximization strategy (EEMS). This method was developed to maximize the energy of the battery and the supercapacitor while keeping the states of charge (SOCs) of the battery and supercapacitor within their operating limits, respectively. Heuristic rule-based state machine control method was employed for energy management of hybrid fuel-cell and battery power system [12]. In [13], classical PI controllers were implemented for controlling the main performance parameters of the hybrid system. A rule-based fuzzy logic control was proposed in [14]. The authors of [15] have used frequency decoupling methods such that the fuel cell provides low frequency demand, whereas the other energy sources deal with high frequency demand. A cost function optimization technology for minimizing fuel consumption was applied in [16]. Greenwell et al. in [17] have proposed a real-time model predictive method for energy management of the hybrid system. Considering the potential of cyber-attacks occurring on emergency hybrid APUs, it is important to develop an attack-resilient intelligent power management architecture to mitigate the potential attacks and ensure the efficient operation of the emergency power system even in the presence of the cyber-attacks. To address this challenge, we develop a situational-aware intelligent security control architecture for energy management systems that exploits deep learning-based prediction method and an adaptive neuro-fuzzy inference system (ANFIS) to evaluate the integrity of the critical measurement data of the hybrid APUs, such as the power output of the fuel cell.

A deep learning technique, including convolutional neural network (CNN), deconvolutional neural network (DCNN), and recurrent neural network (RNN), is employed in our proposed attack detection method. CNNs are well established

neural network models used in deep learning applications. One motivation for CNNs is from the neuron organizations of the mammals visual cortex [18]. Therefore, CNNs inherently perform well and are most commonly used for applications related to natural images [19–24]. However, CNNs have also used and shown exceptional performance in other pattern recognition applications such as spectrum analysis, human action recognition, stock trading trend detection, natural language processing etc. [25–32]. DCNNs essential perform the opposite operation for convolution which is to combine abstract features to generate meaningful patterns. DCNNs are used with CNN for applications that require regenerative capabilities such as saliency map generation, semantic segmentation [33–37]. RNNs are a class of neural networks developed for modeling of sequential dependencies between sequential data such as time-series data [38]. RNNs have been successfully applied to various sequence prediction and sequence labeling tasks in applications such as speech recognition, natural language processing, speech synthesis, handwriting recognition, human action recognition, visual tracking [39–44]. In [33], Kuen et al. have utilized RNN for improvement of a CNN and DCNN based generative network. In our work, we also exploit the combination of RNN with CNN-DCNN generative model to achieve better prediction. In addition, we exploit ANFIS technique and propose an estimation scheme for calculating of the true values of the compromised data. ANFIS is one of the most successful neuro-fuzzy systems developed, which applies neural learning rules to identify and tune the parameters and structure of a fuzzy inference system, based on the available data [45]. It is considered to be a universal estimator since it has the learning capability to approximate nonlinear functions by integrating both neural networks and fuzzy logic principles. Its inference system corresponds to a set of fuzzy if-then rules.

In this chapter, we evaluate our proposed methods targeting a hybrid APU of a MEA. The rest of the chapter is organized as follows. The next section describes the problem settings for our proposed deep learning and ANFIS based power management method. Section 3 details our proposed deep learning and ANFIS based intelligent security control architecture. The simulation results and the conclusions are presented in Sects. 4 and 5, respectively.

2 Problem Setting

To meet the increasing demand of electrical capacity in MEAs, engine based generators are being replaced with the highly efficient, green, continuously operating fuel-cell based auxiliary power units [46]. To account for their natural electrochemical reaction, fuel-cells have a slow response to supply the peak power [47]. Therefore Li-ion batteries and/or supercapacitors are accompanied to form the hybrid power and to supply the load during transient operating condition [11].

Since the hybrid APUs are controlled by the energy management strategies (EMSs), it is imperative to ensure the accuracy of the control signals made by the EMSs. To achieve this goal, one of the most important steps is to validate the

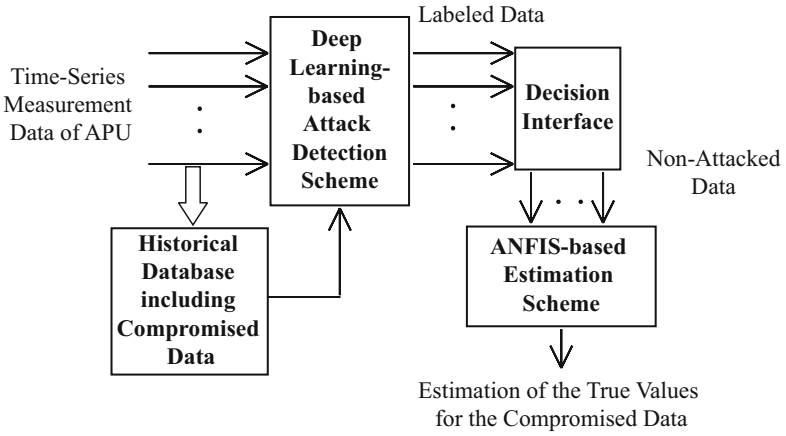


Fig. 1 Overview structure of our proposed security control architecture for hybrid emergency power system

integrity of the measurement data that is critical for obtaining an attack-resilient decision making. Any mismeasurement in calculation of the control output can make the whole system unstable causing life hazards. To address this problem, in our work we propose a situational-aware intelligent security control architecture for the energy management system of hybrid APUs of MEAs, which consists of three essential schemes: (1) WT-based feature visualization for the critical measurement data, (2) deep learning-based online attack-detection scheme, and (3) ANFIS-based estimation scheme. Figure 1 presents the high-level overview of our proposed intelligent security control architecture. As shown in Fig. 1, our proposed method, which employs WT and deep learning techniques, validates the critical time-series measurement data for the estimation scheme by training the measurements from the APUs and historical data as the reference for the validation. If there are abnormal patterns detected in any of the time-series measurements, an alarm label is set. Decision interface activate the ANFIS-based estimation scheme to estimate the true values of the compromised measurements leveraging the physical couplings with data of the other real-time measurements.

In the simulation sections, we evaluate our method considering attackers are interested in performing false data injection (FDI) attacks on APU reading data to compromise the EMS. The FDI attacks are simulated by specifying the following threat model: (1) the attackers can explore and modify any APU measurements; (2) attackers perform FDI attack by adding a random bias to the measured readings, and (3) because of the limited resource, the attacker is only able to modify one measured reading in a single time window. The authors would like to claim that the proposed method can be applied to detect and defend against the other types of cyber-attacks such as spoofing and denial of services (DoS) attacks.

3 Situation-Aware Intelligent Security Control Architecture for Energy Management Strategy

This section describes the details of the proposed intelligent security control architecture consisting of (1) deep learning-based online attack-detection scheme and (2) ANFIS-based estimation scheme.

3.1 Deep Learning-Based Cyber Attack Detection Scheme

The structure of our proposed deep learning-based cyber-attack detection scheme is illustrated in Fig. 2. As shown in Fig. 2, our strategy consists of a coupled structure of a deep convolutional neural network (CNN) and deep deconvolutional neural network (DCNN). Long short-term memory (LSTM) based recurrent neural network (RNN) is designed to process the abstract features achieved by deep CNN and to extract temporal features from the abstract CNN features.

The main building block of deep CNN is the convolution layer. The convolution layer performs convolution on inputs with a set of learnable kernels, obtained values are evaluated through an activation function to produce a set of output feature maps. Convolution output corresponding to jth kernel in lth layer y_j^l is calculated as follows:

$$\mathbf{y}_j^l = f\left(\sum_i \mathbf{x}_i^{l-1} * \mathbf{k}_{ij}^l + b_j^l\right) \tag{1}$$

where $\mathbf{x}_i^{l-1}$ is the ith input vector (ith kernel output from previous layer), k_{ij} is the corresponding learnable kernel mapping the ith input to the jth output, b_j^l is a bias vector corresponding to the jth output in the lth layer, and $f(\cdot)$ is an activation function.

During the training process, each learnable kernel's weight and bias values are adjusted to extract some specific features available in the training set. In the proposed scheme, we adopt rectifier activation ($f(x) = max(0, x)$). Convolution layers are followed by a sub-sampling layer to reduce computations and to gradually build up further spatial invariance. Our implementation uses 2×2 maximum pooling as sub-sampling. As shown in Fig. 2, the second main component of our machine learning method is the deep DCCN. Deconvolution layers are the essential components of the deep DCNN. Similar to convolution operation, decovolution layer outputs are calculated according to Eq. (2). A 2×2 unpooling operation is performed on the input of the deconvolution layer before performing the deconvolution. The number of kernels decreases for lower deconvolution layers. Therefore, the lowest deconvolution layer produces a single output vector with same dimension as the original data vector.

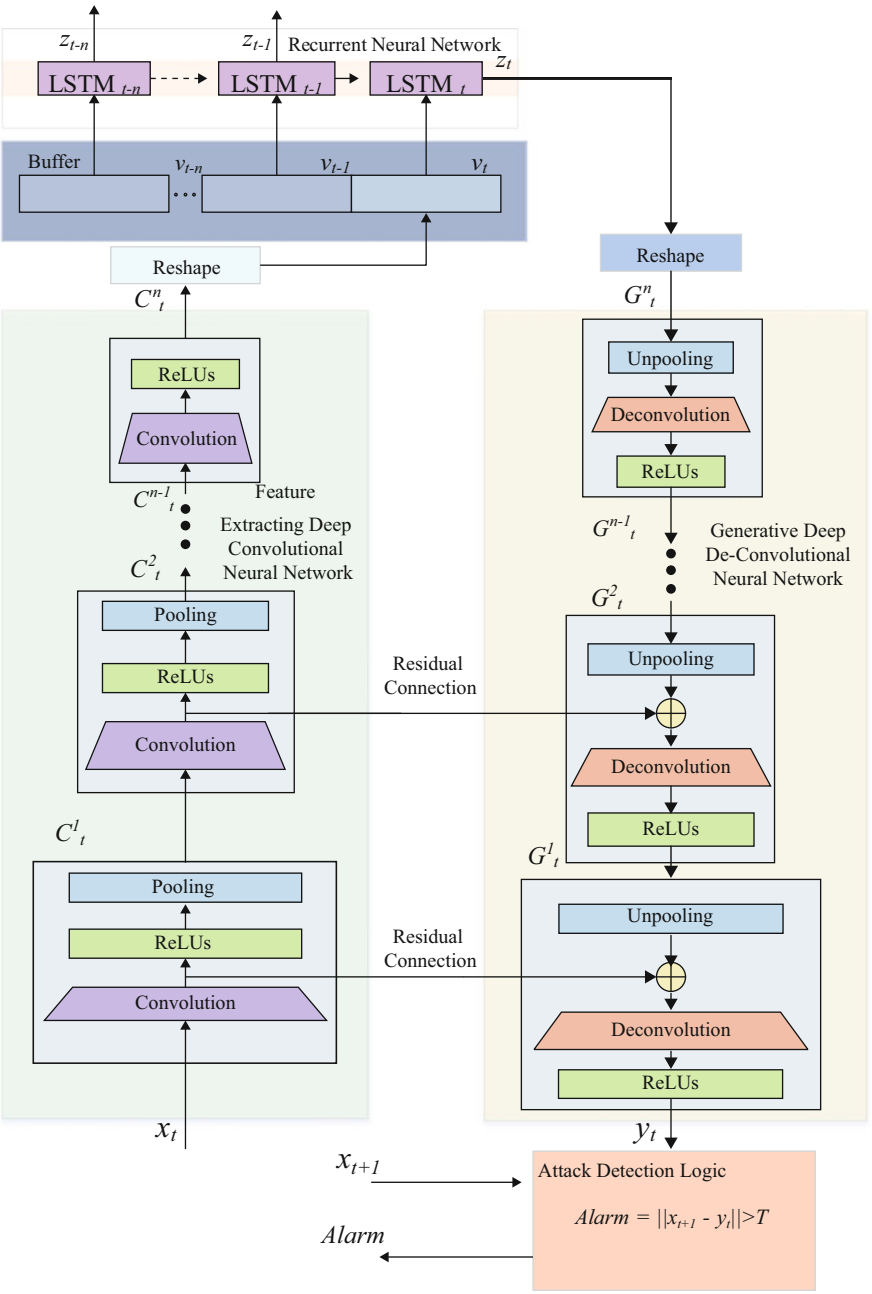


Fig. 2 Proposed architectures for our proposed deep learning-based cyber-attack detection scheme, where C^n_t is the output of the nth CNN at time t, G^n_t is the input for the nth generative DCNN, and ReLU block represent a layer of units with rectifier activation function

$$\mathbf{x}_i^l = f(\sum_j \mathbf{y}_j^{l+1} * \mathbf{k}_{ji}^l + c_i^l) \tag{2}$$

where $\mathbf{y}_h^{l+1}$ is the unpooled jth deconvolution kernel output from above layer, k_{ji} is the corresponding learnable deconvolution kernel mapping jth input to ith output, c_i^l is a bias vector corresponding to ith output in lth layer, and $f(\cdot)$ is the rectifier activation function.

As shown in Fig. 2, the output of the top CNN are reshaped and buffered to set as inputs to an LSTM based RNN network. The operation of the LSTM block is governed by Eq. (3) [48, 49]. Output of the final LSTM block is reshaped and provided as input for deep DCNN. RNN characterize the temporal features within the features learned by CNN and provide the DCNN with a more rich feature set with temporal features. In addition, the residual connections in Fig. 2 are used to improve the convergence of the overall architecture [50].

$$\begin{cases} i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i), \\ f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f), \\ c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c), \\ o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o), \\ m_t = o_t \odot h(c_t), \\ y_t = \varphi(W_{ym}m_t + b_y) \end{cases} \tag{3}$$

where the W terms denote weight matrices (e.g. W_{ix} is the matrix of weights from the input gate to the input), the b terms denote bias vectors, $\sigma(\cdot)$ is the logistic Sigmoid function, m is the cell output activation vector, i, f, o, and c are the input gate, forget gate, output gate and cell activation vectors, respectively, all of which have the same size as m, $\odot$ is the element-wise product of the vectors, $g(\cdot)$ and $h(\cdot)$ are the cell output and input activations that use hyperbolic tangent activation functions, φ is the softmax output activation function.

The architecture shown in Fig. 2 is trained with gradient descent with the loss function shown in Eq. (4), to minimize the mean square error (MSE) between x_t and y_t. Additionally, by setting $y_t = x_{t+1}$, we are able to predict the value of the data stream x at time $t+1$. In our work, x_t is a time window of measurement data from the hybrid APU and thus x_{t+1} is that for the next time window.

$$L(x_t, y_t) = \frac{1}{M}\sum_{\alpha=1}^{M}(x_t - y_t)^2 \tag{4}$$

where M is the length of x_t and y_t.

The functionality of different sections of the neural network is as following. The CNN characterizes the abstract features of the input data stream x_t, which visualize

temporal and spectral characteristics of small range time window of an APU reading through WT. The LSTM-based RNN extracts the long-range temporal features exhibited by the features obtained by CNNs. The generative DCNN achieves an estimation of the next time frame $y_t = \hat{x}_{t+1}$. Therefore, DCNN generates representations similar to the input data with the knowledge on both long and short-term temporal behavior of input data.

As illustrated in Fig. 1, after training with APU reading data, the proposed deep learning-based prediction method is employed to real-time attack detection on APU readings. APU reading data from the previous time windows are used to predict the current time window of APU readings. Then, the actual current time window of reading is observed and mean square error (MSE) between predicted and actual data is calculated. If the MSE is larger than a predefined threshold for an APU reading, an alarm label is set to that particular APU reading. Decision interface will detect the set alarm label and activate the ANFIS-based estimation scheme to estimate the compromised measurement.

3.2 Adaptive Neuro-Fuzzy Inference System (ANFIS)-Based Estimation Scheme

The adaptive neuro-fuzzy inference system (ANFIS) was developed to integrate the benefits of both the learning capability of the artificial neural network (ANN) and the inference ability of rule-based fuzzy logic control. It can be considered as a superset of all kinds of feed-forward neural network with supervised learning capability. ANFIS technique exploits the hybrid learning procedure to form the input-output relationship based on the human knowledge and input-output data. This technique has been proved to be effective in modeling the nonlinear systems, identifying the nonlinear parameters in online control, and predicting the parameters in the time series models [45].

Figure 3 shows the structure of the layered ANFIS used in our work. As shown in the Fig. 3, ANFIS is composed of five functional nodes, which are the input nodes, the fuzzification nodes, the product nodes, the normalization nodes, and the defuzzification nodes. Furthermore, these nodes can be classified into the adaptive and fixed nodes based on whether they can be updated. In Fig. 3, the square nodes denote the adaptive nodes and the circular nodes refer to the fixed nodes. To estimate the power output of the fuel-cell, the inputs for the ANFIS include the SOC of the battery, the power output of the ultracapacitor P_{UC}, and the power consumption of the emergency load P_L. The output of the ANFIS is the estimated value of the power output of the fuel-cell P_{FC}. By employing the relative parameters, the ANFIS generates the rules that are tuned efficiently.

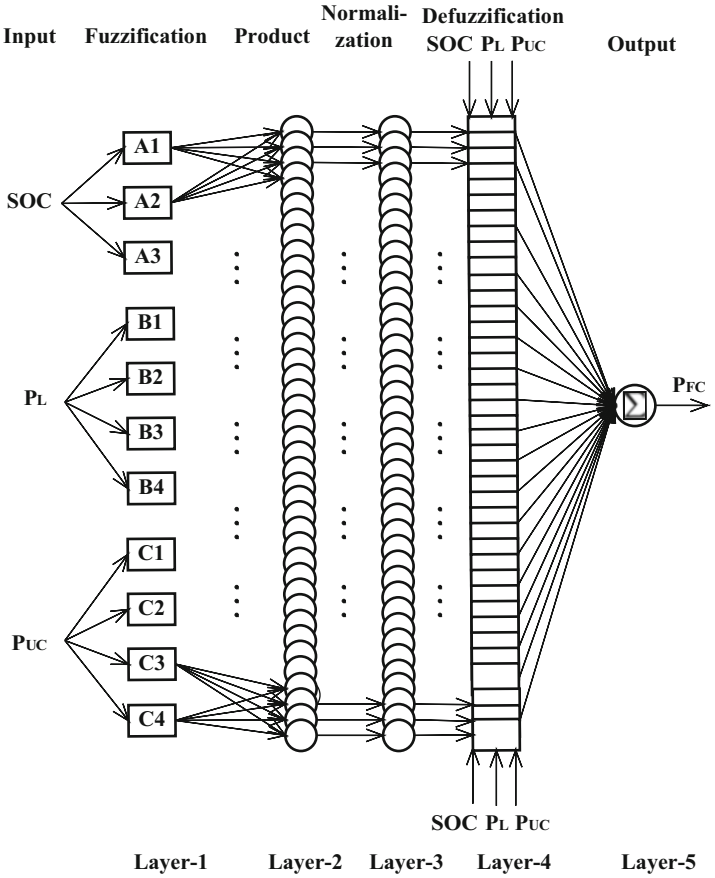


Fig. 3 Illustration of the proposed ANFIS implementation

In the fuzzification layer, the nodes are adaptive and the inputs are fuzzified through the membership functions, such that inputs are labeled linguistically. For simplicity in deriving the equations, we use A, B, and C to represent the linguistic labels as demonstrated in Fig. 3. The membership function μ used in this work is of generalized bell shape and can be calculated by using the following nonlinear equation:

$$\mu(x; a, b, c) = \frac{1}{1 + \left[\frac{x - c}{a}\right]^{2b}} \tag{5}$$

where a, b, and c are antecedent parameters.

The output of the layer can be described as follows:

$$\begin{cases} O_{L_1,i} = \mu A_i(SOC), \\ O_{L_1,j} = \mu B_j(P_L), \\ O_{L_1,k} = \mu C_k(P_{UC}). \end{cases} \tag{6}$$

where $i = 1, 2, 3$, $j = 1, 2, 3, 4$, and $k = 1, 2, 3, 4$.

The functionality of the product layer is to perform the logical AND or product of the input membership functions. The product layer output signifies the input weight function or firing strength of the succeeding node. The output of this layer is described as follows:

$$W_q = O_{L_2,q} = \mu A_i(SOC).\mu B_j(P_L).\mu C_k(P_{UC}) \tag{7}$$

where q denotes the total number of rules that is $3 \times 4 \times 4 = 48$ in our work.

As shown in Fig. 3, the third layer of ANFIS is designed to normalize the input weights. The output of this layer is called the normalized firing strength, which characterizes the IF segment of a fuzzy rule. The layer output can be calculated in the followings:

$$W_q^{'} = O_{L_3,q} = \frac{W_q}{\sum\limits_q W_q} \tag{8}$$

In the defuzzification layer, the nodes are adaptive nodes and the output of this layer can be determined by the following linear equation:

$$W_q^{'} f_q = O_{L_4,q} = \frac{W_q}{\sum\limits_q W_q} \left[r_q(SOC) + s_q(P_L) + t_q(P_{UC}) + u_q \right] \tag{9}$$

where the parameters r_q, s_q, t_q, u_q are considered as consequent parameters which are linear in nature.

Furthermore, the final layer of ANFIS is the output layer which consists of one single fixed node. This single node achieves the overall output by calculating the sum of all the input signals. This layer characterizes the THEN segment of a fuzzy rule. The output of this layer is described as follows:

$$P_{FC} = O_{L_5,q} = W_q^{'} f_q = \frac{W_q f_q}{W_q} \tag{10}$$

In the layered structure of ANFIS, the antecedents of the fuzzy rules are designed to partition the input space into a number of local fuzzy regions, which are the membership functions and the consequents are used to describe the behavior within a given region via a linear equation (the first-order Sugeno fuzzy model is used

in our work). Moreover, we exploit the well-known grid partitioning method for partitioning the input space since this method requires only a small number of MFs for each input [45].

4 Simulation Results

In this section, we evaluate the performance of our proposed intelligent situational awareness system for achieving an attack-resilient emergency control for the hybrid APU of MEA. For performance evaluation, we use the real-time testbed used in [11] which is composed of a 12.5 kW, 30–60 V proton exchange membrane (PEM) fuel cell power module with the nominal power of 10 kW, a 48 V, 40 Ah, Li-ion battery system, a 291.6 V, 15.6 F supercapacitor system, and the associated dc-dc converters and inverters. By using the testbed, we obtain the training datasets for the outcome of energy management strategies. Figure 4a–d show the original data obtained for the SOC of the battery, power output of the ultracapacitor P_{UC}, the power consumption of the emergency load P_L and the power output of the fuel-cell P_{FC}, respectively.

In this simulation, we consider the fuel-cell APU reading is compromised with a false data injection (FDI) attacks. Fuel-cell power output data with a FDI attack is shown in Fig. 4e, where the reading is compromised after time-step 1400. In

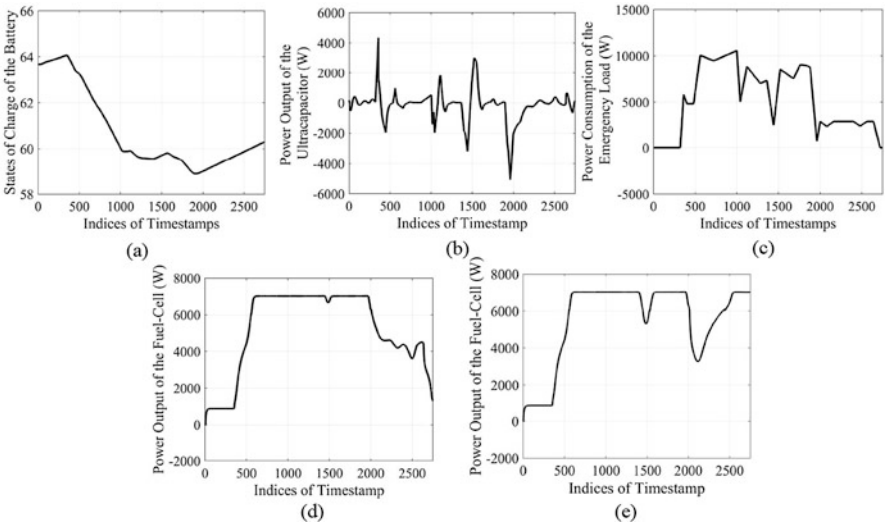


Fig. 4 Original measurement data obtained from APU for, (**a**) the SOC of the battery; (**b**) the power output of the ultracapacitor P_{UC}; (**c**) the power consumption of the emergency load P_L; (**d**) the power output of the fuel-cell P_{FC}; and (**e**) power output of the fuel-cell P_{FC} with a simulated FDI attack

the simulation we consider intelligent attacks launching the false data injection attacks by modifying the measurement of the power output of fuel-cell. The real and compromised measurements of the power output of fuel-cell are shown in Fig. 4d, e, respectively. As illustrated in Fig. 4e, the compromised measurement exhibits similar pattern with the real one. This is because that we assume the attacker is intelligent and has certain knowledge of this system. In this simulation, the time series are divided into 1400 overlapping time windows of size 256 for further processing. Time series for fuel-cell measurement with no attacks are generated for 5 independent simulation instances and all 1400×5 time windows are used for training the network. A separate instance of simulation is used to generated attacked and non-attacked data for testing the trained network.

4.1 Deep Learning Implementation and Results

The deep CNN and DCNN used in the deep learning implementation consist of 3 convolutional and deconvolutional layers, respectively. The input size of the deep CNN is 16×16. The first, second, and third convolution layers consist of 32 5×5, 64 5×5, and 64 2×2 feature kernels, respectively. Each convolution layer is followed by a 2×2 maximum pooling layer. The final $4 \times 4 \times 64$-dimension extracted feature vector is reshaped to a 1×1024 vector and further processed by using LSTM-based RNN. The dimension of the input layer, hidden layer and output layer of the LSTM blocks are set as 1×1024, respectively. The output of the final LSTM block z_t is reshaped back to $4 \times 4 \times 64$ dimension and used as the input of the third deconvolution layer. The dimension parameters of the deep DCNN are set to regenerate the inputs of corresponding convolution layers. Our proposed deep leaning-based attack detection method is implemented using the TensorFlow API [51].

The time windows of size 256 are reshaped to 16×16 data streams and the deep learning method is used to predict APU reading of future time windows when that of current and previous time windows are known. In the implementation, 5 previous time windows are considered. Figure 5 shows the MSE between prediction and actual data and Fig. 6 shows the MSE between prediction and FDI attacked data of power output of the fuel-cell P_{FC}.

From Figs. 5 and 6, we can observe that the MSE between prediction and FDI attacked measurements becomes larger compared to MSE between actual an the predicted measurements for the later time windows, when the actual attack occurred. Therefore, we can define a hard threshold for the measurement to identify an FDI attack considering MSE between predicted and real-time measured data. By using the MSE threshold 0.01 considering the time window indices above 1000 we can detect the considered attack with accuracy of 92.1% with a false alarm rate of 2.7%.

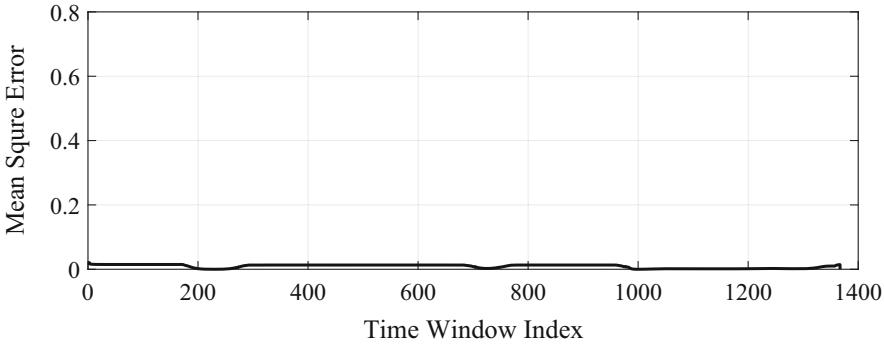


Fig. 5 MSE between actual data obtained from simulation and predictions obtained from deep learning-based method

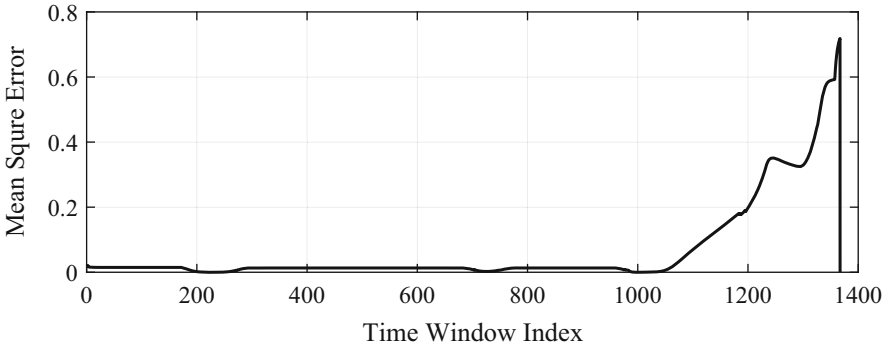


Fig. 6 MSE between compromised data obtained from simulation and predictions obtained from deep learning-based method

4.2 ANFIS Implementation and Results

After using the deep learning-based method for detecting an attack, ANFIS is used to estimate the correct values for the compromised measurement. For the ANFIS implementation, to define the membership functions (MFs), 1000 training data are selected and the system is tested by using 600 different testing datasets. Figure 7 shows the generalized bell shaped MFs used in this simulation. The SOC of the battery has 3 MFs that are Low (L), Medium (M), and High (H). The output power of the ultra-capacitor presents 4 MFs that are Very Low (VL), Low (L), Medium (M), High (H) and the power consumption of the emergency load profile is fuzzified by 4 MFs that are Low (L), Medium (M), High (H) and Very High (VH). The emergency load is simulated by using the typical emergency landing scenario from Bombardier Aerospace [11].

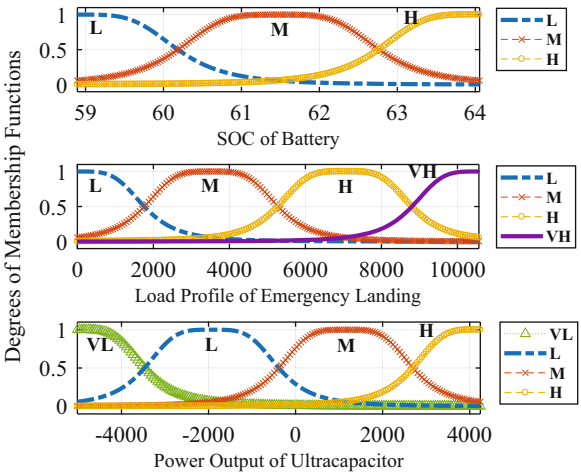


Fig. 7 Input membership functions

Since we considered fuel cell power output as attacked, we feed into the ANFIS the other 3 input measurements and estimated the output power of fuel cell. Figures 8a–d compare the accutal fuel cell power output with the estimated results by considering that the ANFIS-based strategy adopts the following four conventional control strategies to control the hybrid APU: (1) state machine control strategy [12], (2) frequency decoupling fuzzy logic control strategy [15], (3) classical PI control strategy [13], and (4) equivalent fuel consumption minimization (ECMS) control strategy [16]. The mean absolute errors (MAEs) of estimations are shown in Table 1. To achieve the fair comparison, we use the same sets of training and testing data for the evaluation of both ANFIS-based and existing ANN-based scheme. The comparison results are shown in Table 1. Additionally, the numbers of the hidden neurons for the ANN is set to be 48, which is the same as the number of rules used for the ANFIS.

5 Conclusion

In this chapter, we propose an intelligent situational awareness system for securing the energy management systems of emergency hybrid APU in MEA by employing deep learning and ANFIS techniques. Our strategy consists of two mechanisms: (1) deep leaning-based attack detection scheme and (2) ANFIS-based estimation scheme. Our deep learning-based attack detection scheme is developed to detect the cyber-attacks, such as FDI attacks in real time, by exploiting deep CNN, DCNN, and RNN techniques. Additionally, our ANFIS-based estimation scheme is designed to estimate the real values of the compromised measurements that are critical for energy management. In our simulations, we evaluate our intelligent situational awareness system by assuming there are FDI attacks on the measurements of

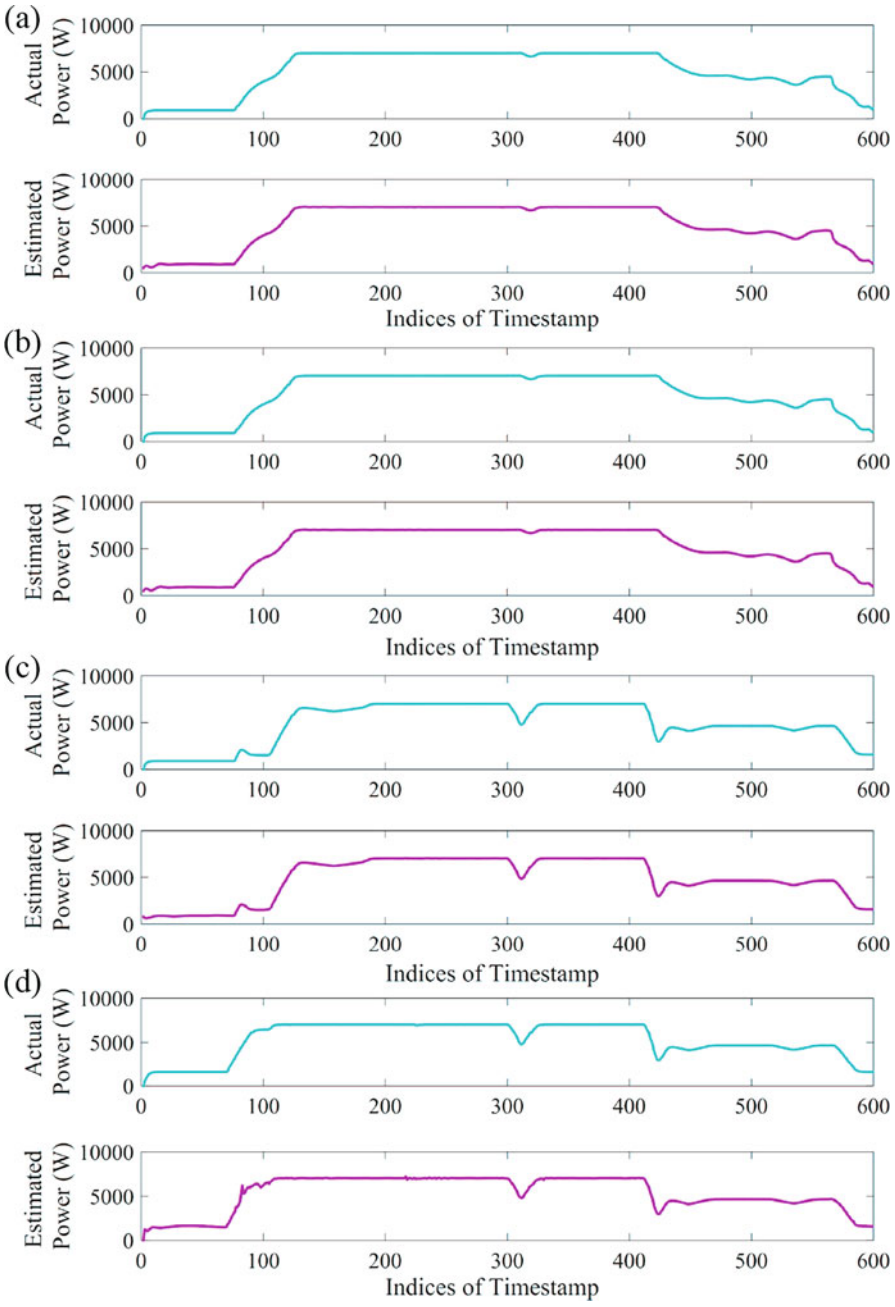


Fig. 8 Comparison results by considering, (**a**) state machine control; (**b**) frequency decoupling fuzzy logic control; (**c**) classical PI control; and (**d**) ECMS control strategies

Table 1 Performance comparison by using mean absolute error (W)

Security control method	State machine control	Frequency decoupling and fuzzy logic	Classical PI control	ECMS
ANFIS method	10.5511	10.5437	12,5001	31.1472
Existing method	13.6000	13.6815	15.0932	35.0248

the power outputs of the fuel-cell in APU. Proposed deep learning-based method shows above 92% accuracy for detecting the considered FDI attack on fuel-cell measurements while the false alarm rate remains under 3%. Additionally, in our simulations we evaluate the ANFIS-based estimation strategy by considering the same scenario in which the measurement of the power output of the fuel-cell is compromised. Also, we consider four different APU control schemes for the evaluation. The experimental results show that our method outperforms the existing method in all the four scenarios.

References

1. Rosero J, Ortega J, Aldabas E, Romeral L (2007) Moving towards a more electric aircraft. IEEE Aerosp Electron Syst Mag 22(3):3–9
2. Wu S, Li Y (2014) Fuel cell applications on more electrical aircraft. In: 2014 17th International Conference on Electrical Machines and Systems (ICEMS). IEEE, pp 198–201
3. Roboam X, Langlois O, Piquet H, Morin B, Turpin C (2011) Hybrid power generation system for aircraft electrical emergency network. IET Electr Syst Transp 1(4):148–155
4. Ziaeinejad S, Sangsefidi Y, Mehrizi-Sani A (2016) Fuel cell-based auxiliary power unit: EMS, sizing, and current estimator-based controller. IEEE Trans Veh Technol 65(6):4826–4835
5. Akhave A, Chrysakis G, Gupta A (2014) Design and evaluation of a turbo-generator as an auxiliary power unit for hybrid vehicles. In: 5th IET Hybrid and Electric Vehicles Conference (HEVC 2014), 5 November 2014, IET, pp 1–7
6. Desideri U, Giglioli R, Lutzemberger G, Pasini G, Poli D (2017) Auxiliary power units for pleasure boats. In: 2017 6th International Conference on Clean Electrical Power (ICCEP). IEEE, pp 650–655
7. Rajashekara K, Jia Y (2016) An induction generator based auxiliary power unit for power generation and management system for more electric aircraft. In: 2016 IEEE Energy Conversion Congress and Exposition (ECCE). IEEE, pp 1–7
8. Waheed M, Cheng M (2017) A system for real-time monitoring of cybersecurity events on aircraft. In: 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC). IEEE, pp 1–3
9. Kumar SA, Xu B (2017) Vulnerability assessment for security in aviation cyber-physical systems. In: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). IEEE, pp 145–150
10. Huang X, Dong J (2017) Adaptive optimization deception attack on remote state estimator of aero-engine. In: 2017 29th Chinese Control and Decision Conference (CCDC). IEEE, pp 5849–5854
11. Motapon SN, Dessaint L-A, Al-Haddad K (2014) A robust H_2-consumption-minimization-based energy management strategy for a fuel cell hybrid emergency power system of more electric aircraft. IEEE Trans Ind Electron 61(11):6148–6156

12. Garcia P, Fernandez LM, Garcia CA, Jurado F (2010) Energy management system of fuel-cell-battery hybrid tramway. IEEE Trans Ind Electron 57(12):4013–4023
13. Thounthong P, Raël S, Davat B (2007) Control strategy of fuel cell and supercapacitors association for a distributed generation system. IEEE Trans Ind Electron 54(6):3225–3233
14. Li C-Y, Liu G-P (2009) Optimal fuzzy power control and management of fuel cell/battery hybrid vehicles. J Power Sources 192(2):525–533
15. Vural B, Boynuegri A, Nakir I, Erdinc O, Balikci A, Uzunoglu M, Gorgun H, Dusmez S (2010) Fuel cell and ultra-capacitor hybridization: a prototype test bench based analysis of different energy management strategies for vehicular applications. Int J Hydrog Energy 35(20):11161–11171
16. Rodatz P, Paganelli G, Sciarretta A, Guzzella L (2005) Optimal power management of an experimental fuel cell/supercapacitor-powered hybrid vehicle. Control Eng Pract 13(1):41–53
17. Greenwell W, Vahidi A (2010) Predictive control of voltage and current in a fuel cell–ultracapacitor hybrid. IEEE Trans Ind Electron 57(6):1954–1963
18. Hubel DH, Wiesel TN (1968) Receptive fields and functional architecture of monkey striate cortex. J Physiol 195(1):215–243
19. Wei Y, Xia W, Lin M, Huang J, Ni B, Dong J, Zhao Y, Yan S (2016) HCP: a flexible CNN framework for multi-label image classification. IEEE Trans Pattern Anal Mach Intell 38(9):1901–1907
20. Dong C, Loy CC, He K, Tang X (2016) Image super-resolution using deep convolutional networks. IEEE Trans Pattern Anal Mach Intell 38(2):295–307
21. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
22. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp 1097–1105
23. Abdullah, Hasan MS (2017) An application of pre-trained CNN for image classification. In: 2017 20th International Conference of Computer and Information Technology (ICCIT), pp 1–6
24. Mondal M, Mondal P, Saha N, Chattopadhyay P (2017) Automatic number plate recognition using CNN based self synthesized feature learning. In: 2017 IEEE Calcutta Conference (CALCON), pp 378–381
25. Zhang J, Li Y, Yin J (2017) Modulation classification method for frequency modulation signals based on the time–frequency distribution and CNN. IET Radar Sonar Navig 12:244–249
26. Savigny J, Purwarianti A (2017) Emotion classification on youtube comments using word embedding. In: 2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA), pp 1–5
27. Ji S, Xu W, Yang M, Yu K (2013) 3D convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intell 35(1):221–231
28. Gudelek MU, Boluk SA, Ozbayoglu AM (2017) A deep learning based stock trading model with 2-D CNN trend detection. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp 1–8
29. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882
30. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188
31. Dos Santos CN, Gatti M (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: COLING, pp 69–78
32. Abdel-Hamid O, Mohamed A-R, Jiang H, Penn G (2012) Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp 4277–4280
33. Kuen J, Wang Z, Wang G (2016) Recurrent attentional networks for saliency detection. arXiv preprint arXiv:1604.03227
34. Liu N, Han J (2016) DHSNet: deep hierarchical saliency network for salient object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp 678–686

35. Badrinarayanan V, Handa A, Cipolla R (2015) SegNet: a deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293
36. Yang J, Price B, Cohen S, Lee H, Yang M-H (2016) Object contour detection with a fully convolutional encoder-decoder network. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp 193–202
37. Noh H, Hong S, Han B (2015) Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp 1520–1528
38. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
39. Song E, Soong FK, Kang HG (2017) Effective spectral and excitation modeling techniques for LSTM-RNN-based speech synthesis systems. IEEE/ACM Trans Audio Speech Lang Process 25(11):2152–2161
40. Jia G, Lu Y, Lu W, Shi Y, Yang J (2017) Verification method for Chinese aviation radiotelephony readbacks based on LSTM-RNN. Electron Lett 53(6):401–403
41. Gelly G, Gauvain JL (2018) Optimization of RNN-based speech activity detection. IEEE/ACM Trans Audio Speech Lang Process 26(3):646–656
42. Sun L, Su T, Zhou S, Yu L (2017) GMU: a novel RNN neuron and its application to handwriting recognition. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol 01, pp 1062–1067
43. Li W, Wen L, Chang MC, Lim SN, Lyu S (2017) Adaptive RNN tree for large-scale human action recognition. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp 1453–1461
44. Hsu HW, Ding JJ (2017) FasterMDNet: learning model adaptation by RNN in tracking-by-detection based visual tracking. In: 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp 657–660
45. Jang J-S (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685
46. Motapon SN, Dessaint LA, Al-Haddad K (2014) A comparative study of energy management schemes for a fuel-cell hybrid emergency power system of more-electric aircraft. IEEE Trans Ind Electron 61(3):1320–1334
47. Michon M, Duarte J, Hendrix M, Simoes MG (2004) A three-port bi-directional converter for hybrid fuel cell systems. In: 2004 IEEE 35th Annual Power Electronics Specialists Conference, PESC'04, vol 6. IEEE, pp 4736–4742
48. Sak H, Senior A, Beaufays F (2014) Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth Annual Conference of the International Speech Communication Association
49. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
50. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
51. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M et al (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467

Deep Learning in Person Re-identification for Cyber-Physical Surveillance Systems

Lin Wu, Brian C. Lovell, and Yang Wang

Abstract The Cyber-physical Systems (CPS) are a combination of integrated physical processes, networking and computation to be minored and controlled y embedded subsystems via networked systems with feedback loops to change their behaviour when needed. Whilst the increased use of CPS brings more threats to the public, and thus security problems in this area have become a global issue to make it necessary to develop new approaches for securing CPS. The CPS utilise three-level architecture based on the respective functions of each layer: the perception layer, the transmission layer, and the application layer. Security in specific, CPS applications is currently the most important security objective of CPS because it offers the importance of CPS in its improving functionality

This chapter focuses on the application aspect which is more related to people's daily lives, and will present a real-time system including distributed multi-camera system that integrates computing and communicating capabilities with monitoring on people in the physical world, namely person re-identification in the cyber-physical surveillance systems. The increasing sophistication and diversity of threats to public security have been causing a critical demand for the development and deployment of reliable, secure, and time-efficient visual intelligent surveillance systems in smart cities. For example, visual surveillance for indoor environments, like metro stations, plays an important role both in the assurance of safety conditions for the public and in the management of the transport network. Recent progress in computer vision techniques and related visual analytics offers new prospects for an intelligent surveillance system. A major recent development is the massive success resulting from using deep learning techniques to enable a significant boosting to visual analysis performance and initiate new research directions to understand visual content. For example, convolutional neural networks have demonstrated superiority

L. Wu (✉) · B. C. Lovell
The University of Queensland, Brisbane, QLD, Australia
e-mail: lin.wu@uq.edu.au

Y. Wang
Hefei University of Technology, Hefei, China

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_3

on modelling high-level visual concepts. It is expected that the development of deep learning and its related visual analytic methodologies would further influence the field of intelligent surveillance systems. In view of the high demand for a prevalent surveillance system by the metropolis communities, this chapter will introduce recent research based on deep neural networks and pipelines to the practitioners and human investigators undertaking forensic and security analysis of large volumes of open-world CCTV video data sourced from a large distributed multi-camera network covering complex urban environments with transport links. This chapter will address the challenges of using deep learning and related techniques to understand and promote the use of ubiquitous intelligent surveillance systems.

Keywords Cyber-physical system with security · Multi-camera networking · Person re-identification in cyber-physical security

1 Introduction

The networked systems of cyber and physical components consist of Cyber physical Systems (CPS) that interact in a feedback loop with the possible help of human intervention, interaction and utilisation. With the increased focus on data handling capacity, data communications capability and integration of information systems, the demand or integrating CPS in different fields is also increasing, resulting in widely gained attention from research and many agencies [3]. The current form of CPS is used in many different areas such as water industry, healthcare, transportation, automotive systems, consumer appliances in addition to many other areas that are directly related to people's daily lives. The CPS are featured to produce complex systems by joining individual components. Typically, data can be collected by sensor devices, and transferred through the networks to the control system with absence of any human to machine interactions [6]. However, the increased connectivity of the cyber and physical world raise significant security challenges to the CPS because the systems are improving the functionality and also the interconnectivity among CPS subsystems are growing. The modern definition of CPS is the integration of three layers: the perception layer, the transmission layer, and the most interactive application layer [3], as shown in Fig.1.

In general, this chapter primarily focuses on the application layer which is missioned to process the received information from the data transmission layer and implements complex decision-making algorithms on the aggregated data to generate correct decisions. The objective of this chapter is to create smart environment, and combine CPS to enable intelligent application in the areas of Intelligent Transportation and Smart Cities. Thus, this chapter falls into the information security of CPS security [3] which involves securing information during data aggregation (i.e. video streams collected from surveillance nodes), processing and large-scale sharing in the network environment (i.e., visual analysis on person association to refer the topology of networks), especially open loosely coupled transportation networks/hubs across the city.

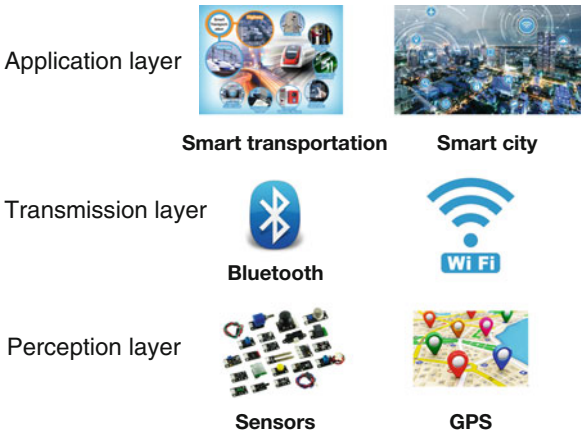


Fig. 1 The architecture of CPS with three layers

Many recent studies on person re-identification (re-id) attempt to generate robust feature representation which is discriminative and robust for describing pedestrian appearance under various changes and conditions [5, 14, 20, 62, 87, 88]. Bazzani et al. [5] represent a person by a global mean colour histogram and recurrent local patterns through epitomic analysis. Farenzena et al. [14] propose the symmetry-driven accumulation of local features (SDALF) which exploits both symmetry and asymmetry, and represents each part of a person by a weighted color histogram, maximally stable colour regions and texture information. Gray and Tao [20] propose to use AdaBoost to select good features out of a set of colour and texture features. Schwartz and Davis propose a discriminative appearance based model using partial least squares, in which multiple visual features: texture, gradient and colour features are combined [47]. Recently, saliency information has been investigated for person re-id [86, 87], leading to a novel feature representation and improved discriminative power in person re-id. In [87], a method of eSDC is presented to learn salience for persons under deformation. Moreover, salience matching and patch matching can be integrated into a unified RankSVM framework (SalMatch [86]). They also propose mid-level filters (MidLevel) for person re-identification by exploring the partial area under the ROC curve (pAUC) score [88]. Lisanti et al. [37] leverage low-level feature descriptors to approximate the appearance variants in order to discriminate individuals by using sparse linear reconstruction model.

Metric learning approaches to person re-id essentially formalise the problem as supervised metric/distance learning where a projection matrix is sought out so that the projected Mahalanobis alike distance is small when feature vectors represent the same person and large otherwise. Typically, Large Margin Nearest Neighbor Learning (LMNN) [63], Information Theoretic Metric Learning (ITML) [12], and Logistic Discriminant Metric Learning (LDM) [21] are three representative methods. By applying these metric learning methods into person re-id, many effective approaches are developed [23, 26, 34, 35, 40, 43, 74, 76, 91]. Mignon et al. [40] proposed Pairwise Constrained Component Analysis (PCCA) to learn a projection into a low dimensional space in which the distance between pairs of

samples respects the desired constraints, exhibiting good generalisation properties in the presence of high dimensional data. Zheng et al. [91] presented a Relative Distance Comparison (RDC) to maximise the likelihood of a pair of true matches having a relatively smaller distance than that of a mismatched pair in a soft discriminant manner. Koestinger et al. propose the large-scale metric learning from equivalence constraint (KISSME) which considers a log likelihood ratio test of two Gaussian distributions [26]. Li et al. propose the learning of locally adaptive decision functions (LADF), which can be viewed as a joint model of a distance metric and a locally adapted thresholding rule [34]. The Cross-view Quadratic Discriminant Analysis (XQDA) algorithm learns a discriminant subspace and distance metric simultaneously. It is able to perform dimension reduction and select the optimal dimensionality. To make the metric learning more efficient, they further present a positive semi-definite constrained method to reduce the computation cost and get more robust learned metric. In [35], an efficient feature representation called Local Maximal Occurrence is proposed, followed by a subspace and metric learning method. Last but not least, learning to rank can be employed in person re-id, and approaches include ensemble RankSVM [45], Metric Learning to Rank (MLR) [38] with application to person re-id [74] and structured metric ensembling [42].

With the resurgence of deep Convolutional Neural Networks (CNNs), a large body of deep learning based person re-id algorithms have been proposed [1, 9–11, 30, 32, 53, 55, 56, 66, 84, 85, 94] to learn high-level representations with discriminations. In [1], a Siamese CNN with a symmetry structure comprising two independent sub-nets is introduced and jointly learned the distance metric in pairwise. Li et al. [32] designed a different network, which begins with a single convolution layer with max pooling, followed by a patch-matching layer that multiplies convolutional feature responses from the two inputs at a variety of horizontal offsets. The PersonNet is a more deep architecture from [1] where more convolution/max pooling layers are applied in the context of very small filters. To further improve the re-id accuracy, some deep models are designed to consider the spatial patch alignment [7, 30, 55, 56, 72, 94] or consider to encode local body parts into a global representation [9, 85]. All above mentioned methods, if not all, are supervised and they require label supervision or body part annotations to produce the discriminative representations against identities. On the other hand, unsupervised learning alternatives [13, 58, 79, 87] are introduced into tackle person re-id by discovering salient features [58, 87] or transferring representations across datasets [13, 44]. Some researchers attempt to bring domain adaptation techniques to alleviate the cross-view disparity in the matching of pedestrians [80].

This chapter aims to introduce the readers more recent progress towards deep learning based person re-id for CPS security in the matter of information security, and shed light on the development of varied frameworks to the case of person re-id in different settings. The rest of the chapter is structured as follows. In Sect. 2, some recent supervised and unsupervised learning methods will be briefly described. In Sect. 3, a set of deep learning methodologies are presented in terms of image-based person re-id (Sect. 3.1) and video-based person re-id (Sect. 3.2). In the stream of image-based person re-id, a prototype of deep feature learning

framework is presented, and an improved hash learning model for the case of fast searching candidates in person re-id is also introduced. Also, an unsupervised learning algorithm based on generative adversarial neural networks is introduced to allow the effective re-identification without supervision. Finally, Sect. 4 concludes this chapter and some open problems are discussed in Sect. 5.

2 Background

Since computing devices are becoming lightweight, portable, and capable of being connected with the real world, CPS components can be interconnected through the Internet with the capability of system monitoring and controlling with proper operation and real-time response. CPS are providing a coupled environment that real-time response contains interconnectivity of thousands of device, enabling more convenience in management and control.

More recently, person re-id cyber system with security is mainly related to real-time systems including distributed real-time control camera networks that integrate computing and communication capabilities with monitoring and control of entities in the physical world, for example, the transportation hub, airport terminals, and a whole smart city. When designing next generation security solutions, it is crucial to combine sensing, computing, understanding, communication and prediction in such networked-camera systems. Examples include automated video surveillance platforms and smart camera networked systems that are monitoring behaviour and activities, or other changing information in the environment, for the purpose of influencing, managing, directing, or protecting people. They exhibit a high-level of awareness beyond primitive actions, in support of persistent and long-term autonomy. However, some core problems such as person identification and tracking, and behaviour analysis in intelligent surveillance, are still affected by a number of practical problems. They typically involve a variety of representation, reasoning and efficiency mechanisms in the context of an extended distance and period of time and low resolution/frame rate in poor quality capturing conditions. A significant research problem in visual surveillance is person re-identification [17] which refers to associating individuals across disjointed cameras at different locations and in different time periods. As core to multi-camera surveillance systems, person re-id is also of great security interest and can be used for various surveillance applications, such as facilitating cross-camera tracking of people and understanding their global behaviour in a wider context.

2.1 Supervised Learning in Person Re-identification

The task of person re-identification can be accomplished by two categories of methods: (i) learning distance or similarity measures to predict if two images

describe the same person [7, 31, 34, 49, 76, 81, 91], and (ii) designing distinctive signature to represent a person under different cameras, which typically performs classification on cross-image representation [1, 7, 32, 53, 55, 57, 68]. For the first category of methodologies, they usually use many kinds of hand-crafted features including local binary patterns [26, 76], colour histogram [26, 67], local maximal occurrence (LOMO) [35, 36], and focus on learning an effective distance/similarity metric to compare the features. For the second category, deep convolutional neural networks are very effective in localising and extracting relevant features to form discriminative representations against view variations. For instance, identity-preserving projections can be learned through optimising the identification loss [53] or a combination loss of identification and pairwise loss [57]. However, all these re-id models are in a supervised manner and rely on substantial labeled training data, which are typically required to be in pair-wise for each pair of camera views. Their performance depends highly on the quantity and quality of labeled training data, which also limits their application to large-scale networked cameras. In contrast, unsupervised learning methods based on generative modelling do not require any labeled data, and thus are free from prohibitively high cost of manual labelling and the risk of incorrect labelling.

2.2 Unsupervised Learning in Person Re-identification

Most of existing re-id models are developed in a supervised manner to learn discriminative features [68, 85, 89] or learn distance metrics [33, 57, 81]. However, all these models rely on substantial labeled training data, which hinder the application of them in large networked cameras. Semi-supervised and unsupervised methods are presented to overcome the scalability issue by using limited number of labeled samples or without using label information. These techniques often focus on designing handcrafted features (e.g., color, texture) [4, 58, 87] that should be robust to visual changes in imaging conditions. However, low-level features are not expressive in view-invariance because features are not learned to be apt to view-specific bias. Transfer learning has been applied into re-id [13, 44, 92], and these methods learn the model using large labeled datasets and then transfer the discriminative knowledge to the unlabelled target camera pair. More recent approaches to re-id aim to learn a cross-view metric either by asymmetric clustering on person images [79] or by transferable colour metric from a single pair of images [4]. However, there is still a considerable performance gap relative to supervised learning approaches because it is not principled to fully explore the discriminative space in the context of limited labeled samples available in training. To achieve comparable performance to supervised pipelines, this chapter will introduce an effective view alignment approach to statistically discover co-occurrence patterns that help to produce discriminative features automatically.

3 Deep Learning in Person Re-identification for Cyber-Physical Surveillance Systems: New Trending Methodologies

Person re-identification is essentially to measure the similarity for pairs of pedestrian images in a way that a pair is assigned a high similarity score in case of depicting the same identity and a low score if displaying different identities. This typically involves constructing a robust feature representation and an appropriate similarity measure in order to estimate accurate similarity scores. To this end, many methods focusing on feature representation and distance function learning are designed separately or jointly to deal with person re-id problem. Low-level features such as colour and texture can be used for this purpose. Some studies have obtained more distinctive and reliable feature representations, including symmetry-driven accumulation [14], horizontal partition [35], and salience matching [87]. However, it is still difficult to design a type of feature that is discriminative and invariant to severe changes in terms of misalignment across disjoint camera views. Another pipeline of person re-id system is to learn a robust distance or similarity function to deal with complex matching problem. Many metric learning algorithms are proposed for this purpose [4, 34, 35, 43, 49, 57, 68, 76, 79]. In practice, most of metric learning methods exhibit a two-stage processing which typically extract hand-crafted features and subsequently learn the metrics. Thus, these approaches often lead to sub-optimal solutions.

Convolutional Neural Networks (CNNs) have proven highly successful at image recognition problems and various surveillance applications including pedestrian detection [41, 93], and tracking [60]. Also, substantial progress is witnessed in person re-id such as [1, 9–11, 30, 32, 53, 55, 56, 66, 84, 85, 94]. By applying CNNs, a joint feature representation and metric learning can be achieved. The FPNN algorithm [32] makes the first attempt to introduce patch matching in CNNs, followed by an improved deep learning framework [1] where layers of cross-input neighborhood differences and patch summary are added. These two methods both evaluate the pair similarity early in the CNN stage, so that it could make use of spatial correspondence of feature maps. In fact, spatial misalignment is very notable in person re-id due to similar appearance or occlusion [72, 94]. As a result, a more deep model is demanded to well address this challenge by faithfully capturing non-linear relationship between patches.

3.1 Image-Based Person Re-identification

The task of image-based person re-identification is to match pedestrian static images observed from multiple non-overlapping camera views with varied visual features [34, 36, 43, 53, 58, 66, 72, 76, 87, 91]. This problem has been extensively studied to cope with the challenges which are presented in the form of compounded variations

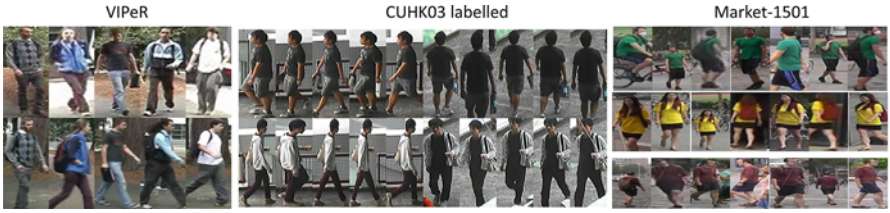


Fig. 2 Typical samples of pedestrian images in person re-identification from the VIPeR [19], labeled CUHK03 [32] and Market-1501 [90] datasets. Each column shows two images of the same individual observed by two different camera views

in visual appearance across different camera views, human poses, illuminations, background clutter, occlusions, relatively low resolution and the different placement of the cameras. Some image examples from widely known person re-id datasets are shown in Fig. 2. In what follows, the section will outline two plausible pipelines in addressing the problem of person re-id in terms of the employment of label supervision on data samples or without label supervision.

3.1.1 Supervised Deep Representation Learning

PersonNet [66] is a pioneering work that is inspired by deep VGG network [50] to deepen a state-of-the-art [1] by using very small convolutional filters throughout the architecture. During training, the input to the PersonNet is a pair of fixed-size 160×60 RGB images. The pair of images is passed through a stack of tied convolutional layers, where we use filters with a very small receptive filed: 3×3. The convolution stride is fixed to 1 pixel. Spatial pooling is carried out by three max-pooling layers, which follow some of the convolution layers (not all the convolution layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2. After a stack of convolution layers, we have three fully-connected layers where the first two have 4096 dimension and the third is 512, which is put through soft-max to determine the pair is same or different.The overall architecture of the PersonNet is shown in Fig. 3.

Convolution and Max Pooling

The first two layers are convolutional and max-pooling layers. Given two pedestrian images $\mathbf{I}$ and $\mathbf{J}$ observed by two different camera views with three color channels and sized 160×60, the convolutional layer outputs local features extracted by filter paired. The filters $(\mathbf{W}, \mathbf{V})$ applied to two camera views are shared. Given the input $\mathbf{I}$ and $\mathbf{J}$, consisting of C channels of height H and width W, if we use K filters and

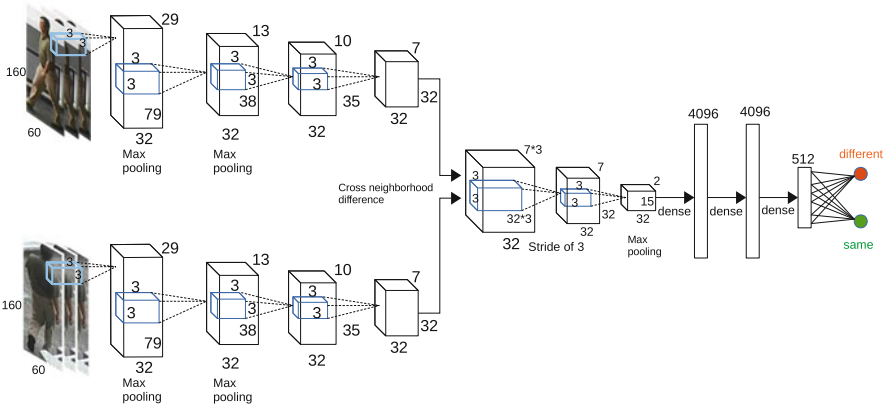


Fig. 3 The architecture of PersonNet. The network takes a pair of RGB images as input, which is put through a stack of convolution layers, matching layer, and higher layers computing relationships between them

each filter is in size of $m \times m \times C$, the output consist of a set of C' channels of height H' and width W'. We define the filter functions as $f, h : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H' \times W' \times C'}$

$$f_{ij}^{k} = \sigma((\mathbf{W}_k * \mathbf{I})_{ij} + b_k^I) \tag{1}$$
$$h_{ij}^{k} = \sigma((\mathbf{V}_k * \mathbf{J})_{ij} + b_k^J),$$

where $k \in \mathbb{R}^{K \times K \times C}$. Rather than using relatively large receptive fields in the first two convolutional layers (e.g., 5×5 in [1, 32]), we use small receptive fields with 3×3 throughout the whole net to convolute with the input at every pixel with stride of 1. Apparently, a stack of two 3×3 convolution layers (without spatial pooling between them) amounts to working as a receptive field of 5×5. By doing this, more non-linear activation functions are embedded which can make the decision function more discriminative [50].

Activation function can increase the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolutional layer. Instead of using ReLU, $\sigma(\mathbf{x}) = \max(0, \mathbf{x})$, as the activation function in deep network, we choose the nonlinear activation function $\sigma(\mathbf{x})$ to be hyperbolic tangent function, $\sigma(\mathbf{x}) = tanh(\mathbf{x}) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, which can re-scale the linear output in the range $[-1, 1]$. Scaling the activation function to be $\sigma(\mathbf{x}) = tanh(\frac{3\mathbf{x}}{2})$ is able to ensure the training is spread uniformly over each layer particularly helpful in networks with very deep layering. The max-pooling layer is used to reduce the dimensionality of the output and variance in deformable objects to ensure that the same result will be obtained even when image features undergo slight translations. The max-pooling operation is applied on every pixel around its neighbourhood.

Modelling Neighbourhood Patch Matching

The person re-id problem is typically characterised of large cross-view variations, misalignment and pose variations. This renders some patches mismatched on the same person. The patch matching layer is introduced to computes the differences in filter responses of local patches across two views. Since we have $f_{\mathbf{I}}, g_{\mathbf{J}} \in \mathbb{R}^{32\times7}$, the difference around the neighbourhood of each feature location yields to a set of feature maps $K_i \in \mathbb{R}^{32\times7\times3\times3}$ $(i = 1, \ldots, 32)$, where 3×3 is the window size of neighborhood around a feature value. In other words, K_i indicates a 32×7 grid of 3×3 blocks, $K_i(x, y) \in \mathbb{R}^{3\times3}$ where $1 \leqslant x \leqslant 7$ and $1 \leqslant y \leqslant 32$. Following [1], we have

$$K_i(x, y) = f_{\mathbf{I}}(x, y)\mathbb{I}(3, 3) - \mathcal{N}[h_{\mathbf{J}}(x, y)] \tag{2}$$

where $\mathbb{I} \in \mathbb{R}^{3\times3}$ is a 3×3 indicator matrix with all elements being 1s, and $\mathcal{N}[h_{\mathbf{J}}(x, y)] \in \mathbb{R}^{3\times3}$ is a 3×3 neighbourhood of $h_{\mathbf{J}}$ centered at (x, y). Here we use a small neighbourhood of size 3×3 to model the displacement of body parts caused by pose and viewpoint variations [32]. Our architecture can avoid symmetric operation on computing K_i because we use online sampling to generate pairs of images.

The visualisation of feature responses at each layer of the network are shown in Fig. 4. We can see that after Conv0, the features responds to bright regions of the images. After a few convolutions and max-pooling, higher responses are given to body as a whole. In this process, part-based CNNs maybe beneficial to further improve the accuracy of recognition since human body parts can be very different across camera views and matching different parts are helpful in matching. Recall from Sect. 3.1.1 and [1], the neighbourhood layer is to compute the difference of corresponding feature maps across two views in a small range. This can robustly match some patches that undergo variations in viewpoints, and poses. For a negative pair, a neighbourhood difference layer can highlight some local patches that are visually different, as shown in Fig. 4a. By contrast, for a positive pair, the difference map is expected to be close to zero and nonzero values should be small and uniformly distribute across the map, as shown in Fig. 4b. The difference layer is followed by another patch summary layer that extract these difference maps into

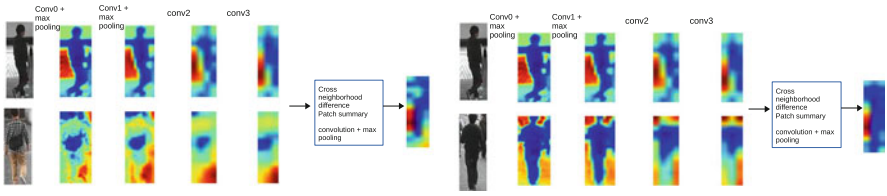


Fig. 4 Feature responses of each layer learned by the PersonNet network. (**a**) A pair of negative images. (**b**) A pair of positive images

a holistic representation of the differences in each 3×3 block. Then, we use another convolution layer with max pooling to learn spatial relationships across neighbourhood differences. The network ends up with three fully connected layers with soft-max output.

Training Strategies

We use the ReLU as the non-linear activation function in our models. Training data are divided into mini-batches where each mini-batch consists of randomly sampled image pairs. Input pairs are put through the network to compute feature representations feed-forwardly, and gradients computed from SGD with variance reduction over each mini-batch are back-propagated to update sets of parameters. To train the parameters of these supervised layers, we use the objective function that involves minimising the cross-entropy between the network output and the ground-truth. The last soft-max layer uses the soft-max function to determine whether the two input images **I**,**J** belong to the same person or not. The output is a binary variable $y_{I,J}$, defined as

$$p(y_{\mathbf{I},\mathbf{J}} = i|\mathbf{W}, b) = \frac{e^{(\mathbf{W}_i \cdot \mathbf{x} + b_i)}}{\sum_i e^{(\mathbf{W}_i \cdot \mathbf{x} + b_i)}}. \tag{3}$$

Let $y_{\mathbf{I},\mathbf{J}} = 1$ if **I** and **J** are matched, and $y_{\mathbf{I},\mathbf{J}} = 0$ otherwise. **x** is the input from the last fully-connected layer. **W** and b are the weights and bias terms to be learned. Thus, following the setting in [32], given the class labels of N training sample pairs, the negative log-likelihood is used as the contrastive loss function for training and could be written as

$$\begin{aligned} loss = &\sum_{n}^{N} y_n \log(p(y = 1|\Phi, (\mathbf{I}_n, \mathbf{J}_n))) \\ &+ (1 - y_n) \log(1 - p(y = 1|\Phi, (\mathbf{I}_n, \mathbf{J}_n))). \end{aligned} \tag{4}$$

3.1.2 Deep Hashing Learning for Fast Person Re-identification

Recently, deep learning methods [1, 8, 32, 66–68, 72, 75] have been proposed to address the problem of person re-identification by learning deeply discriminative Convolutional Neural Network (CNN) features in a *feed-forward* and *back-propagation* manner. It extracts hierarchical CNN features from pedestrian images; the subsequent metric-cost part compares the CNN features with a chosen metric encoded by specific loss functions, e.g., contrastive (pair-wise) [1, 32, 66] or triplet [8, 57] loss functions. However, such typical deep learning methods are not efficient in real-time scenario, due to the less-efficiency of matching two pedestrian images by extracting and comparing hierarchical CNN features. In fact,

the excellent recognition accuracy in neural network-based architectures comes at expense of high computational cost caused by the sequential updates in gradient decent of optimising an objective function with a contrastive or triplet loss. As a result, the learning is slow by taking pairwise or triplet units as input, and the main computational expense for these deep models comes from many weight updates using stochastic gradient descent. This is, in our view, is mostly due to the less effective data sampling strategy underpinned by the simplified objective function without any high-order information. And directly matching these CNN features to obtain similarity values is not fast enough to be applicable in real-world applications. In this section, we aim to reduce the computational burden of person re-identification by developing a fast re-identification framework whereby discriminative feature representations and hashing codes are learned effectively under a structured loss function which can automatically mine out meaningful hard examples, and exert boosting effect on parameter update during the back-propagation, leading to fast convergence and improved performance.

To cope with ever-growing amounts of visual data, deep learning based hashing methods have been proposed to simultaneously learn similarity-preserved hashing functions and discriminative image representation via a deep architecture [28, 73, 82, 83]. Simply delving existing deep hashing approaches into a person re-identification system is not trivial due to the difficulty of generalising these pre-trained models to match pedestrian images in disjoint views. Fine-tuning is a plausible way to make pre-trained models suitable to re-identification, however, to suit their models, training images are commonly divided into mini-batches, where each mini-batch contains a set of *randomly* sampled positive/negative pairs or triplets. Thus, a contrastive or triplet loss is computed from each mini-batch, and the networks try to minimise the loss function and update the parameters through back-propagation by using Stochastic Gradient Decent (SGD) [64]. We remark that randomly sampled pairs/triplets carry little helpful information to SGD. For instance, many triplet units can easily satisfy the relative comparison constraint in a triplet loss function, resulting into a slow convergence rate in the training stage since many of them easily satisfy the constraint well and give nearly zero loss. Worse still, mini-batches with random samples may fail to obtain a stable solution or collapse into a local optimum if a contrastive/triplet loss function is optimised [51]. To this end, a suitable loss function is highly demanded to work well with SGD over mini-batches, which has desirable property of augmenting meaningful hard examples.

This section will present a deep hashing scheme based on CNNs to efficiently address the problem of person re-identification [70]. To mitigate the undesirable effects caused by contrastive/triplet loss function, a structured loss function that *actively* adds hard negative samples into mini-batches is introduced, leading to a structured deep hashing framework. The proposed structured loss can guide sub-gradient computing in SGD to have correct directions, and thus achieves a fast convergence in training. Meanwhile, similarity-preserving hashing functions are jointly learned to enable a fast re-identification system. The overview of the hash framework is illustrated in Fig. 5.

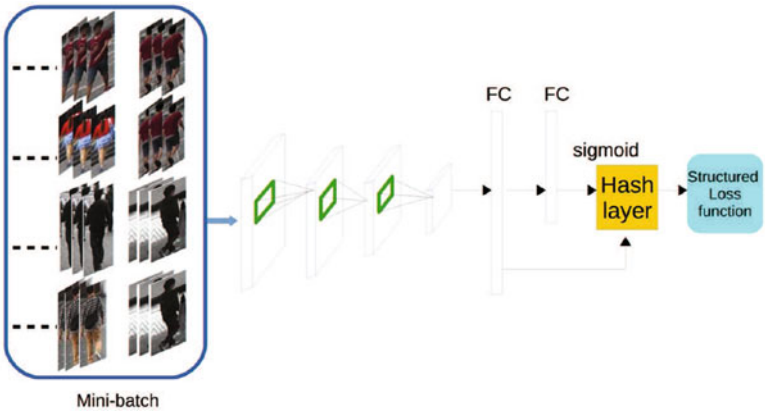


Fig. 5 Overview of the deep hashing framework for person re-identification. The deep neural network takes a feed-forward, back-propagation strategy to learn features and hash codes simultaneously. During the feed-forward stage, the network performs inference from a mini-batch. The mini-batch is put through a stack of convolutional layers to generate nonlinear yet discriminative features, which are subsequently mapped to output feature vectors by fully-connected layers (FC). Meanwhile, a hash function layer is introduced atop of FC layer to learn hash codes that are optimized by a structured loss function to preserve their similarities/dissimilarities. In back-propagation, parameters are updated by computing their Stochastic Gradient Decent (SGD) w.r.t. the mini-batch

Learning Deep Hashing Functions

Assuming $\mathcal{I}$ to be the original image space, a hash function $f : \mathcal{I} \rightarrow \{0, 1\}^r$ is treated as a mapping that projects an input image $\mathcal{I}$ into a r-bit binary code $f(\mathcal{I})$ while preserving the similarities of person images across camera views. Learning based hashing methods aim to seek a set of hash functions to map and quantize each sample into a compact binary code vector. Assuming we have r hash functions to be learned, which map an image $\mathcal{I}$ into a r-bit binary code vector $f(\mathcal{I}) = [f_1(\mathcal{I}), f_2(\mathcal{I}), \ldots, f_r(\mathcal{I})]$.

During training, the input to the network is a mini-batch containing pairs of fixed-size 160×60 RGB images. The images are passed through four convolutional layers, where we use filters with a very small receptive filed: 3×3. The convolution stride is fixed to 1 pixel. Spatial pooling is carried out by three max-pooling layers. Max-pooling is performed over a 2×2 pixel window, with stride 2. After a stack of convolution layers, we have two fully-connected layers where the first one has 4096 dimension and the second is 512-dim, which are then fed into the hash layer to generate a compact binary code. Inspired by [52], a bypass connection between the first fully connected layer and the hash layer is added to reduce the possible information loss. Another reason is features from the second fully connected layer are very semantic and invariant, which is unable to capture the subtle difference between person images. Thus, we rewrite the deep hash function as:

$$f(\mathcal{I}, \mathbf{w}_i) = \pi\left(sigmoid\left(\mathbf{w}_i^T[g_1(\mathcal{I}); g_2(\mathcal{I})]\right)\right), \tag{5}$$

where $sigmoid(t) = 1/(1 + \exp^{(-\beta^T t)})$, β is a hyper-parameter. $\mathbf{w}_i$ denotes the weights in the i-th hash function, $g_1(\cdot)$ and $g_2(\cdot)$ represent feature vectors from the outputs of the two fully connected layers, respectively. $\pi(\cdot)$ is the piece-wise threshold function, which is to encourage binary outputs. Specifically, for an input variable $s = sigmoid(t) \in [0, 1]$, the piece-wise function is defined as

$$\pi(s) = \begin{cases} 0, & s < 0.5 - \epsilon \\ s, & 0.5 - \epsilon \leq s \leq 0.5 + \epsilon \\ 1, & s > 0.5 + \epsilon, \end{cases}$$

where ϵ is a small positive hyper-parameter. Then, we have the hash function as $f(\mathcal{I}, \mathbf{w}) = [f(\mathcal{I}, \mathbf{w}_1), \ldots, f(\mathcal{I}, \mathbf{w}_r)]$. After the deep architecture is trained, in prediction, the hashing code for a new image $\mathcal{I}$ can be done by a simple quantization $b = sign(f(\mathcal{I}, \mathbf{w}) - 0.5)$, where $sign(v)$ is a sign function on vectors that for $i = 1, 2, \ldots, r$, $sign(v_i) = 1$ if $v_i > 0$, otherwise $sign(v_i) = 0$. Note that for input images in prediction, the proposed deep hashing approach produces approximate (real-value) hash codes, which are converted into binary ones by the quantiation process. Nonetheless, the piece-wise threshold function serves to convert some of the values in the approximate hash codes into zeros or ones in training, e.g., the outputs from the sigmoid function in $[0, 0.5 - \epsilon]$ or $[0.5 + \epsilon, 1]$ are truncated to be 0 or 1, respectively. This can minimise the errors induced by the quantisation step.

Optimizing the Structured Loss

Previous works on person re-identification implement a SGD [64] by drawing pairs or triplets of images uniformly at random. They didn't fully make use of the information of the mini-batch that is sampled at a time and not only individual pairs or triplets. By contrast, we propose a structured loss over a mini-batch in order to take full advantage of the training batches used in SGD. Meanwhile, the proposed structured loss can ensure fast convergence and stableness in training. The structured loss is conducted on all positive pairs and corresponding close ("difficult") negative pairs across camera views. Specifically, it can be formulated as

$$\begin{aligned} F &= \frac{1}{|\bar{P}|} \sum_{\mathbf{x}_i, \mathbf{y}_i \in \bar{P}} \max\left(0, F_{\mathbf{x}_i, \mathbf{y}_i}\right), \\ F_{\mathbf{x}_i, \mathbf{y}_i} &= \max\left(\max\left(0, \alpha - ||\mathbf{x}_i - \mathbf{y}_k||_H\right), \max\left(0, \alpha - ||\mathbf{y}_i - \mathbf{y}_l||_H\right)\right) \\ &+ ||\mathbf{x}_i - \mathbf{y}_i||_H, \ s.t. \ \mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_k, \mathbf{y}_l \in \{0, 1\}^r, (\mathbf{x}_i, \mathbf{y}_i) \in \bar{P}, (\mathbf{y}_k, \mathbf{y}_l) \in \bar{N}, \end{aligned} \tag{6}$$

where $\bar{P}$ and $\bar{N}$ denote the set of positive and negative pairs in each mini-batch. The process of selecting positive and negative samples is elaborated in paragraph 3.1.2.

Relationship to contrastive, triplet and quadruplet [29] ranking loss

- In pairwise training with $O(m)$ separate pairs in the batch, a total of $O(m^2)$ pairs can be generated accordingly. However, these negative edges induced between randomly sampled pairs carry very limited information [51]. By contrast, selected difficult exemplars are sharper cases that a full sub-gradient method would more likely focus on;
- Compared with triplet embedding containing randomly sampled triplets, our training batch is augmented by adding negative neighbours bilaterally for each positive pairs. By doing this, the optimisation process is conducted on most violate constraints, leading to fast convergence. Moreover, the proposed structured loss makes best use of the mini-batch structure to further reduce the intra-class variations and enlarging the inter-class variations by pushing away negative pairs from positive pairs with respect to the different probe images (i.e., $||\mathbf{y}_i - \mathbf{y}_l||_H < ||\mathbf{y}_i - \mathbf{x}_i||_H$). Specifically, the structured loss can be equivalently transformed to be two regular triplets: $[\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_k]$, and $[\mathbf{y}_i, \mathbf{x}_i, \mathbf{y}_l]$, which provide the correct orders of relative comparison in the Hamming embedding space, that is, $\max(||\mathbf{x}_i - \mathbf{y}_i||_H - ||\mathbf{x}_i - \mathbf{y}_k||_H + \alpha, 0) + \max(||\mathbf{y}_i - \mathbf{x}_i||_H - ||\mathbf{y}_i - \mathbf{y}_l||_H + \alpha, 0)$. Thus, the proposed loss in practice leads to improved capacity of the deep hashing model by simultaneously considering the correct orders in Hamming space and reducing (enlarging) intra-class (inter-class) variations.
- Compared with quadruplet constraint that expresses inequality constraints on dissimilarities, our structured loss can be transformed to a rich class of quadruplet constraints. For example, given $(\mathbf{x}_i, \mathbf{y}_i) \in \bar{P}$, $(\mathbf{y}_k, \mathbf{y}_l) \in \bar{N}$, our structured loss in (6) is equivalent to be two sets of quadruplet inequality: $d(\mathbf{x}_i, \mathbf{y}_i) < d(\mathbf{x}_i, \mathbf{y}_k)$, $d(\mathbf{y}_i, \mathbf{x}_i) < d(\mathbf{y}_k, \mathbf{y}_l)$.

For ease of optimizsation, we relax Eq. (6) by replacing the Hamming norm with the ℓ_2-norm and replacing the integer constraints on $\mathbf{x}$'s and $\mathbf{y}$'s with the range constraints. The margin of α is set to be 1. The modified loss function is

$$
\begin{aligned}
F &= \frac{1}{|\bar{P}|} \sum_{\mathbf{x}_i, \mathbf{y}_i \in \bar{P}} \max\left(0, F_{\mathbf{x}_i, \mathbf{y}_i}\right), \\
F_{\mathbf{x}_i, \mathbf{y}_i} &= \max\left(\max\left(0, 1 - ||\mathbf{x}_i - \mathbf{y}_k||_2^2\right), \max\left(0, 1 - ||\mathbf{y}_i - \mathbf{y}_l||_2^2\right)\right) \\
&+ ||\mathbf{x}_i - \mathbf{y}_i||_2^2, \; s.t. \; \mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_k, \mathbf{y}_l \in [0, 1]^r, (\mathbf{x}_i, \mathbf{y}_i) \in \bar{P}, (\mathbf{y}_k, \mathbf{y}_l) \in \bar{N}.
\end{aligned}
\tag{7}
$$

The variant of structured loss is convex. Its sub-gradients with respect to $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{y}_k$, and $\mathbf{y}_l$ are

$$\begin{aligned}
\frac{\partial F}{\partial \mathbf{x}_i} &= (2\mathbf{y}_k - 2\mathbf{y}_i) \times \mathbb{I}[2 + ||\mathbf{x}_i - \mathbf{y}_i||_2^2 > ||\mathbf{x}_i - \mathbf{y}_k||_2^2 + ||\mathbf{y}_i - \mathbf{y}_l||_2^2] \\
\frac{\partial F}{\partial \mathbf{y}_i} &= (2\mathbf{y}_l - 2\mathbf{x}_i) \times \mathbb{I}[2 + ||\mathbf{x}_i - \mathbf{y}_i||_2^2 > ||\mathbf{x}_i - \mathbf{y}_k||_2^2 + ||\mathbf{y}_i - \mathbf{y}_l||_2^2] \\
\frac{\partial F}{\partial \mathbf{y}_k} &= 2\mathbf{x}_i \times \mathbb{I}[2 + ||\mathbf{x}_i - \mathbf{y}_i||_2^2 > ||\mathbf{x}_i - \mathbf{y}_k||_2^2 + ||\mathbf{y}_i - \mathbf{y}_l||_2^2] \\
\frac{\partial F}{\partial \mathbf{y}_l} &= 2\mathbf{y}_i \times \mathbb{I}[2 + ||\mathbf{x}_i - \mathbf{y}_i||_2^2 > ||\mathbf{x}_i - \mathbf{y}_k||_2^2 + ||\mathbf{y}_i - \mathbf{y}_l||_2^2]
\end{aligned} \tag{8}$$

The indicator function $\mathbb{I}[\cdot]$ is the indicator function which outputs 1 if the expression evaluates to true and outputs 0 otherwise. Thus, the loss function in Eq. (6) can be easily integrated into back propagation of neural networks. We can see that our structured loss provides informative gradients signals for all negative pairs which are within the margin of any positive pairs. In contrast to existing networks like [1, 32] where only hardest negative gradients are updated, making the training easily over-fit, the proposed structured loss makes the optimisation much more stable.

Hard Negative Mining for Mini-batches

As aforementioned, our approach differs from existing deep methods by making full information of the mini-batch that is sampled at a time, including positive pairs and their difficult neighbours. Please note that difficult neighbours are defined only with respect to the gallery camera view. The rationale is to enhance the mini-batch optimisation in network training because the sub-gradient of $F_{\mathbf{x}_i,\mathbf{y}_i}$ would use the close negative pairs. Thus, our approach biases the sample towards including "difficult" pairs. In this section, we particularly select a few positive pairs at random, and then **actively** add their difficult (hard) neighbours into the training mini-batch. This augmentation adds relevant information that a sub-gradient would use. Specifically, we determine the elements in mini-batches by online generation where all anchor-positive pairs in any identity are kept while selecting the hard negatives for both the anchor and its positive correspondence. In fact, this procedure of mining hard negative edges amounts to computing the loss augmented inference in structured prediction setting [22, 51, 54]. Intuitively, the loss from hard negative pairs should be penalizsed more heavily than a loss involving other pairs. In this sense, our structured loss function contains enough negative examples within the margin bound, which can push the positive examples towards the correct direction and thus making the optimisation much more stable.

3.1.3 Unsupervised Deep Representation Learning with Generative Models

These deep learning methods are inherently limited due to two presumable assumptions: the availability of large numbered labeled samples across views and the two fixed camera views are supposed to exhibit a unimodal inter-camera transform. In practice, building a training dataset with tuples of labeled corresponding images is impossible for every pair of camera views in the context of a large camera network in video surveillance. Thus, this correspondence dependency greatly limits the applicability of the existing approaches with training samples in correspondence. Secondly, the practical configurations (which are the combinations of view points, poses, lightings, and photometric settings) of pedestrian images are *multi-modal* and view-specific [31] even if they are observed under the same camera. Therefore, the complex yet multi-modal inter-camera variations cannot be well learned with a generic metric which is incapable of handling multiple types of transforms across views. Last but not the least, existing deep learning methodologies directly compute the difference between intermediate CNN features and propagate only the similarity value to a ultimate scalar. This would lose important information since they ignore feature alignment in cross-view.

To overcome these limitations, a crossing net based on a couple of generative adversarial networks (GANs) [18] is proposed to seek effective cross-view representations for person re-id [80]. To combat the first issue of relying on supervision, as shown in Fig. 6, we observe some patterns that appear commonly across image pairs are distinct to discriminate positive pairs from negatives. Thus, these co-occurrence patterns should be mined out automatically to facilitate the task of re-id. Specifically, as shown in Fig. 7, the proposed network starts from a tuple of variational auto-encoder (VAE) [24], each for one image from a camera view, to encode the input images into their respective latent variables without any region-level annotations

Fig. 6 Left: Pedestrian images selected from CUHK03 dataset [32]. Each column indicates images in pairs regarding the same person observed by disjoint camera views. Right: Illustration of co-occurrence regions in positive image pairs

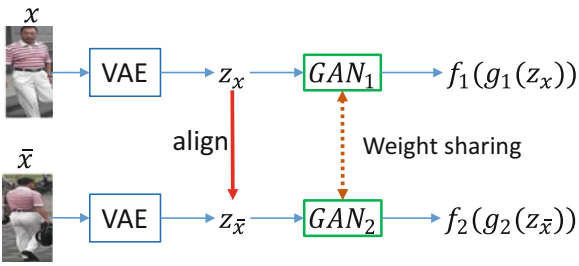


Fig. 7 The schematic overview of the proposed crossing GAN for person re-id

on person images. The technique of VAE has been established a viable solution for image distribution learning tasks while in this chapter, we describe VAE to statistically generate latent variables for paired images without correspondence labelling [80]. It is remarked that it is not desirable to use the Siamese CNNs [55] to encode the input pair because CNNs are composed of fixed receptive fields which may not flexible to capture the varied local patterns. Also, the Siamese architecture enforces the weight sharing across CNN layers which are not suited for multi-modal view-specific variations.

To address the view disparity, a cross-view alignment can be adopted which is bridged over VAE outputs to allow the comparable matching. This alignment operation is to derive a shared latent space by modeling the statistical relationships between generative variables, and it is empirically demonstrated this explicit alignment is crucial for cross-view representation learning [80]. Then, the crossing net is coupled with adversarial networks to produce joint view-invariant distribution which gives a probability function to each joint occurrence of cross-view person images.

Systematic Overview: Crossing GANs for Person Re-identification

The complete network is then trained end-to-end for learning a joint invariant distribution of images across camera views. Figure 8 illustrates the overview of our architecture. It consists of a pair of (VAE, GAN)s, that is, (VAE_1, GAN_1) and (VAE_2, GAN_2); each is responsible for synthesizing one image in one camera view. In Fig. 8, the blue and green routes represent the forward paths of the VAE and GAN for images $\mathbf{x}$ and $\bar{\mathbf{x}}$, respectively. The blue route, i.e., the VAE flow, is the use of expressive latent variables to model the variability observed in the data. It essentially captures the statistics of each individual image. The auto-encoding procedure is explained in paragraph 3.1.3. The red route denotes the cross-view alignment that links the latent variables $(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\bar{\mathbf{x}}})$ to ensure the shared latent representations. The details of alignment is given in paragraph 3.1.3. The green routes represent the adversarial learning which works to optimise optimal latent features corresponding to the joint invariance across paired images. During training, the two GANs are enforced to share a subset of parameters (the brown routes), which results in synthesised pairs of corresponding images without correspondence supervision.

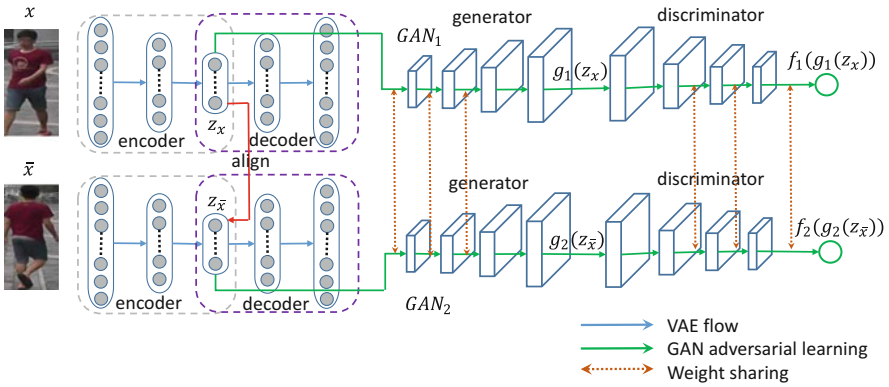


Fig. 8 The overview on Crossing-GANs. Best view in color

Auto-encoding

Given a pair of data points $(\mathbf{x}^{(i)}, \bar{\mathbf{x}}^{(i)})$ from a dataset $\mathbf{X} = \{\mathbf{x}^{(i)}, \bar{\mathbf{x}}^{(i)}\}_{i=1}^{M}$ containing $N = 2M$ samples in M pairs. The auto-encoding algorithm uses unobserved random variable $\mathbf{z}^{(i)}$, to generate a data point $\mathbf{x}^{(i)}$. As the generating process can be repeated on either $\mathbf{x}^{(i)}$ or $\bar{\mathbf{x}}^{(i)}$, in the following, we describe $\mathbf{x}^{(i)}$ as illustration. The process is composed of two phases: (1) a value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p(\mathbf{z}^{(i)})$; (2) a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})$. From a coding theory perspective, the unobserved variable $\mathbf{z}^{(i)}$ have an interpretation as a latent representation or *code*. Following VAE [24] which introduces a recognition model $q(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$: an approximation to the intractable true posterior $p(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$, we will therefore refer to $q(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$ as a probabilistic *encoder*, since given a data point $\mathbf{x}^{(i)}$ its produces a distribution (e.g., a Gaussian) over the possible values of the code $\mathbf{z}^{(i)}$ from which the data point $\mathbf{x}^{(i)}$ could be generated. In a similar vein, we refer to $p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})$ as a probabilistic *decoder*, since given a code $\mathbf{z}^{(i)}$ it produces a distribution over the possible corresponding values of $\mathbf{x}^{(i)}$.

In this framework, neural networks are used as probabilistic encoders and decoders, namely multi-layered perceptions (MLPs). Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \boldsymbol{I})$ whose distribution parameters are computed from $\mathbf{z}$ with a MLP. We assume the true posterior $p(\mathbf{z}|\mathbf{x})$ takes on an approximate Gaussian form with an approximately diagonal covariance. In this case, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure:

$$\log q(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I}) \tag{9}$$

where the mean and standard of the approximate posterior, $\boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{(i)}$ are outputs of the encoding MLP. i.e., nonlinear functions of data point $\mathbf{x}^{(i)}$ and the variational parameters.

Specifically, we sample from the posterior $\mathbf{z}^{(i)} \sim q(\mathbf{z}|\mathbf{x}^{(i)})$ using $\mathbf{z}^{(i)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. With $\odot$ we signify an element-wise product. In this model, both $p(\mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$ are Gaussian. The resulting estimator loss for data point $\mathbf{x}^{(i)}$ is:

$$\begin{aligned}
&\tilde{L}_{vae}(\mathbf{x}^{(i)}) = D_{KL}(q(\mathbf{z}|\mathbf{x}^{(i)}||p(\mathbf{z}))) - \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})}[\log p(\mathbf{x}^{(i)}|\mathbf{z})] \\
&= D_{KL}(q(\mathbf{z}|\mathbf{x}^{(i)}||p(\mathbf{z}))) - \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}) \\
&\simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) - \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}) \\
&\mathbf{z}^{(i)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})
\end{aligned} \tag{10}$$

where the KL-divergence $D_{KL}(q(\mathbf{z}|\mathbf{x}^{(i)}||p(\mathbf{z})))$ can be integrated analytically, such that only the expected reconstruction error $\mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})}[\log p(\mathbf{x}^{(i)}|\mathbf{z})]$ requires estimation by sampling. Given multiple data points from a dataset $\mathbf{x}$ with M pairs of data points, we can construct an estimator loss of as follows:

$$\tilde{L}_{vae}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\left(\tilde{L}_{vae}(\mathbf{x}^{(i)}) + \tilde{L}_{vae}(\bar{\mathbf{x}}^{(i)})\right). \tag{11}$$

Learning Cross-View Alignment on Latent Codes

In this paragraph, we introduce cross-view alignment over latent representations provided by VAE, which is capable of modeling complex multi-modal distributions over data space. Note that for notation convenience, we use $\mathbf{z}_{\mathbf{x}}$ and $\mathbf{z}_{\bar{\mathbf{x}}}$ to distinguish the latent representation for $\mathbf{x}$ and $\bar{\mathbf{x}}$.

$$\mathcal{L}_{align} = \max(||\mathbf{z}_{\mathbf{x}} - \text{ Align}(\mathbf{z}_{\bar{\mathbf{x}}})||^2, \tau), \tag{12}$$

where we model Align$(\cdot)$ as a single fully connected neuron with a *tanh* activation function. The threshold τ is $\tau = 1$. In essence, Align$(\cdot)$ is implicitly learning a mapping across two normal distributions $(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\bar{\mathbf{x}}})$. The parameters of the mapping $\boldsymbol{\theta}_{Align}$ are optimized through back-propagation. Since both the VAE and the GAN are able to learn low-dimensional representations (in our case, both $\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\bar{\mathbf{x}}}$ are set to be 100 dimensions.), we are able to fit the cross-view alignment with moderate pairs. The strategy of alignment is designed to align the transformation across cameras by revealing underlying invariant properties among different views. As a result, unsupervised matching pedestrian images can be statistically inferred through aligned latent representations. This is motivated by the observation that some regions are distributed similarly in images across views and robustly maintain their appearance in the presence of large cross-view variations.

Adversarial Learning

Generator Let g_1 and g_2 be the generators of GAN_1 and GAN_2, which map corresponding inputs $\mathbf{z_x}$ and $\mathbf{z_{\bar{x}}}$ to images that have the same support as $\mathbf{x}$ and $\bar{\mathbf{x}}$, respectively. Both g_1 and g_2 are realized as convolutions [46]:

$$\begin{aligned} g_1(\mathbf{z_x}) &= g_1^{(m)}(g_1^{(m-1)}(\dots g_1^{(2)}(g_1^{(1)}(\mathbf{z_x})))), \\ g_2(\mathbf{z_{\bar{x}}}) &= g_2^{(m)}(g_2^{(m-1)}(\dots g_2^{(2)}(g_2^{(1)}(\mathbf{z_{\bar{x}}})))); \end{aligned} \tag{13}$$

where $g_1^{(i)}$ and $g_2^{(i)}$ are the i-th layer of g_1 and g_2 and m is the number of layers in generators. Through layers of convolution operations, the generator gradually decode information from more abstract concept to more material details. The first layer decode high-level semantics while the last layer decode low-level details. Note this information flow is opposite to that in a standard deep neural network [27] where the first layers extract low-level features while the last layers extract high-level features. Based on the observation that a pair of person images from two camera views share the same high-level concept (i.e., they belong to the same identity but with different visual appearance), we enforce the first layers of g_1 and g_2 to have identical structures and share the weights, which means $\boldsymbol{\theta}_{g_1^{(i)}} = \boldsymbol{\theta}_{g_2^{(i)}}$, for $i = 1, 2, \dots, k$ where k is the number of shared layers, and $\theta_{g_1^{(i)}}$ and $\theta_{g_2^{(i)}}$ are the parameters of $g_1^{(i)}$ and $g_2^{(i)}$, respectively. This constraint can force the high-level semantics to be decoded in the same way in g_1 and g_2, which can also be propagated into the VAE to update the parameters simultaneously. Thus, the generator can gradually decode the information from more abstract concepts to more finer details, and the view-alignment is embedded to ensure the common finer regions can be preserved with high correlations.

Discriminator Let f_1 and f_2 be the discriminators of GAN_1 and GAN_2 given by

$$\begin{aligned} f_1(\mathbf{x}) &= f_1^{(n)}(f_1^{(n-1)}(\dots f_1^{(2)}(f_1^{(1)}(\mathbf{x})))), \\ f_2(\bar{\mathbf{x}}) &= f_2^{(n)}(f_2^{(n-1)}(\dots f_2^{(2)}(f_2^{(1)}(\bar{\mathbf{x}})))); \end{aligned} \tag{14}$$

where $f_1^{(i)}$ and $f_2^{(i)}$ are the i-th layer of f_1 and f_2, and n is the number of layers. Note that GAN_1 and GAN_2 have the identical network structure. The discriminator maps an input image to a probability score, estimating the likelihood that the input is drawn from a true data distribution. The first layers of the discriminator extract low-level features while the last layers of layers extract high-level features. Considering that input image pair are realisations of the same person in two camera views, we force f_1 and f_2 to have the same last layers, which is achieved by sharing the weights of the last layers via $\boldsymbol{\theta}_{f_1^{(n-i)}} = \boldsymbol{\theta}_{f_2^{(n-i)}}$, for $i = 0, 1, \dots, l-1$ where l is the number of weight-sharing layers in the discriminator, and $\boldsymbol{\theta}_{f_1^{(i)}}$ and $\boldsymbol{\theta}_{f_2^{(i)}}$ are the network parameters of $f_1^{(i)}$ and $f_2^{(i)}$,

respectively. The weight-sharing constraints herein helps reduce the number of trainable parameters of the network, and also effective in deriving view-invariant features in joint distribution across $\mathbf{x}$ and $\bar{\mathbf{x}}$.

Therefore, we cast the problem of learning jointly invariant feature distribution as a constrained objective function with the training loss given by

$$\begin{aligned} \mathcal{L}_{gan}(f_1, f_2, g_1, g_2) &= \log f_1(\mathbf{x}) + \log(1 - f_1(g_1(\mathbf{z}))) \\ &+ \log f_2(\bar{\mathbf{x}}) + \log(1 - f_2(g_2(\mathbf{z}_{\bar{\mathbf{x}}}))) \\ \text{subject to } \boldsymbol{\theta}_{g_1^{(j)}} &= \boldsymbol{\theta}_{g_2^{(j)}},\ j = 1, 2, \ldots, k \\ \boldsymbol{\theta}_{f_1^{(n-i)}} &= \boldsymbol{\theta}_{f_2^{(n-i)}},\ i = 0, 1, \ldots, l-1 \end{aligned} \tag{15}$$

The crossing GAN can be interpreted as min-max game with two teams and each team has two players.

3.2 Video-Based Person Re-identification

An image sequence can be viewed as a 3-dim space-time volume and space-time features can be extracted based on space-time interest points [48]. In person re-id, some works focus on motion features to describe the appearance variations of pedestrians. For instance, Wang et al. [61] employ the HOG3D [25] descriptor with dense sampling after identifying walking periodicity. However, a pre-processing is needed to select discriminative video fragments from which HOG3D features can be extracted. Another work from Liu et al. is to extract 3D low-level features that encode both spatially and temporal aligned appearance of a pedestrian. They also need to manually align video fragments by using regulated flow energy profile and spatial alignment is achieved by partitioning human body into six rectangles to describe different body parts. More recently, deep neural networks have been demonstrated to be effective for image based person re-id and have achieved notable improved performance [9, 55–57, 67, 68, 75, 84, 85]. In contrast, video-based re-id is less paid attention and only a few approaches are developed to address this challenge. McLaughlin et al. [39] extract CNN features for each frame, and put through a RNN to impose their temporal dependency. In [65], an end-to-end approach is proposed to simultaneously learn deep features and their similarity metric for a pair of videos as input. Also Yan et al. propose a recurrent feature aggregation network (RFA-Net) [78] which shares the similar idea of using RNN to aggregate frame-wise features into a sequence level representation. Another effective approach [94] is to use two RNNs where one is to exploit the temporal dimension and the other is to calculate the spatially context similarity of two corresponding locations in a pair of video sequences. Similar to [94], an attentive spatial-temporal pooling is presented [77] to employ spatial pooling to select regions from frame while temporal pooling is used to select informative frames.

To go beyond individual image-level appearance information and exploit the temporal information, Simonyan et al. [15] proposed the two-stream architecture, cohorts of spatial and temporal ConvNets. The input to the spatial and temporal networks are RGB frames and stacks of multiple-frame dense optical flow fields, respectively. Wang et al. [59] further improve the two-stream architecture by enforcing consensus over predictions in individual frames These networks are still limited in its capacity to capture temporal information, because they operate on a fixed number of regularly spaced, single frames from the entire video. Another recent work of ActionVLAD [16] extends the NetVLAD aggregation layer [2] to video and shows that aggregating the last layers of convolutional descriptors regarding spatial and temporal networks performs better. Inspired by these two-stream architectures, a two stream Siamese convolutional neural network architecture [11] is proposed to use two CNNs to process spatial and temporal information separately, and the outputs of two streams are fused together using a weighted cost function. However, these deep learning based methods commonly employ 2D ConvNets to extract feature activations for each frame which are leveraged into RNNs to ensure the temporal dependency among frames. In this way, temporal priors are completely collapsed right after convolutions. In contrast to these methods, we develop an end-to-end trainable video architecture that combines the recent advances in 3D ConvNets with a trainable spatiotemporal body part alignment based VLAD aggregation layer. This is to our best knowledge, has not be done before.

Recent video-based re-id methods have focused on learning deep features by considering both spatial and temporal information. The neural networks for this purpose can be forked into two types: *one-stream ConvNets* [39, 65, 78], and *two-stream ConvNets* [11, 69, 71, 77, 94]. As to the one-stream ConvNets, spatial networks perform appearance modelling from individual frames, and the appearance features are in combination of recurrence to aggregate the frame-wise representation at each time step to yield sequence-level representation. However, such image based deep features are not suitable for videos due to lack of motion modeling, and more critically 2D ConvNets lose temporal priors of the input signal right after the convolutions. The two-stream ConvNets are demonstrated to outperform the one-stream ConvNets. They decompose the video into motion and appearance streams, and train separate CNNs for each stream, fusing the outputs in the end. However, two-stream architectures largely disregard the long-term temporal structure of the video and essentially learn a classifier that operates on individual frames or short block of few frames [39]. In other words, they may enforce the consensus of classification scores over different segments of a video. This approach would raise the question whether such temporal averaging is capable of modelling the complex spatiotemporal structure of a person video. And this problem becomes exaggerated when some persons have large intra-class variability caused by the changes of viewpoints, illuminations and human poses. Suppose only a few consecutive frames of a person video are given, it can be easily confused with other persons who exhibit similar visual appearance, for example, persons who often wear dark jackets and pants. In this end, what is needed in a reliable person re-id system is a *global*

evidence over the entire video about both the appearance and the motion of the individuals without requiring every frame to be assigned to a single person identity.

4 Conclusions

Developing a security model for CPS is comparatively less-studied with limited work being accomplished in the field. Considering that the CPS is composed of three layers, a more efficient approach for dealing with security in CPS is by using a multi-layered approach where the security is considered at the beginning of the design for each layer. However, any solution should consider the application situation and context as part of assessing security tasks. Thus, enhancing the application security will improve the security of the whole system, and a security mechanism should be designed for the entire system rather than in a single layer.

This chapter has presented several fundamental yet effective models in person re-id which are primarily based on convolutional neural networks. These methods provide visual features that perform well in person re-id under the complex appearance variations both in static images and videos. Typical deep learning methods towards person re-id can be categorised into supervised and unsupervised streams, depending on the requirement of label annotations in their training procedure.

5 Open Problems

Problem 1

Deep learning models in the variants of convolutional neural networks and recurrent neural networks have been extensively studied to show their great promise in person re-id. However, the performance gain cannot be achieved further by these models due to the limitation to access larger datasets, and thus the generalisation of learned deep models are somewhat poor. The generalisation of deep models in person re-id remains a problem to be addressed which is attracting more attention from the computer vision community.

Problem 2

One promising direction in person re-id is to seek a plausible combination with machine learning paradigms such as learning to rank and few-shot learning. There are little efforts in person re-id literature to carry out the effective framework built on the basis of machine learning progresses. In this end, working out this sort of combination is unresolved yet a difficult challenge.

References

1. Ahmed E, Jones M, Marks TK (2015) An improved deep learning architecture for person re-identification. In: CVPR
2. Arandjelovic R, Gronat P, Torii A, Pajdla T, Sivic J (2016) NetVLAD: CNN architecture for weakly supervised place recognition. In: Computer Vision and Pattern Recognition
3. Ashibani Y, Mahmoud QH (2017) Cyber physical systems security: analysis, challenges and solutions. Comput Secur 68:81–97
4. Bak S, Carr P (2017) One-shot metric learning for person re-identification. In: CVPR
5. Bazzani L, Cristani M, Perina A, Murino V (2012) Multiple-shot person re-identification by chromatic and epitomic analyses. Pattern Recogn 33(7):898–903
6. Bhabad MA, Scholar P (2015) Internet of things: architecture, security issues and countermeasure. Int J Comput Appl 125(4):1–4
7. Chen D, Yuan Z, Chen B, Zhang N (2016) Similarity learning with spatial constraints for person re-identification. In: CVPR, pp 1268–1277
8. Chen SZ, Guo CC, Lai JH (2016) Deep ranking for re-identification via joint representation learning. IEEE Trans Image Process 25(5):2353–2367
9. Cheng D, Gong Y, Zhou S, Wang J, Zhang N (2016) Person re-identification by multi-channel parts-based CNN with improved triplet loss function. In: CVPR, pp 1335–1344
10. Cho YJ, Yoon KJ (2016) Improving person re-identification via pose-aware multi-shot matching. In: IEEE Conference on Computer Vision and Pattern Recognition
11. Chung D, Tahboub K, Delp EJ (2017) A two stream siamese convolutional neural network for person re-identification. In: International Conference on Computer Vision
12. Davis JV, Kulis B, Jain P, Sra S, Dhillon IS (2007) Information-theoretic metric learning. In: ICML
13. Fan H, Zheng L, Yan C, Yang Y (2018) Unsupervised person re-identification: clustering and fine-tuning. ACM Trans Multimed Comput Commun Appl 14(4):Article 83:1–18
14. Farenzena M, Bazzani L, Perina A, Murino V, Cristani M (2010) Person re-identification by symmetry-driven accumulation of local features. In: CVPR
15. Feichtenhofer C, Pinz A, Zisserman A (2016) Convolutional two-stream network fusion for video action recognition. In: CVPR
16. Girdhar R, Ramanan D, Gupta A, Sivic J, Russell B (2017) Actionvlad: learning spatio-temporal aggregation for action classification. In: Computer Vision and Pattern Recognition
17. Gong S, Christani M, Loy CC, Hospedales TM (2014) Person re-identification. Springer, London
18. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: NIPS
19. Gray D, Brennan S, Tao H (2007) Evaluating appearance models for recognition, reacquisition, and tracking. In: Proceedings of International Workshop on Performance Evaluation for Tracking and Surveillance
20. Gray D, Tao H (2008) Viewpoint invariant pedestrian recognition with an ensemble of localized features. In: ECCV
21. Guillaumin M, Verbeek J, Schmid C (2009) Is that you? Metric learning approaches for face identification. In: ICCV
22. Joachims T, Finley T, Yu CNJ (2009) Cutting-plance training of structural SVMS. J Mach Learn Res 77:27–59
23. Kedem D, Tyree S, Sha F, Lanckriet GR, Weinberger KQ (2012) Non-linear metric learning. In: NIPS
24. Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: ICLR
25. Klaser A, Marszaek M, Shmid C (2008) A spatio-temporal descriptor based on 3d-gradients. In: British Machine Vision Conference
26. Kostinger M, Hirzer M, Wohlhart P, Roth PM, Bischof H (2012) Large scale metric learning from equivalence constraints. In: CVPR

27. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems
28. Lai H, Pan Y, Liu Y, Yan S (2015) Simultaneous feature learning and hash coding with deep neural networks. In: CVPR
29. Law MT, Thome N, Cord M (2013) Quadruplet-wise image similarity learning. In: ICCV
30. Li D, Chen X, Zhang Z, Huang K (2017) Learning deep context-aware features over body and latent parts for person re-identification. In: CVPR
31. Li W, Wang X (2013) Locally alligned feature transforms across views. In: CVPR
32. Li W, Zhao R, Tang X, Wang X (2014) Deepreid: Deep filter pairing neural network for person re-identification. In: CVPR
33. Li W, Zhu X, Gong S (2017) Person re-identification by deep joint learning of multi-loss classification. In: IJCAI
34. Li Z, Chang S, Liang F, Huang TS, Cao L, Smith J (2013) Learning locally-adaptive decision functions for person verification. In: CVPR
35. Liao S, Hu Y, Zhu X, Li SZ (2015) Person re-identification by local maximal occurrence representation and metric learning. In: CVPR, pp 2197–2206
36. Liao S, Li SZ (2015) Efficient psd constrained asymmetric metric learning for person re-identification. In: ICCV
37. Lisanti G, Masi I, Del Bimbo A, Bagdanov, AD (2015) Person re-identification by iterative re-weighted sparse ranking. IEEE Trans Pattern Anal Mach Intell 37(8):1629–1642
38. McFee B, Lanckriet GRG (2010) Metric learning to rank. In: ICML
39. McLaughlin N, del Rincon JM, Miller P (2016) Recurrent convolutional network for video-based person re-identification. In: CVPR
40. Mignon A, Jurie F (2012) PCCA: a new approach for distance learning from sparse pairwise constraints. In: CVPR, pp 2666–2672
41. Ouyang W, Wang X (2013) Joint deep learning for pedestrian detection. In: ICCV
42. Paisitkriangkrai S, Shen C, van den Hengel A (2015) Learning to rank in person re-identification with metric ensembles. In: CVPR
43. Pedagadi S, Orwell J, Velastin S, Boghossian B (2013) Local fisher discriminant analysis for pedestrian re-identification. In: CVPR
44. Peng P, Xiang T, Wang Y, Pontil M, Gong S, Huang T, Tian Y (2016) Unsupervised cross-dataset transfer learning for person re-identification. In: CVPR
45. Prosser B, Zheng WS, Gong S, Xiang T, Mary Q (2010) Person re-identification by support vector ranking. In: BMVC
46. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434
47. Schwartz W, Davis L (2009) Learning discriminative appearance-based models using partial least squares. In: Proceedings of SIBGRAPI
48. Scovanner P, Ali S, Shah M (2007) A 3-dimensional sift descriptor and its application to action recognition. In: ACM Multimedia
49. Shi H, Yang Y, Zhu X, Liao S, Lei Z, Zheng W, Li SZ (2016) Embedding deep metric for person re-identification: a study against large variations. In: ECCV, pp 732–748
50. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
51. Song HO, Xiang Y, Jegelka S, Savarese S (2016) Deep metric learning via lifted structured feature embedding. In: CVPR
52. Sun Y, Wang X, Tang X (2014) Deep learning face representation from predicting 10,000 classes. In: CVPR
53. Sun Y, Zheng L, Deng W, Wang S (2017) SVDnet for pedestrian retrieval. In: ICCV
54. Tsochantaridis I, Hofman T, Joachims T, Altun Y (2004) Support vector machine learning for interdependent and structured output spaces. In: ICML
55. Varior RR, Haloi M, Wang G (2016) Gated siamese convolutional neural network architecture for human re-identification. In: ECCV, pp 791–808

56. Varior RR, Shuai B, Lu J, Xu D, Wang G (2016) A siamese long short-term memory architecture for human re-identification. In: ECCV, pp 135–153
57. Wang F, Zuo W, Lin L, Zhang D, Zhang L (2016) Joint learning of single-image and cross-image representations for person re-identification. In: CVPR, pp 1288–1296
58. Wang H, Gong S, Xiang T (2014) Unsupervised learning of generative topic saliency for person re-identification. In: BMVC
59. Wang L, Xiong Y, Wang Z, Qiao Y, Lin D, Tang X, Gool LV (2016) Temporal segment networks: towards good practices for deep action recognition. In: ECCV
60. Wang N, Yeung D (2013) Learning a deep compact image representation for visual tracking. In: NIPS
61. Wang T, Gong S, Zhu X, Wang S (2014) Person re-identification by video ranking. In: ECCV
62. Wang X, Doretto G, Sebastian T, Rittscher J, Tu P (2007) Shape and appearance context modeling. In: ICCV
63. Weinberger K, Blitzer J, Saul L (2006) Distance metric learning for large margin nearest neighbor classification. In: NIPS
64. Wilson D, Martinez T (2003) The general inefficiency of batch training for gradient decent learning. Neural Netw 16(10):1429–1451
65. Wu L, Shen C, van den Hengel A (2016) Deep recurrent convolutional networks for video-based person re-identification: an end-to-end approach. In: arXiv: 1606.01609
66. Wu L, Shen C, van den Hengel A (2016) Personnet: Person re-identification with deep convolutional neural networks. In: CoRR abs/1601.07255
67. Wu L, Shen C, van den Hengel A (2017) Deep linear discriminant analysis on fisher networks: a hybrid architecture for person re-identification. Pattern Recogn 65:238–250
68. Wu L, Wang Y, Gao J, Li X (2018) Deep adaptive feature embedding with local sample distributions for person re-identification. Pattern Recogn 73:275–288
69. Wu L, Wang Y, Gao J, Li X (2018) What-and-where to look: deep siamese attention networks for video-based person re-identification. IEEE Trans Multimedia. https://doi.org/10.1109/TMM.2018.2877886
70. Wu L, Wang Y, Ge Z, Hu Q, Li X (2018) Structured deep hashing with convolutional neural networks for fast person re-identification. Comput Vis Image Underst 167:63–73
71. Wu L, Wang Y, Li X, Gao J (2018) Deep attention-based spatially recursive networks for fine-grained visual recognition. IEEE Trans Cybern 99:1–12
72. Wu L, Wang Y, Li X, Gao J (2018) What-and-where to match: deep spatially multiplicative integration networks for person re-identification. Pattern Recogn 76:727–738
73. Wu L, Wang Y, Shao L (2019) Cycle-consistent deep generative hashing for cross-modal retrieval. IEEE Trans Image Process 28(4):1602–1612
74. Wu Y, Mukunoki M, Funatomi T, Minoh M, Lao S (2011) Optimizing mean reciprocal rank for person re-identification. In: Advanced Video and Signal-Based Surveillance
75. Xiao T, Li H, Ouyang W (2016) Learning deep feature representations with domain guided dropout for person re-identification. In: CVPR, pp 1249–1258
76. Xiong F, Gou M, Camps O, Sznaier M (2014) Person re-identification using kernel-based metric learning methods. In: ECCV
77. Xu S, Cheng Y, Gu K, Yang Y, Chang S, Zhou P (2017) Jointly attentive spatial-temporal pooling networks for video-based person re-identification. In: ICCV
78. Yan Y, Ni B, Song Z, Ma C, Yan Y, Yang X (2016) Person re-identification via recurrent feature aggregation. In: ECCV
79. Yu HX, Wu A, Zheng WS (2017) Cross-view asymmetric metric learning for unsupervised person re-identification. In: ICCV
80. Zhang C, Wu L, Wang Y (2018) Crossing generative adversarial networks for cross-view person re-identification. In: arXiv:1801.01760
81. Zhang L, Xiang T, Gong S (2016) Learning a discriminative null space for person re-identification. In: CVPR
82. Zhang R, Lin L, Zuo W, Zhang L (2015) Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE Trans Image Process 24(12):4766–4779

83. Zhao F, Huang Y, Wang L, Tan T (2015) Deep semantic ranking based hashing for multi-label image retrieval. In: CVPR
84. Zhao H, Tian M, Sun S, Shao J, Yan J, Yi S, Wang X, Tang X (2017) Spindle net: person re-identification with human body region guided feature decomposition and fusion. In: CVPR
85. Zhao L, Li X, Zhuang Y, Wang J (2017) Deeply-learned part-aligned representations for person re-identification. In: ICCV
86. Zhao R, Ouyang W, Wang X (2013) Person re-identification by salience matching. In: ICCV
87. Zhao R, Ouyang W, Wang X (2013) Unsupervised salience learning for person re-identification. In: CVPR
88. Zhao R, Ouyang W, Wang X (2014) Learning mid-level filters for person re-identification. In: CVPR
89. Zheng L, Huang Y, Lu H, Yang Y (2017) Pose invariant embedding for deep person re-identification. arXiv:1701.07732
90. Zheng L, Shen L, Tian L, Wang S, Wang J, Tian Q (2015) Scalable person re-identification: a benchmark. In: ICCV
91. Zheng WS, Gong S, Xiang T (2011) Person re-identification by probabilistic relative distance comparison. In: CVPR
92. Zheng WS, Gong S, Xiang T (2016) Towards open-world person re-identification by one-shot group-based verification. TPAMI 38(3), 591–606
93. Zheng X, Ouyang W, Wang X (2013) Multi-stage contextual deep learning for pedestrian detection. In: ICCV
94. Zhou Z, Huang Y, Wang W, Wang L, Tan T (2017) See the forest for the trees: joint spatial and temporal recurrent neural networks for video-based person re-identification. In: CVPR

Deep Learning-Based Detection of Electricity Theft Cyber-Attacks in Smart Grid AMI Networks

Mahmoud Nabil, Muhammad Ismail, Mohamed Mahmoud, Mostafa Shahin, Khalid Qaraqe, and Erchin Serpedin

Abstract Advanced metering infrastructure (AMI) is the primary step to establish a modern smart grid. AMI enables a flexible two-way communication between smart meters and utility company for monitoring and billing purposes. However, AMI suffers from the deceptive behavior of malicious consumers who report false electricity usage in order to reduce their bills, which is known as electricity theft cyber-attacks. In this chapter, we present deep learning-based detectors that can efficiently thwart electricity theft cyber-attacks in smart grid AMI networks. First, we present a customer-specific detector based on a deep feed-forward and recurrent neural networks (RNN). Then, we develop generalized electricity theft detectors that are more robust against contamination attacks compared with customer-specific detectors. In all detectors, optimization of hyperparameters is investigated to improve the performance of the developed detectors. In particular, the hyperparameters of the detectors are optimized via sequential, random, and genetic optimization-based grid search approaches. Extensive test studies are carried out against real energy consumption data to investigate all detectors performance. Also, the performance of the developed deep learning-based detectors is compared with a shallow machine learning approach and a superior performance is observed for the deep learning-based detectors.

M. Nabil (✉) · M. Mahmoud
Department of Electrical and Computer Engineering, Tennessee Technical University, Cookeville, TN, USA
e-mail: mnmahmoud42@students.tntech.edu; mmahmoud@tntech.edu

M. Ismail · K. Qaraqe · E. Serpedin
Electrical and Computer Engineering Department, Texas A&M University at Qatar, Doha, Qatar
e-mail: m.ismail@qatar.tamu.edu; khalid.qaraqe@qatar.tamu.edu; eserpedin@tamu.edu

M. Shahin
School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW, Australia
e-mail: mostafa_shahin@ieee.org

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_4

Keywords Electricity theft · Cyber-attacks · AMI networks · Deep machine learning

1 Introduction

Power grid undergoes two sources of electricity losses known as technical losses and non-technical losses (NTL). Technical losses regularly result from the power dissipated naturally through heat and irradiation during the process of transmitting and distributing the electricity through power lines from the generation centers to the end users (customers). Technical losses are inevitable, and their cost can be approximately estimated [1]. On the other hand, NTLs are hard to be precisely estimated and result from different reasons. One reason for NTLs is calibration errors occurs in meters which result in a billing mistake. Another reason is the amount of consumed electricity that is not billed for. However, the most substantial reason that affects the performance of the power grid is the losses due to improper regulation and outlawed manipulation of meters. Due to electricity theft, NTLs impose unendurable outcomes for both developing and developed economies. For developing countries, the total NTLs cost $46 million in Jamaica in 2013, which estimates as 18% of the country energy consumption [2]. India loses $17 billion every year due to NTLs [3, 4]. Other developing countries lose almost 50% of their electricity revenue due to theft [5]. Similarly, developed countries experience a considerable amount of NTLs. However, these losses have fewer consequences on non-developed economics. In the United States, the NTLs cost is $6 billion/year [3] while in the United Kingdom NTLs cost $173 million [6]. In Canada, there is a loss of about $100 million per year [7].

The smart grid (SG) revolution aims to upgrade and replace the old electricity supply chain by integrating, automating and optimizing the interaction between generators, suppliers, and the customers to allow sustainable and quick demand and response behaviour. SG uses Smart meters (SMs) deployed at the customer side to enable fine-grained data collection for customer's consumption, in addition to promoting customer engagement regarding dynamic pricing, billing and load management. This approach has the potential to hinder traditional physical electricity theft attacks such as line hooking or meter tampering [8]. Nevertheless, The massive deployment of SMs at the customer premises emerges new cyber-attack surface that targets software vulnerabilities in meters [9]. A cyber-attack aims to fraudulently manipulate the SM real consumption and reporting fabricated readings to the utility to reduce the customer's electricity bill, which can negatively affect the whole SG system. These attacks jeopardize the integrity of the customer's energy consumption data as malicious customers hack into their SMs to manipulate (reduce) their own energy consumption values. Despite the difficulties associated with detecting such cyber-attacks, the customers' fine-grained energy consumption data is a promising tool that can boost automated electricity theft detection mechanisms.

Detection of cyber-attacks has been gaining increasing interest recently in today's power engineering research. Generally, solutions for cyber-attack detection can be classified into two main categories: hardware solutions and software solutions. Hardware solutions require the installation and deployment of specific metering hardware such as tempering sensors or Radio-frequency identification (RFID) chips. Despite the reliability of these solutions in detecting different kinds of cyber-attacks, a high cost is associated with the required hardware. [2]. On the other hand, software solutions aim to make a tradeoff between the installation cost and the detection rate of the solution. Several software solutions for electricity theft detection have been proposed in literature [10–22].

Currently, machine learning research exhibits a significant advancement in different applications due to the high availability of data and computing power. Similarly, for electricity theft detection, machine learning-based solutions show a superior detection rate on real datasets [3]. However, the existing research suffers from several limitations. First, few existing detectors employ deep learning architectures, while shallow machine learning architectures such as support vector machines (SVMs) are used extensively. Intuitively, deep learning architectures can better capture the behaviour of the input data resulting in a higher detection rate. Besides, the existing detectors ignore the temporal correlation within the time-series energy consumption data of the customers. Using this correlation can further enhance the detection performance. Moreover, some of the existing works adopt customer-specific models [3], where a detector's model is developed for each customer using only its energy consumption data in order to detect any future thefts. However, the feasibility of customer-specific models is controversial for different reasons. On one hand, customer-specific models require training new detector for every new customer joins the system due to the lack of historical energy consumption records for this customer. In addition, customer-specific detectors are vulnerable to contamination attacks, where the customer data that is used to train the detector might be malicious yet falsely labelled as honest data to confuse the detector.

In this chapter, we address three fundamental questions posed while building any deep neural network-based theft detector.

- First, should we build a customer-specific or generalized detectors? The trade-off is between the accuracy versus the robustness and the practicality. Since customer-specific detector is more accurate as it is trained on a specific customer's data and hence there is less confusion, yet it is not robust against contamination attacks and impractical for newly coming users.
- Second, for the architecture of the detector should it be a simple feed-forward or recurrent neural network (RNN)? The tradeoff is between the optimality versus complexity. RNNs offer a considerable leverage while working with time series data since it is represented as a feedback network where the output at the current state is the input for the next state, yet are more complex than feed-forward models.

- Third, which algorithm is better for hyper-parameters tunning: sequential, random, or genetic-based optimization? Again, the tradeoff is between the optimality versus complexity. Genetic-based optimization for hyper-parameters tunning employs a smart and adaptive searching heuristic based on natural selection, however, it is far complicated than sequential and random search methods.

To answer the aforementioned questions, we present deep learning-based detectors that can efficiently thwart electricity theft cyber-attacks in AMI networks. First, we implement customer-specific detectors based on a deep feed-forward and RNNs. Then, we develop generalized electricity theft detectors that are more robust against contamination attacks compared with the customer-specific detector. Furthermore, we investigate the optimization of the detectors via sequential, random, and genetic optimization-based grid search approaches. All the developed detectors are tested against real energy consumption data and a comprehensive set of cyber-attacks for performance evaluation. Also, the developed deep learning-based detectors performance is compared with a shallow machine learning approach, and a superior performance is observed for the deep learning-based detectors.

The remainder of this chapter is organized as follows: a review of the existing literature is presented in Sect. 2. Dataset used and cyber-attacks injection are presented in Sects. 3.1 and 3.2 respectively. The proposed customer-specific and generalized detectors architectures are explained in Sect. 4. Hyper-parameters tuning schemes are explained in Sect. 5. Numerical results are discussed in Sect. 6. Finally, conclusions and future research are drawn in Sect. 7.

2 Review of State-of-the-Art Detection Techniques

Several studies have addressed the problem of NTLs due to electricity thefts. These studies can be roughly categorized into hardware-based solutions [23–27] and software-based solutions [10–22, 28]. In the following subsections, we will give an overview for these solutions by discussing advantages and disadvantages of both categories.

2.1 Hardware-Based Solutions

Hardware solutions either require the installation of specific metering hardware such as tampering sensors and RFID chips or change the whole infrastructure of the AMI network such as equipment at customers premises, distribution centers and substations.

In [23], a solution based on ARM-Cortex M3 processor is proposed to protect SMs against phase tampering. In case of a detection of any malicious activity, a text message is sent automatically to the utility server. An anti-tampering algorithm is

proposed in [24] which uses ADE7953 chip [29]. This chip can detect any voltage irregularity to the connected load, hence, reporting and classifying different types of tampering behaviours. The experiments suggest that a 7% voltage difference is an acceptable threshold for tampering events detection. In [25], the use of RFID technology is evaluated using a case study on a Chinese utility company, and the Return Over Investment (ROI) is calculated. RFID technology replaces the lead stealing that locks the SM's cover, which can easily be replaced or destroyed, by unique RFID tag that cannot be manipulated. An electricity measurement device is proposed in [26]. The device continuously measures and compares the electric current before, and after the physical connections to the SM, consequently, any inconsistency in both measurements indicates electricity theft. A prototype of the system is tested in Brazil to verify the feasibility of the solution.

A wireless architecture for the power grid is proposed in [27]. The architecture aims to upgrade and optimize the power grid in India. It can also overcome other problems such as electricity theft and manual billing. The solution incorporates a wireless sensor network in the operation of the SG, where a set of sensors is associated with the grid such as transformers, controlling stations and the customers. Despite the efficiency of the solution, there is no cost analysis associated with it. The main disadvantages of these hardware solutions are as following:

1. The installation and deployment cost of the required equipment. In addition to the large cost needed in case of changing the infrastructure [27].
2. The maintainability issues associated with hardware such as failures, battery replacement, etc.
3. The hassle of integrating some solutions with the existing power grid.

2.2 Software-Based Solutions

Recently, software-based solutions have drawn significant attention due to their superior performance and low cost. Several studies have been proposed in the literature which can be classified into three research directions: game theory, state estimation, and machine learning based solutions. Each category is described in the following paragraphs.

Game theory This direction intends to model honest and malicious customers as game players, and study their interaction with the utility company. In [15], a formal game theory model is developed. The study models the utility function for both honest and malicious customers and estimates the costs at the utility company under several statistical assumptions. A Non-cooperative Game (NCG) model is developed in [14], where the system encompasses two stages. First, a statistical model is used to obtain self-synchronization errors between profiled usage and real-time meter-reading usage. Then, malicious behaviours can be detected using a decision-making model based on a non-cooperative game. The main drawback of these studies is the necessity to assure strong assumptions regarding the malicious behaviours which restricts the feasibility of the studies.

State estimation This direction intends to estimate the state of the SG using available data such as customers' readings, demand rate, and load profiles which can help in estimating the errors or irregularities in the SG due to electricity theft. In [30], a central observer meter is used to estimate the difference between the charged and the actual consumption for a group of meters under investigation. False data injection has been investigated with state estimation in [31] where the attack model assumes a malicious customer can lower its consumption readings while increasing others' readings. In addition, a grid-placed sensors algorithm has been studied for better network observability. Recently, a spatiotemporal method that estimates the geographic locations of NTLs over time is proposed in [32]. One drawback of these techniques is that they estimate an aggregate loss for a group or subarea, while the origins of the loss cannot be determined.

Machine learning Machine learning studies are the most pertinent to this chapter. In [10], a comparison between the performance of two classification algorithms against SVM is proposed. Both algorithms are based on single feed-forward neural network, namely, extreme learning machine (ELM), and online sequential extreme learning machine (OSELM). The proposed approach is able to achieve 70% classification rate with ELM. Shallow SVM based detector has also been employed in [33] and achieved a detection accuracy up to 60% on a real dataset. The performance of this detector is further improved in [11] to reach an accuracy of 72% using a fuzzy inference system as a post-processing stage. A graph-based approach is proposed in [12] that uses optimum path forest for electricity theft detection and outperforms SVM with an accuracy of 89%.

Deep learning techniques for electricity theft detection are studied in [19, 28]. A comprehensive comparison study is presented in [19]. It compares between different deep learning architectures, such as convolutional neural networks (CNN), LSTM-RNNs, and stacked auto-encoders. Nevertheless, only two types of cyber-attacks are presented, and synthetic data is used to evaluate the performance of the proposed models, which cannot be compared reliably against models trained on real consumption data. Moreover, the effect of fine-tuning the models' hyper-parameters is not studied in [19]. In [28], a new CNN architecture is proposed. The architecture consists of two components: a deep component which encompasses multiple cascaded CNN layers, and a wide component which encompasses a fully connected flat layer. A real dataset from an electricity supply company in China is used for evaluation. However, no hyper-parameters tuning is performed.

To the best of our knowledge, the best-reported performance on a publicly available dataset is achieved by [3], which employs an SVM classifier, with an average detection rate of 94% and a false acceptance rate of 11%.

Other approaches Some other studies addressed the electricity theft problem with less prevalent methods such as random matrix [22], stochastic Petri net [21], dynamic programming [16], and partial observable Markov decision process [17]. However, the datasets used in many studies are synthetic and may not be reliable for practical applications. In addition, the experiments were conducted under different assumptions and conditions.

3 Dataset Used and Cyber Attacks Injection

To evaluate the performance of the proposed general electricity theft detectors, real smart meter data from the Irish Smart Energy Trials is used [34]. In this section, we describe the dataset used in our experiments and the method to inject cyber attacks into the dataset.

3.1 Energy Consumption Dataset

The dataset was published by the Electric Ireland and Sustainable Energy Authority of Ireland in January 2012. It contains the energy consumption readings for 4, 622 customers over 536 days between 2009–2010. The customers report their readings every 30 min, and hence, the total number of reports per customer is 25, 728. Table 1 gives a summary for the customers categories in the dataset. The dataset contains 3,639 residential customers, 427 small-to-medium enterprise (SME), and 556 of other types. In this work, we used residential customers to train and test our model, as this type is more vulnerable to electricity cyber-attacks.

3.2 Electricity Theft Cyber-Attacks

Since the dataset contains only honest customers and gathering a large set of malicious patterns is a cumbersome problem, we followed false data injection approach is presented in [3] to create a set of cyber-attack patterns that aim to reduce the total customer's energy consumption. The details for the cyber-attack generation process are explained in this Section.

Define a set of customers $\mathcal{C}$, a set of days $\mathcal{D}$=$\{1, \ldots, D\}$, and a set of periods $\mathcal{T} = \{1, \ldots, T\}$ of equal duration within each day. A smart meter is installed for each customer to report the energy consumption for load monitoring and billing regularly. For each customer c, the energy consumption value at a specific day d and time t is denoted as $E_c(d, t)$. Hence, each customer has an energy consumption record matrix E_c of actual consumptions, where the rows of E_c span the days in $\mathcal{D}$, while the columns span the periods in $\mathcal{T}$. The smart meter reports the consumption to the utility for customer c on day d and period t as $R_c(d, t)$. An honest customer reports the actual energy consumed at the end of the consumption period, and

Table 1 Customers categories in Irish smart energy trials dataset [34]

Users type	Count
Residential	3639
SME	427
Others	556

Table 2 Cyber attacks in [3]

Cyber attacks in [3]
$f_1(E_c(d,t)) = \alpha E_c(d,t)$
$f_2(E_c(d,t)) = \beta(d,t)E_c(d,t)$
$f_3(E_c(d,t)) = \begin{cases} 0 & \forall t \in [t_i(d), t_f(d)] \\ E_c(d,t) & \forall t \notin [t_i(d), t_f(d)] \end{cases}$
$f_4(E_c(d,t)) = \mathbb{E}[E_c(d)]$
$f_5(E_c(d,t)) = \beta(d,t)\mathbb{E}[E_c(d)]$
$f_6(E_c(d,t)) = E_c(d, T-t+1)$

hence, $R_c(d,t) = E_c(d,t)$. On contrary, a malicious customer aims to manipulate $E_c(d,t)$ to lower its electricity bill by modifying the smart meter readings. Hence, the cyber-attack is formally defined as $R_c(d,t) = f(E_c(d,t))$, where $f(\cdot)$ denotes a cyber-attack function that results in a reduced version for the malicious customer electricity bill. Although it is easy to collect honest samples for a customer c by monitoring its reports to the utility, malicious reports might not be easy to collect because the customer reports may all be honest or it has not been detected previously as a malicious customer. To tackle this problem, a set of cyber-attack functions were defined in [3] as indicated in Table 2. Each function $f(\cdot)$ creates a different attack scenario that aims to reduce the customers energy consumption $E_c(d,t)$. The first attack $f_1(\cdot)$ aims to reduce $E_c(d,t)$ by some fraction, where α denotes a flat reduction percentage. On the other hand, attack $f_2(\cdot)$ dynamically reduces $E_c(d,t)$ by a value controlled by the time function $\beta(d,t)$. The third attack $f_3(\cdot)$ represents a selective time filtering function, where the malicious customer reports zero reading during the interval $[t_i(d), t_f(d)]$, otherwise, the customer reports the actual consumption $E_c(d,t)$. Here, $t_i(d)$ and $t_f(d)$ denotes the initial and the final periods of the interval. The next two attacks $f_4(\cdot)$ and $f_5(\cdot)$ are based on the expected value of the energy consumption for the malicious customer for a given day denoted by $\mathbb{E}[E_c(d)]$. In $f_4(\cdot)$, the attacker reports a flat value during the day, while in $f_5(\cdot)$ the attacker reduces $\mathbb{E}[E_c(d)]$ dynamically from time to time using function $\beta(d,t)$. The last attack $f_6(\cdot)$ is a reverse function that reorders the energy consumption reports during the day so that higher consumption reports are assigned to low tariff periods.

Applying the aforementioned cyber-attacks on the energy consumption record matrix E_c results in six matrices $M_{c,i}$, where $i \in \{1, \ldots, 6\}$, for each attack $f_i(\cdot)$. The complete dataset for a customer is denoted as $\hat{X}_c$, which is a concatenation for E_c and all malicious matrices $M_{c,i}$. In addition, each row $\hat{X}_c(d,\cdot)$ represents either an honest day sample (i.e, E_c samples) labeled with 0 or a malicious sample (i.e., $M_{c,i}$ samples) labeled with 1. Since $\hat{X}_c$ is an imbalanced dataset where the ratio of the honest samples to the malicious samples is 1:6, the detector may be biased towards the dominant class category (i.e., malicious samples). To mitigate the bias effect, the adaptive synthetic sampling approach (ADASYN) [35] is used to oversample the minor class so that the ratio between the honest and the malicious samples is almost the same. Each dataset $\hat{X}_c$ is partitioned into two disjoints

datasets for model evaluation, specifically, a training set $\hat{X}_{c,\mathrm{TR}}$ and a testing set $\hat{X}_{c,\mathrm{TST}}$ with ratio 3:2. Three metrics are used to evaluate the performance of the detector, described as follows. The detection rate (DR) measures the percentage of the correctly detected malicious attacks (Eq. (1)). The false acceptance (FA) rate measures the percentage of the honest samples that are falsely identified as malicious (Eq. (2)). The highest difference (HD) measures the difference between DR and FA [3]. The detection latency is ignored as electricity theft attacks do not result in an immediate loss to the utility. Moreover, malicious customers can be fined later for any detected thefts.

$$Detection\ Rate\ (DR) = \frac{t_p}{t_p + f_p} \tag{1}$$

$$False\ Acceptance\ (FA) = \frac{f_p}{t_n + f_p} \tag{2}$$

Where, t_p corresponds to the number of true positives, f_p to false positives, and t_n to true negatives. These values can be obtained from the confusion matrix.

4 Proposed Deep Electricity Theft Detectors

In this section, we opt to address our first two questions: (1) customer-specific or generalized detector, and (2) feed-forward or RNNs. Therefore, we present the neural network architectures used in our experiments and the procedure used to train each architecture depending on the detector's type.

4.1 Deep Customer-Specific Electricity Theft Detector

To design a customer-specific detector (see Fig. 1b), two deep neural network architectures have been studied: a feed-forward neural network and a recurrent neural network. A model for each architecture is designed for each customer, and the optimal model parameters are learnt from training honest and malicious samples for that customer. Each architecture is designed to be trained on customer training set $\hat{X}_{c,TR}$ and tested on customer test set $\hat{X}_{c,TR}$ after pre-processing each set. Feature scaling is applied to customer training set $\hat{X}_{c,\mathrm{TR}}$ prior the training stage, since the range of energy consumption values for each customer varies widely across the day. Besides, a standard score is calculated by computing the mean and variance of $\hat{X}_{c,\mathrm{TR}}$, which is then used to scale the dataset so that it would have a zero mean and unit standard deviation over each dimension. The scaled training data is denoted as $X_{c,\mathrm{TR}}$. The same standard score is then applied to any test sample in $\hat{X}_{c,\mathrm{TST}}$ prior to its prediction stage to get $X_{c,\mathrm{TST}}$.

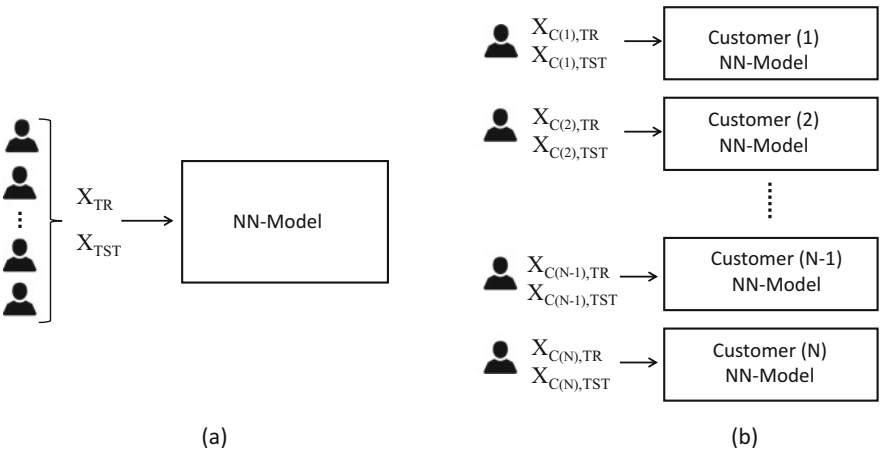


Fig. 1 Generalized versus customer-specific model. (**a**) Generalized NN-model. (**b**) Customer-Specific NN-model

4.2 Feedforward Detector

The neural network architecture of the feed-forward detector is shown in Fig. 2. The input to the neural network is given by the vector $x_c(d)$, which represents the energy consumption values for a customer $c \in \mathcal{C}$ at different time instants $t \in \mathcal{T}$ along a specific day d. The network architecture consists of L layers (including input and output layers) and N neurons within each hidden layer. The output layer has two neurons that define the malicious and the honest classes, where the true label can be represented by one-hot vector $y(x_c(d)) = (0\ 1)^T$ for honest customers, and $y(x_c(d)) = (1\ 0)^T$ for malicious customers. The output vector from each layer $l \in [1, \ldots, L]$ is denoted as o^l, where $o^L = x_c(d)$.

The weight of the connection from neuron n' in layer $l-1$ to neuron n in layer l is $w^l_{nn'}$ and the weight matrix for layer l is W^l. Let b^l_n represent the bias of neuron n in layer l and b^l is the bias vector for layer l. The activation of neuron n in layer l is denoted as a^l_n and the activation vector for layer l is a^l. The activation a^l_n is related to the activations in previous layer $l-1$ via an activation function $\sigma(z^l_n)$, where z^l_n represents a weighted sum of the inputs to neuron n, i.e., $z^l_n = \sum_{n'} w^l_{nn'} a^{l-1}_{n'} + b^l_n$. In a vector form, $z^l = W^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$. The detector design (training) stage involves defining the feed-forward network parameters W^l and b^l for all layers that yield the desired output labels $y_n(x(d))$ for any input $x(d)$. This can be achieved by minimizing the cross-entropy cost function C given by

$$C = \min_{\Theta} \frac{-1}{S} \sum_{X_{\mathrm{TR}}} \{y^{\mathrm{T}}(x_c(d)) \ln(o^L) + (1 - y^{\mathrm{T}}(x_c(d))) \ln(1 - o^L)\} \tag{3}$$

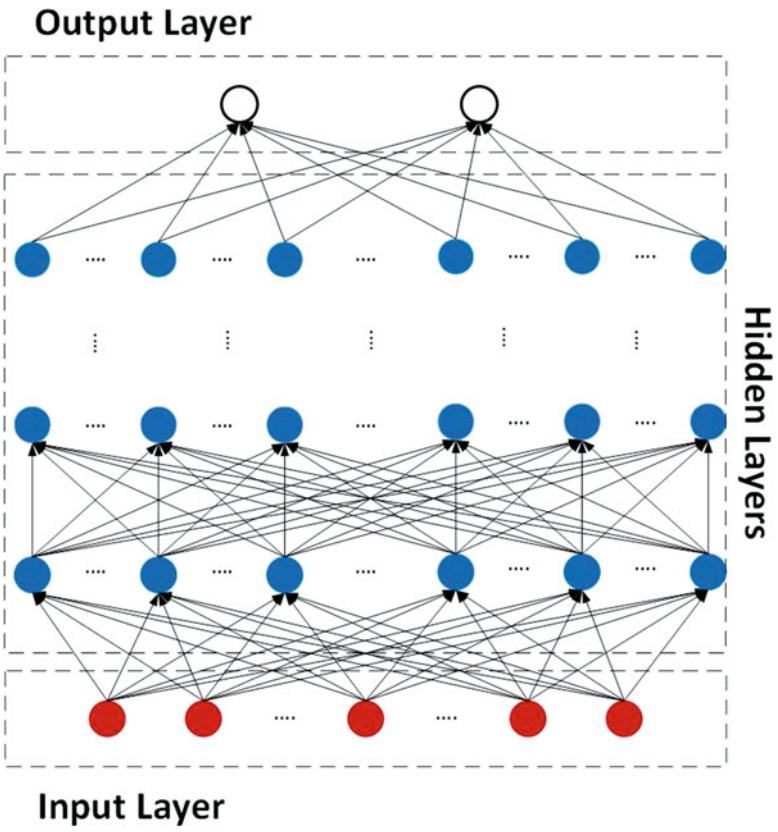


Fig. 2 Architecture of customer-specific feedforward electricity theft detector

where Θ denotes the network parameters W^l and b^l for all layers and S represents the number of training samples, given by the number of rows in X_{TR}. The first summation in Eq. (3) is over the output neurons and the second summation is over the training samples. The minimization in Eq. (3) is solved using an iterative gradient descent-based optimization algorithm.

The total number of iterations is denoted as I. The training samples in X_{TR} are divided into M mini-batches of equal size K. For each iteration i, the back-propagation algorithm calculates the gradient of the cost function in Eq. (3) as a function of an error term δ using the mini-batches [36]. Subsequently, the calculated gradient is used to update the weights and biases for each iteration. The complete training algorithm is given in Algorithm 1 using a stochastic gradient descent (SGD) optimization algorithm of learning rate η. In Algorithm 1, ∇_a denotes partial derivative with respect to a, $\odot$ represents the Hadamard product, $\sigma'(z^l(x))$ is the reciprocal of the partial derivative of z^l with respect to a^l, and T denotes the transpose operation.

Algorithm 1: Feed-forward electricity theft detector training

Data: $X_{c,\mathrm{TR}}$
Result: Optimal parameters W^l and b^l for all layers
Initialization: Weights w^l and biases b^l for all $l, i = 1$
while $i \neq I$ **do**
  Initialize: $m = 1$
  while $m \neq M$ **do**
    for *each training example* x *in mini-batch* m **do**
      Feed Forward: Compute: $z^l(x) = w^l a^{l-1}(x) + b^l$ and $a^l(x) = \sigma(z^l(x))$ for all layers $l = 2, \ldots, L$
      Back propagation: Compute $\delta^L(x) = \nabla_a C(x) \odot \sigma'(z^l(x))$ and for all layers $l = L-1, \ldots, 2$ compute $\delta^l(x) = ((w^{l+1})^{\mathrm{T}}\delta^{l+1}(x)) \odot \sigma'(z^l(x))$
    end
    Weight and bias update:
    $w^l = w^l - \frac{\eta}{K}\sum_x \delta^l(x)(a^{l-1}(x))^{\mathrm{T}}$
    $b^l = b^l - \frac{\eta}{K}\sum_x \delta^l(x)$
  end
end

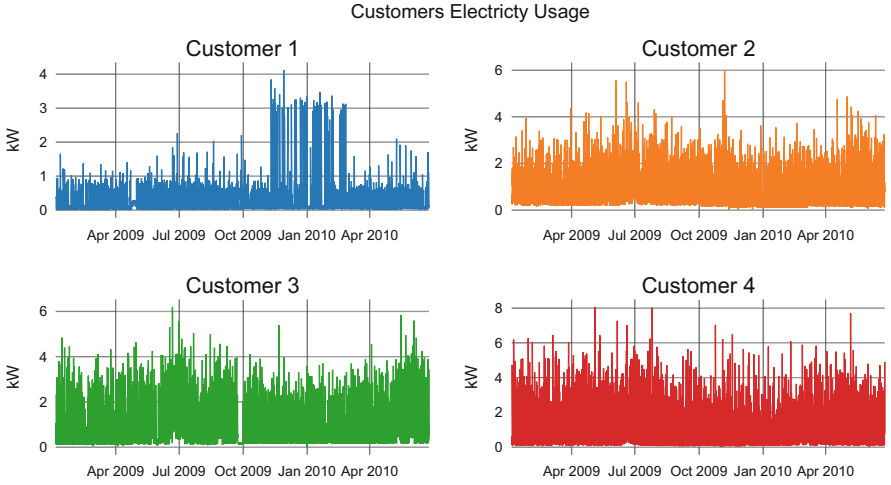


Fig. 3 Energy consumption of four customers from the dataset

4.3 Gated Recurrent Detector

Figure 3 gives the energy consumption of four random customers from the dataset. As shown in the figure, the consumption varies depending on the daily usage of each customer. It is also observed that it is hard to notice long-range correlation from this figure. On the other hand, Fig. 4 gives the energy consumption for customer one over four weeks period. It can be noticed from Fig. 4a that there is a weekly and daily correlation between customer consumption. Figure 4b gives the auto-correlation

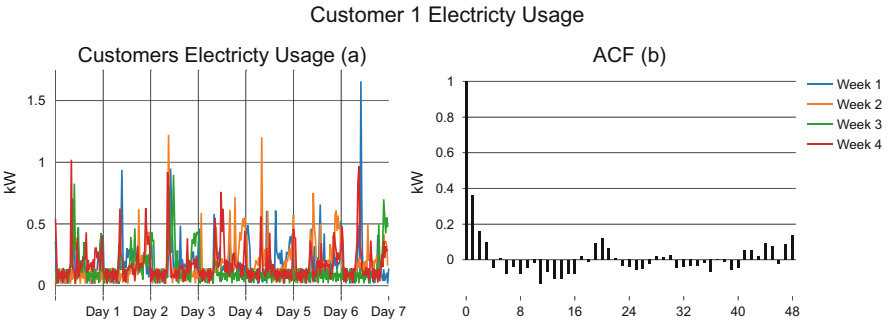


Fig. 4 Energy consumption of customer 1 over four weeks

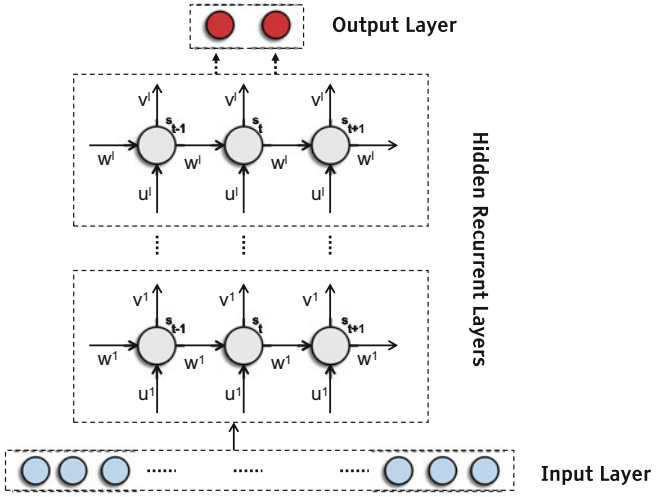


Fig. 5 Architecture of general RNN-based electricity theft detector

function for customer 1 with different time lags, where the positive correlation can be noticed at 48-time lag (i.e., one day). Based on this preliminary analysis we can conclude that *exploiting the temporal correlation within the time series energy consumption data of the customers can further enhance the theft detection performance.*

To make use of the temporal correlation within the energy consumption data of each customer, we designed a gated RNN-based detector. Gated recurrent unit (GRU) is considered as a variation of long-short-term-memory (LSTM) architecture which aims to solve the vanishing gradient problem with less complicated model (i.e., less training time) and reasonably comparable performance [37]. The architecture of the RNN-based general detector is shown in Fig. 5. The input layer is denoted by vector $x_c(d)$, which represents the consumption vector of customer c during day d at different reporting periods $t \in \mathcal{T}$. The RNN architecture consists of L hidden

GRU layers each with N neurons. Each GRU layer except the last one accepts a sequence vector as input and produces a sequence output vector. Unlike traditional feed-forward layers, recurrent layers are more efficient in exploiting patterns from sequential information. The output layer has two neurons that define the malicious and the honest classes, where the true label can be represented by one-hot vector $y(x_c(d)) = (0\ 1)^T$ for honest customers, and $y(x_c(d)) = (1\ 0)^T$ for malicious customers. The output vector from each layer $l \in \{1, \ldots, L\}$ is denoted as o^l, where $o^1 = x(d)$. Each hidden GRU layer $l \in \{2, \ldots, L-1\}$ has the following parameters:

- The input at time step t is o_t^{l-1}, which results from the previous layer $l-1$.
- The hidden state s_t^l at time step t that represents the memory computed based on the previous hidden state s_{t-1}^l of the same layer.
- The update gate $z_t^l = \sigma(o_t^{l-1} U_z^l + s_{t-1}^l W_z^l + b_z^l)$, which determines a combination between the new input o_t^{l-1} and the previous memory s_{t-1}^l. Here, U_z^l and W_z^l are learnable weights, $\sigma(.)$ is the activation function, and b_z^l is the bias vector.
- The reset gate $r_t^l = \sigma(o_t^{l-1} U_r^l + s_{t-1}^l W_r^l + b_r^l)$, which determines the amount of the previous memory s_{t-1}^l that will contribute to the next state using the equation $h_t^l = \tanh(o_t^{l-1} U_h^l + (s_{t-1}^l \odot r^l) W_h^l + b_h^l)$. Here, U_r^l, W_r^l, U_h^l, and W_h^l are learnable weight matrices, b_r^l and b_h^l are the bias vectors, and $\odot$ is the Hadamard product.
- The next state is simply $s_{t+1}^l = (1 - z^l) \odot h^l + z^l \odot s_t^l$, and the output at time $t+1$ is $o_{t+1}^l = \text{softmax}(V^l s_{t+1}^l + b_o^l)$, where V^l is a learnable weight matrix.

The detector training stage involves learning the parameters $U_{(.)}^l$,$W_{(.)}^l$,$V_{(.)}^l$, and $b_{(.)}^l$ for all layers, which results in the desired output vector $y(x_c(d))$ for any input $x_c(d)$. The optimization objective to find these weights can be achieved by minimizing the cross-entropy cost function C defined in (3), where Θ denotes the RNN parameters $U_{(.)}^l$, $W_{(.)}^l$, $V_{(.)}^l$, and $b_{(.)}^l$ for all layers and S represents the total number of the training samples for all customers, which equals to the number of rows in X_{TR}. Solving Eq. (3) for the optimal parameters Θ is done using an iterative gradient descent-based optimization algorithm. This is achieved by partitioning X_{TR} into M mini-batches of equal size and executing Algorithm 2 for I total iterations. Each iteration i has two main stages, namely, feed-forward and back-propagation. In the feed-forward stage, the training samples in the mini-batch are passed through all the network layers to calculate the predicted output vectors. On the other hand, the back propagation stage uses the mini-batches to calculate the gradient of the cost function in Eq. (3) with respect to the network weights [36]. Thereafter, the computed gradients are used to modify the weights and biases for each iteration. Note that, the back propagation within each layer is basically a back propagation through time (BPTT) as each layer is a GRU. In addition, ∇_a in Algorithm 2 denotes the partial derivative with respect to a. Algorithm 2 uses stochastic gradient descent (SGD) for optimization with learning rate η.

Algorithm 2: Recurrent electricity theft detector training

Data: X_{TR}
Result: Optimal parameters $U^l_{(.)}$, $W^l_{(.)}$, $V^l_{(.)}$, and $b^l_{(.)}$ $\forall l \in \{1, \ldots, L\}$
Initialization: Weights $U^l_{(.)}$, $W^l_{(.)}$, $V^l_{(.)}$, and $b^l_{(.)}$ $\forall l \in \{1, \ldots, L\}$, $i = 1$
while $i \neq I$ **do**
  Initialize: $m = 1$
  while $m \neq M$ **do**
    for *each training example x in mini-batch m* **do**
      Feed-Forward:
      for *each recurrent layer l* **do**
        for *each time step t* **do**
          $z^l_t = \sigma(o^{l-1}_t U^l_z + s^l_{t-1} W^l_z + b^l_z)$
          $r^l_t = \sigma(o^{l-1}_t U^l_r + s^l_{t-1} W^l_r + b^l_r)$
          $h^l_t = \tanh(o^{l-1}_t U^l_h + (s^l_{t-1} \odot r^l) W^l_h + b^l_h)$
          $s^l_t = (1 - z^l) \odot h^l + z^l \odot s^l_{t-1}$
          $o^l_t = \mathrm{softmax}(V^l s^l_t + b^l_o)$
        end
      end
      Back propagation: Compute $\nabla_{U^l_{(.)}} C(x)$, $\nabla_{V^l_{(.)}} C(x)$, $\nabla_{W^l_{(.)}} C(x)$, and $\nabla_{b^l_{(.)}} C(x)$
    end
    Weight and bias update:
    $U^l_{(.)} = U^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{U^l_{(.)}} C(x)$
    $V^l_{(.)} = V^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{V^l_{(.)}} C(x)$
    $W^l_{(.)} = W^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{W^l_{(.)}} C(x)$
    $b^l_{(.)} = b^l_{(.)} - \frac{\eta}{K} \sum_x \nabla_{b^l_{(.)}} C(x)$
  end
  m=m+1
end

4.4 Deep Generalized Electricity Theft Detector

Customer-specific detector suffers from contamination attacks, where the data used to train each customer model might be malicious and falsely labelled as honest. Moreover, newly joined customers need to submit historical data for their consumption to train their detector. Besides, there is an unlimited number of choices for the hyper-parameters of each architecture. To solve these issues, we opt to design a generalized detector that is trained and optimized on the data that comes from different customers (see Fig. 1b). The network architectures for the generalized detector is the same as the customer-specific. However, the training sets of all or a subset number of customers in $\mathcal{C}$ are merged to form $\hat{X}_{\mathrm{TR}}$. Similarly, the test sets of all or a subset of customers are merged to form $\hat{X}_{\mathrm{TST}}$. As different customers have different ranges of energy consumption values during different report periods $\mathcal{T}$, different consumption periods have different opportunities to

influence the optimization algorithm for the final model. Hence, in order to enforce equal opportunities for different consumption periods, feature scaling is applied to $\hat{X}_{\mathrm{TR}}$. The mean and the variance of $\hat{X}_{\mathrm{TR}}$ are calculated and used to scale $\hat{X}_{\mathrm{TR}}$ so that each consumption period would have zero mean and unit standard deviation. The output of the feature scaling stage is denoted as X_{TR}. The same scaling scores are further applied to the test set to get X_{TST}.

5 Hyper-Parameters Tunning

In this section, we discuss three hyper-parameters tuning algorithms: sequential, random, and genetic-based optimization each of them can be used with the neural architectures presented in Sect. 4. These algorithms opt to address our third question of choosing the tuning algorithm that makes a trade-off between the optimality and the complexity. Unlike sequential and random optimization methods, genetic-based optimization employs a smart and adaptive searching heuristic; however, it is more complicated than former methods.

5.1 Sequential Grid Search Tunning for Hyper-Parameters

The optimal choice of the detector's hyper-parameters results in improved detection performance. Therefore, the goal is to optimize following hyper-parameters for our detectors: the number of hidden layers (L), the number of neurons within each hidden layer (N), the type of gradient descent-based optimization algorithm used in finding the optimal feed-forward parameters (weights and biases), the initialization method for the parameters, the dropout rate, the weight constraint, and the type of activation functions used at the feed-forward layers [36]. Let $\mathcal{L}$ and $\mathcal{N}$ represent finite sets of the number of hidden layers and the number of neurons that can be adopted for the detector architecture. Denote the set of optimization algorithms that can be used to find the optimal detector parameters by $\mathcal{O}$, which can include SGD, RMSProp, Adagrad, etc. [36]. The set of initializers that can be used for the feed-forward network parameters is given by $\mathcal{H}$, which can include Normal, Uniform, Glorot Normal, etc. [36]. The set of dropout rates and weight constraint values are given by $\mathcal{B}$ and $\mathcal{G}$, respectively. Let $\mathcal{A}_{\mathrm{H}}$ and $\mathcal{A}_{\mathrm{O}}$ represent sets of activation functions for the hidden layers and output layer, respectively, which can include Relu, Sigmoid, etc. [36]. Due to the large number of hyper-parameters that can be optimized, an exhaustive grid search over all these hyper-parameters incurs high computational complexity. Hence, in this section, we adopt a sequential grid search analysis, which presents a sub-optimal mechanism. As shown in Algorithm 3, we optimize each hyper-parameter separately in sequential stages by looking for the best one or two hyper-parameters at each stage. Since this approach is greedy, a sub-optimal choice for the hyper-parameters could be achieved. Cross-validation

Algorithm 3: Sequential grid search

Result: Optimal hyper-parameters $P^* = \{L^*, N^*, O^*, H^*, B^*, G^*, A_{\mathrm{H}}^*, A_{\mathrm{O}}^*\}$

Initialization: Optimizer = SGD, Initializer = Glorot Uniform, dropout rate = 0, weight constraint = none, hidden and output layers activation functions = Relu and Sigmoid
for $\mathcal{L}$ **do**
  for $\mathcal{N}$ **do**
    Apply training algorithm and record the detection accuracy.
  end
end
Define model MD1 with hyper-parameters: L^*, N^*, and initial setting for the rest of hyper-parameters.
for $\mathcal{O}$ **do**
  Apply training algorithm using MD1 and record the detection accuracy.
end
Define model MD2 with hyper-parameters: L^*, N^*, O^*, and initial setting for the rest of hyper-parameters.
for $\mathcal{H}$ **do**
  Apply training algorithm using MD2 and record the detection accuracy.
end
Define model MD3 with hyper-parameters: L^*, N^*, O^*, H^*, and initial setting for the rest of hyper-parameters.
for $\mathcal{B}$ **do**
  for $\mathcal{G}$ **do**
    Apply training algorithm using MD3 and record the detection accuracy.
  end
end
Define model MD4 with hyper-parameters: L^*, N^*, O^*, H^*, B^*, G^*, and initial setting for the rest of hyper-parameters.
for $\mathcal{A}_{\mathrm{H}}$ **do**
  for $\mathcal{A}_{\mathrm{O}}$ **do**
    Apply training algorithm using MD4 and record the detection accuracy.
  end
end
Define model MD5 with hyper-parameters: L^*, N^*, O^*, H^*, B^*, G^*, A_{H}^*, A_{O}^*.

is implemented over X_{TR} to decrease the sub-optimality chance. Let P^* denote the hyper-parameter setting that results in improved detection accuracy against the validation set. A given combination of hyper-parameters results in a specific feed-forward model (MD), as shown in Algorithm 3.

5.2 Random Grid Search Tunning for Hyper-Parameters

Despite the efficiency of sequential search for hyper-parameters tuning over the extensive search, the searching process is greedy and may result in a sub-optimal and non-optimized solution for the hyper-parameters. In this sub-section, we employ a random search for hyper-parameters optimization. Random search is used to

find a sub-optimal but an efficient solution in a reduced time compared with the exhaustive grid search [38]. The following hyper-parameters are optimized using random search: the network architecture in terms of the number of hidden layers and the number of neurons within each hidden layer, the type of activation functions used at the hidden layers, the type of activation functions used at the output layer, and the type of gradient descent-based optimization algorithm used in finding the optimal RNN parameters (weights and biases) [36]. Denote $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{N}}$ as two uniform distributions that represent the number of hidden layers and neurons to be sampled by the random search algorithm. In addition, Let $\mathcal{O}$ represent a uniform distribution of optimization algorithms that can include Adam, SGD, Adamax, etc. [36]. Moreover, uniform distributions for the set of initializers that can be used for the feedforward network parameters as $\tilde{\mathcal{H}}$, which can include Normal, Uniform, Glorot Normal, etc. [36]. Also, two uniform distributions for the set of dropout rates and weight constraint values as $\tilde{\mathcal{B}}$ and $\tilde{\mathcal{G}}$, respectively. Finally, two uniform distributions are defined to represent the set of activation functions used by the hidden and output layers, namely, $\tilde{\mathcal{A}}_H$ and $\tilde{\mathcal{A}}_O$, respectively. Algorithm 4 shows the random search over X_{TR} for the selection of the optimal hyper-parameters defined as $\tilde{P}^*$. The maximum number of search iterations is denoted as I, where at each iteration all the distributions are sampled uniformly to create random vectors which correspond to different architectures, the sampled architecture is evaluated using K-fold cross-validation, and the best three models are selected. It should be noted that unlike the sequential search where no interaction among hyper-parameters is considered during each selection stage (greediness), in random search interaction is accounted but only a random subset of hyper-parameter options are examined.

Algorithm 4: Random search hyper parameters optimization

Data: X_{TR}

Result: Optimal hyper-parameters $\tilde{P}^* = L^*, N^*, O^*, H^*, B^*, G^*, A_H^*$, and A_O^*

Initialization: Weights $U_{(.)}^l$, $W_{(.)}^l$, $V_{(.)}^l$, and $b_{(.)}^l$ $\forall l \in \{1, \ldots, L\}$, $i = 1$

while $i \neq I$ **do**

  $L[i] \leftarrow \tilde{\mathcal{L}}$, $N[i] \leftarrow \tilde{\mathcal{N}}$, $O[i] \leftarrow \tilde{\mathcal{O}}$, $H[i] \leftarrow \tilde{\mathcal{H}}$, $B[i] \leftarrow \tilde{\mathcal{B}}$, $G[i] \leftarrow \tilde{\mathcal{G}}$, $A_H[i] \leftarrow \tilde{\mathcal{A}}_H$, $A_O[i] \leftarrow \tilde{\mathcal{A}}_O$

  for *each* $\bar{X}_{\text{TR}}$, $\bar{X}_{\text{TST}}$ *in k-folds*(X_{TR}) **do**

    Apply training algorithm using sampled hyper-parameters and record DR, FA and accuracy.

  end

  Record average values across folds DR$[i]$, FA$[i]$ and accuracy$[i]$.

  $i = i + 1$

end

Report the hyper-parameters of the top three models.

5.3 Genetic Algorithm-Based Tunning for Hyper-Parameters

Previously discussed hyper-parameters tuning solutions are sub-optimal but yet efficient in finding a fast solution. Unlike random optimization where a group of random models are examined, and the best is selected, genetic-based optimization applies a smart searching heuristic that considers the detector performance while exploring the hyper-parameters search space. As discussed in Sect. 3.2 the performance evaluation of our detectors is done using three metrics: DR, FA, and their difference HD. The former two metrics are conflicting objectives, where the optimal hyper-parameters should maximize the DR and minimize the FA. Therefore, a two objectives optimization problem can be formulated for our hyper-parameters search heuristic, where the overall objective is denoted as $maximize(DR, -FA)$. Conversely, maximizing only the HD could also define our search heuristic objective. However, same HDs points could be evaluated to different DRs and FAs, which cannot be compared reasonably with each other. In this section, a multi-objective genetic algorithm is used to tune the hyper-parameters for both the feed-forward and the gated recurrent model. In particular, Nondominated Sorting Genetic Algorithm II (NSGA-II) [39] is used to find a set of optimal (DRs and FAs) solutions known as Pareto frontier, after that, a compromise solution can be calculated. Therefore, the performance of our neural network detector is the fitness function for the NSGA-II algorithm. In addition, the chromosome for the NSGA-II algorithm comprises the set of hyper-parameters to be optimized (As shown in Fig. 6). More specifically, the chromosome includes the following: the number of hidden layers (L), the number of neurons within each hidden layer (N), the type of gradient descent-based optimization algorithm (O), the initialization method for the parameters (H), the dropout rate (B), the weight constraint (G), and the type of activation functions (A_H and A_O).

As shown in Algorithm 5, the maximum number of genetic iterations is denoted as M, where at each iteration the main steps of the genetic algorithm are executed. Firstly, the fitness functions, namely DR and FA, are computed for each individual in the population $G_1[i]$ by training each neural network population on a random subset of combined customers dataset. Note that, in genetic-based hyper-parameters optimization, only a subset of customers is used for each neural network model since the number of networks to be trained and tested is large. Hence, all customers have the same opportunity in training and evaluating the general model. Then, an off-spring population $G_2[i]$ is created from $G_1[i]$ by selection, crossover,

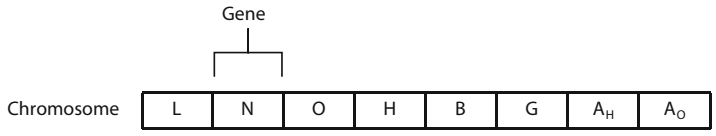


Fig. 6 Hyper-parameter chromosome

Algorithm 5: Genetic algorithm-based tunning for hyper-parameters

Result: Optimal hyper parameters L^*,N^*,O^*, H^*, B^*, G^*, A_H^*, and A_O^*

Initialization: Weights

$G[0]$ = Create generation of size n randomly-generated individuals;

$X_{TR}[0]$ = Create a population of n training sets ; `/* Each training set has m customers randomly sampled with replacement from all the training customers */`

$i = 0$

while $i \neq M$ **do**

`// Compute fitness for each individual x ∈ G1[i]`

For each individual $x \in G_1[i]$ Compute fitness on $X_{TR}[i]$.

`// Off-spring generation from G1[i]`

Generate an off-spring population $G_2[i]$ from $G_1[i]$, via selection, crossover, and mutation.

`// Compute fitness for each individual x ∈ G2[i]`

For each individuals $x \in G_2[i]$ Compute fitness on $X_{TR}[i]$.

`// Combine G1[i] and G2[i]`

Assign $G[i] = G_1[i] \cup G_2[i]$

`/* Sort generation individuals and partition them to sorted subsets of non dominated fronts */`

$\{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_l\}$ = Non-dominated-sort($G[i]$)

`// Create generation i + 1`

$G_1[i+1] = [\,]$

$j = 1$

`// Fill G1[i+1] from the nondominated fronts`

while $|G_1[i+1]| + |\mathcal{F}_j| < n$ **do**

Add $\mathcal{F}_j$ to $G_1[i+1]$

$j = j + 1$

end

`/* In case of the size of G1[i+1] is less than n, the individuals within the last Pareto frontier are crowd sorted and top k are selected */`

if $|G_1[i+1]| < n$ **then**

Crowd-distance-sorting($\mathcal{F}_j$)

Add top k individuals from $\mathcal{F}_j$ to $G_1[i+1]$, such that $|G_1[i+1]| = n$

end

$X_{TR}[i]$ = Create a population of n training sets

$i = i + 1$

end

Calculate a compromise solution for the hyper-parameters from the Pareto frontier $\mathcal{F}_1$ of $G[M-1]$.

and mutation and combined with $G_1[i]$ to form $G[i]$. The nondominated sort arranges and splits the trained individuals according to the optimization objective $maximize(DR, -FA)$ into a set of Pareto fronts $\{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_l\}$ ranked by their dominance. Specifically, for any individual $\mathbf{a} \in \mathcal{F}_2$ there exist an individual $\mathbf{b} \in \mathcal{F}_1$ dominates $\mathbf{a}$ in terms of the two fitness values DR and FA, i.e., the following two inequalities are satisfied $DR(\mathbf{a}) < DR(\mathbf{b})$ and $FA(\mathbf{b}) > FA(\mathbf{a})$. If a Pareto frontier $\mathcal{F}_j$ has to be split to complete the correct size of generation $G[i+1]$, then

a crowd distance sorting is done inside this frontier, and the topmost individuals that complete the correct size of generation are selected. Crowd distance sorting results in a more diverse solution in the Pareto frontier by selecting individuals with less density. The crowd distance is calculated for each individual by computing the difference between its two fitness values and the two fitness values of its right and left neighbours. Then, the overall crowd distance of that individual is calculated by summing the crowd distance calculated for the two fitness values. The individuals within the frontier $\mathcal{F}_j$ are then sorted in descending order of their crowd distance, and the topmost individuals are selected to complete the new population $G[i+1]$. Finally, using the Pareto frontier of the last generation, a compromise solution of the hyper-parameters can be calculated.

Theoretically, any point on the Pareto frontier can be considered as an optimal hyper-parameters operating point. However, extreme points (DR = 1 AND FA = 1 OR DR = 0 AND FA = 0) does not offer a right balance between our two objectives DR and FA. In addition, a theoretical ideal solution where the DR is maximized and the FA is minimized (i.e., DR = 1 AND FA = 0), also know as utopia point [40], is impractical due to the requirement to optimize two conflicting objectives simultaneously, which is not feasible. Hence, Eq. (4) gives a compromise but practical solution S, which is defined as the point that minimizes the distance between the Pareto frontier and the utopia point.

$$S = \min_{1 \le i \le K} \sqrt{(DR_i - DR^*)^2 + (FA_i - FA^*)^2} \tag{4}$$

where K denotes the number of points on the Pareto frontier, DR^* denotes the ideal detection rate (i.e., DR = 1), FA^* denotes the ideal false alarm (i.e., FA = 0).

6 Numerical Results and Discussion

Real smart meter data from the Irish Smart Energy Trials is used to evaluate the performance of the proposed RNN-based general electricity theft detector, [34]. The dataset was published by the Electric Ireland and Sustainable Energy Authority of Ireland in January 2012. It contains the energy consumption readings for 3,639 residential customers over 536 days between 2009–2010. The customers report their readings every 30 minutes (i.e., the neural network input size is 48), and hence, the total number of reports per customer is 25,728. Each customer dataset $\hat{X}_c$ is used to for cyber-attacks injection given in Table 2. For cyber-attacks $f_1(\cdot)$, $f_2(\cdot)$, and $f_5(\cdot)$, α and $\beta(d,t)$ are random variables that are uniformly distributed over the interval $[0.1, 0.8]$ [3]. For attack $f_3(\cdot)$, $t_{\mathrm{i}}(d)$ is a uniform random variable in $[0, 42]$, and the duration of the attack, i.e., $t_{\mathrm{f}}(d) - t_{\mathrm{i}}(d)$, is a uniform random variable in $[8, 48]$, and hence, the maximum value of $t_f(d) = 48$. Applying all cyber-attacks for

each customer results in $\hat{X}_c$ dataset which contains 536 honest samples (days) and 3, 216 malicious samples. Each sample in $\hat{X}_c$ has 48 energy consumption values. For each $\hat{X}_c$ dataset, over-sampling is performed to balance the size of honest and malicious classes. Consequently, the total size of each $\hat{X}_c$ dataset contains a total of 6,432 honest and malicious samples. Each $\hat{X}_c$ dataset is partitioned into training set $\hat{X}_{c,\mathrm{TR}}$ and testing set $\hat{X}_{c,\mathrm{TST}}$ with ratio 3:2. All the experiments were performed on the high-performance cluster (HPC) of the Tennessee Tech University using two NVIDIA Tesla K80 GPUs. Tensorflow and Python Keras Library are used for the implementation.

6.1 Customer-Specific Detector Evaluation with Sequential Grid Search for Hyper-Parameters Tuning

In Algorithms 1 and 2 , the number of epochs $I = 50$ and batch sizes $K = 100$. For the neural architectures, the following initial set up is used: parameter initialization = Glorot Uniform, optimizer = SGD, hidden layer activation = Relu, output layer activation = Sigmoid. Algorithm 3 is used for grid search with the following sets $\mathcal{L} = \{4, 6, 8\}$ for the feed-forward and $\mathcal{L} = \{2, 3, 4\}$ for the gated-recurrent, $\mathcal{N} = \{100, 500, 1000\}$, for the feed-forward,and $\mathcal{N} = \{300, 400, 600\}$ for the gated-recurrent, $\mathcal{O} = \{$SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam$\}$ [36], $\mathcal{H} = \{$Glorot Uniform, Normal, Lecun Uniform, Glorot Normal, He Normal, He Uniform, Uniform $\}$ [36], $\mathcal{B} = \{0, 0.2, 0.5\}$, $\mathcal{G} = \{1, 3, 5\}$, $\mathcal{A}_{\mathrm{H}} = \{$Relu, Softplus, Linear, Sigmoid, Tanh$\}$, and $\mathcal{A}_{\mathrm{O}} = \{$Softmax, Sigmoid$\}$.

Figure 7 shows the grid search results for the hyper-parameters of the customer-specific electricity theft detector. As shown in the figure, the feed-forward detector benefits from the deep architecture as 60% of the customers' implement a feed-forward of 8 hidden layers. Roughly, an equal number of customers employs 100, 500, and 1000 neurons in their hidden layers. Similarly, for the optimizers, almost an equal number of customers employs SGD, RMSProp, Adadelta, Adam, Adamax, and Nadam as optimizers while almost 4% of customers employs Adagrad as their optimizer. Furthermore, the majority of customers (42%) employs Golort Uniform for parameter initialization. Besides, the majority of the customers do not use dropout (62%) or weight constraints (40%). For the activation functions, the majority of customers employ Relu for hidden layers (almost 90%) and Sigmoid for the output layer (almost 75%).

Table 3 presents the average detection performance over the customer's test data in terms of DR, FA, and HD. Each column presents the average performance over all customers using the hyper-parameters determined by the feed-forward model specified in the first row. Hence, the first column presents performance results when using the hyper-parameters defined by model MD1 in Algorithm 1 (i.e., number of layers and neurons following the distribution is shown in Fig. 7a, b respectively, while using initial set up for other hyper-parameters). The next column

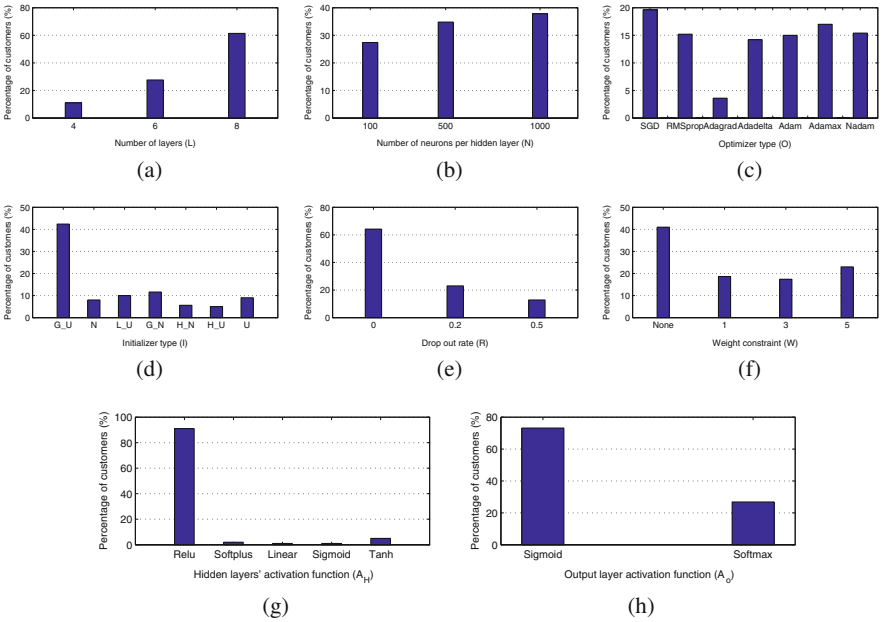


Fig. 7 Feed-forward grid search analysis results for all customers. (**a**) Optimal layers. (**b**) Optimal neurons. (**c**) Best optimizers. (**d**) Optimal initializer. (**e**) Optimal dropout rate. (**f**) Optimal weight constraint. (**g**) Optimal hidden layer activation. (**h**) Optimal output layer activation

Table 3 Customer-specific feed-forward average detection performance using sequential search for hyper-parameters tuning

		MD1	MD2	MD3	MD4	MD5	[3]
Feed-forward	DR (%)	91.3	92.3	92.6	92.7	93	**94**
	FA (%)	5.8	2.8	**2.3**	2.9	3	11
	HD (%)	85	89.5	**90.3**	89.8	90	83

presents performance results when using the hyper-parameters defined by model MD2 in Algorithm 1 (i.e., optimizer types following the distribution in Fig. 7c while using layer and neuron distributions according to Fig. 7a, b and initial set up for the remaining hyper-parameters), etc. The last column presents results from [3]. The work in [3] uses the same data we are using in our study; however, the detector in [3] is based on a shallow machine learning architecture (SVM). Since the grid search analysis is carried out over an accuracy metric, we can see that the DR keeps improving when grid search is applied over different hyper-parameters. As shown in Table 3, the best HD (and the lowest FA rate) is achieved when MD3 hyper-parameters are applied (i.e., layers, neurons, optimizers, and initializers follow the distributions shown in Fig. 7a–d, respectively, while no dropout or weight constraints are applied, and Relu and Sigmoid activation functions are used for hidden and output layers, respectively). The optimal feed-forward model (MD3)

presents 80% reduction in FA rate compared with the shallow architecture in [3], as the FA rate reduced from 11% in [3] to 2.3% in the feed-forward based detector. The best-reported HD by the feed-forward based detector is improved from 83% in [3] to 90.3%. The results suggest that improved electricity theft detection performance is achieved when deep feed-forward architectures are implemented with optimized hyper-parameters compared with other shallow machine learning architectures.

6.2 Generalized Detector Evaluation with Random Grid Search for Hyper-Parameters Tuning

In this experiment, the training sets for a set of 200 customers are merged together to form $\hat{X}_{\mathrm{TR}}$. Similarly, the test sets for all customers are merged together to form $\hat{X}_{\mathrm{TST}}$. Feature scaling on training data produces X_{TR} and applying the same scores on the test data produces X_{TST}. In Algorithms 1 and 2, the total number of epochs is $I = 10$ and the batch size is 350. For Algorithm 4, the options used for the random search are the following: $\mathcal{L} = \{2, 3, 4\}$, $\mathcal{N} = \{100, 500\}$, $\mathcal{O}$ ={SGD,Adadelta, Adagrad, Adam, Adamax, Nadam, RMSprop}, $\mathcal{A}_H$ ={Sigmoid, Relu, Hard Sigmoid, Tanh}, and $\mathcal{A}_O$ ={Sigmoid, Softmax}. In addition, the total number of iterations is $I = 30$, the drop out rate is set to 0.2 and the initialization of the network neurons is set to Glorot Uniform [36].

Table 4 gives the random search evaluation results obtained by applying Algorithm 4 on the sampled hyper-parameters using GRU model. The results give the average detection performance over K-fold cross-validation (with $K = 3$) on X_{TR} in terms of DR, FA, and HD. As indicated in the table, the general electricity theft detector benefits from the recurrent deep architecture as the HD can achieve 91.4%. It can also be noticed that all the networks that have Relu samples failed to optimize properly, which is attributed to the finding discussed in [41] in which deep sigmoidal networks can outperform deep Relu networks if initialized properly. Moreover, all the networks with SGD optimizer struggle to achieve good results, which can be attributed to the small number of epochs used for the training. Finally, all the promising models have Softmax activation for the output layer. The top three models with respect to HD for each number of layers $l \in \{2, 3, 4\}$ are highlighted in Table 4 and further used for performance evaluation using the test set X_{TST}.

Table 5 gives a comparison between the top three models highlighted in Table 4 and the model proposed in [3], which was tested by the same data used in this chapter. The first three columns in Table 5 give the average performance results when using the best hyper-parameters highlighted in Table 4, such that the models with $l = 2$, $l = 3$, and $l = 4$ are represented by MD1, MD2, and MD3, respectively. The last column gives the results from [3]. While the detector in [3] is trained on the same dataset used in this chapter, it is a customer-specific electricity theft detector based on a shallow machine learning architecture (SVM). As indicated in Table 4, the best HD is achieved when MD2 or MD3 hyper-parameters are applied, which outperforms the proposed detector in [3] by a 4% increase. Moreover, the FA is

Table 4 Sampled hyper-parameters results of Algorithm 4 using k-fold cross validation.

L	N	A_H	A_O	O	DR	FA	HD	Accuracy
2	201	Hard Sigmoid	Sigmoid	Adam	0.930	0.029	0.900	0.950
	384	Sigmoid	Sigmoid	SGD	0.707	0.313	0.394	0.697
	428	**Sigmoid**	**Softmax**	**Adamax**	**0.946**	**0.036**	**0.910**	**0.955**
	270	Relu	Sigmoid	Nadam	0	0	0	0.499
	306	Hard Sigmoid	Softmax	SGD	0.727	0.324	0.403	0.701
	258	Hard Sigmoid	Sigmoid	SGD	0.713	0.311	0.402	0.701
	154	Hard Sigmoid	Softmax	Adagrad	0.880	0.067	0.812	0.906
	250	Tanh	Sigmoid	Adadelta	0.916	0.051	0.865	0.932
	351	Sigmoid	Softmax	Adamax	0.941	0.035	0.905	0.952
	284	Hard Sigmoid	Softmax	SGD	0.739	0.340	0.398	0.699
	105	Tanh	Sigmoid	Adagrad	0.877	0.083	0.793	0.896
3	385	Tanh	Sigmoid	Adadelta	0.928	0.037	0.890	0.945
	307	Sigmoid	Sigmoid	RMSprop	0.923	0.031	0.891	0.945
	239	Tanh	Sigmoid	Adagrad	0.609	0.328	0.281	0.640
	272	Hard Sigmoid	Sigmoid	Adadelta	0.885	0.071	0.813	0.906
	110	Sigmoid	Softmax	Nadam	0.928	0.036	0.891	0.945
	251	Tanh	Sigmoid	Nadam	0	0	0	0.499
	310	**Hard Sigmoid**	**Softmax**	**Adam**	**0.939**	**0.033**	**0.905**	**0.952**
	407	Hard Sigmoid	Sigmoid	Adagrad	0.867	0.075	0.791	0.895
	193	Sigmoid	Sigmoid	Adadelta	0.875	0.068	0.807	0.903
4	492	Relu	Sigmoid	SGD	0	0	0	0.499
	198	Tanh	Softmax	RMSprop	0.419	0.186	0.232	0.615
	352	Sigmoid	Softmax	Adamax	0.931	0.026	0.905	0.952
	215	**Tanh**	**Softmax**	**Adamax**	**0.942**	**0.028**	**0.914**	**0.957**
	275	Tanh	Softmax	Adam	0	0	0	0.499
	202	Relu	Sigmoid	Adagrad	0	0	0	0.499
	393	Sigmoid	Sigmoid	RMSprop	0.939	0.044	0.894	0.947
	165	Sigmoid	Softmax	Adagrad	0.898	0.073	0.825	0.912
	390	Hard Sigmoid	Sigmoid	Adadelta	0.608	0.359	0.249	0.624
	310	Tanh	Sigmoid	RMSprop	0.187	0.126	0.061	0.530

Table 5 Average Performance of the generalized theft detector using a random search for hyper-parameters tuning

		MD1	MD2	MD3	[3]
GRU	DR (%)	**93.4**	92.5	92.7	94
	FA (%)	6.9	**5.0**	5.3	11
	HD (%)	86.4	**87.4**	87.4	83

improved by almost 55%. The results indicate that an improvement in electricity theft detection performance can be achieved using RNN architecture. In addition, the generalized detector is a general model that does not rely on specific customer data. Hence, it is more robust against contamination attacks compared with the proposed detector in [3].

6.3 *Generalized Detector Evaluation with Genetic-Based Search for Hyper-Parameters Tuning*

In this experiment, each genetic individual is a neural network which is either feed-forward or GRU architecture. In Algorithm 5, the number of individuals inside each population n is 20, and the total number of generations $M = 15$. During each generation, each neural individual is trained on the training sets of 30 sampled customers after merging them together to form $\hat{X}_{\mathrm{TR}}$. Similarly, the test sets the 30 customers are merged together to form $\hat{X}_{\mathrm{TST}}$. Feature scaling on training data produces X_{TR} and applying the same scores on the test data produces X_{TST}. In Algorithms 1 and 2, the total number of epochs is $I = 20$ and the batch size is 350. For Algorithm 4, the options used for the random search are the following: for GRU $\mathcal{L}$ ={1, 2, 3, 4}, for feed-forward $\mathcal{L}$ ={2, 4, 6, 7}, $\mathcal{N}$ ={64, 128, 256, 512, 768, 1024}, $\mathcal{O}$ ={SGD, Adadelta, Adagrad, Adam, Adamax, Nadam, RMSprop}, $\mathcal{A}_H$ ={Sigmoid, Relu, Hard Sigmoid, Tanh}, $\mathcal{A}_O$ ={Sigmoid, Softmax}, $\mathcal{B}$ = {0.2, 0.3, 0.4, 0.5}, and $\mathcal{G} = \{1, 3, 5\}$.

Figure 8 gives the Pareto frontier result of the last generation for both the feed-forward and the GRU model. From the figure, we notice that GRU has a sharper elbow than feed-forward, which can be attributed to the benefits that the GRU acquired from the time-series correlation features. Solving both frontiers for the compromise solution of gives **FeedForward = { L: 8, N: 1024, H: Lecun Uniform, O: Nadam, $\mathbf{A_O}$: Softmax, $\mathbf{A_H}$: Relu, B: 0.2, G:1 }** as a solution for the feed-forward architecture, and **GRU= {L: 2, N: 512, H: Glorot Normal, O: Adam, $\mathbf{A_O}$: Sigmoid, $\mathbf{A_H}$: Linear, B: 0.0, G: 3}** as a solution for the GRU architecture. Table 6 gives a comparison between the two Pareto models **FeedForward** and **GRU** and the model proposed in [3] by training and testing the final model on the merged data of 200 customers. As indicated in Table 6, the best HD is achieved the **GRU**

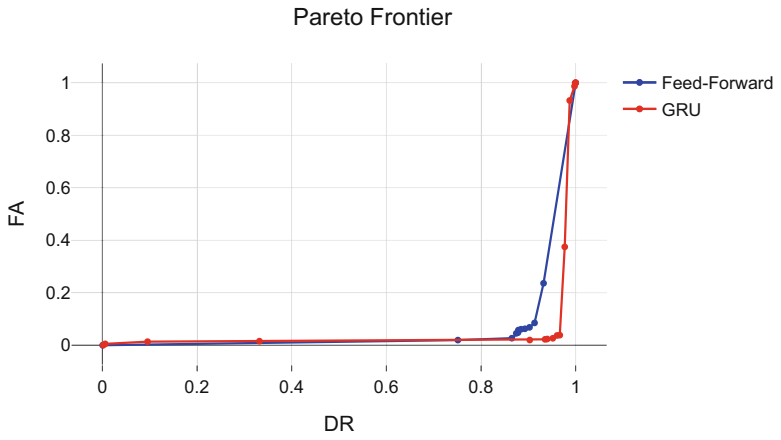


Fig. 8 Pareto frontier results

Table 6 Average performance of the generalized theft detector using genetic-based optimization for hyper-parameters tuning

	FeedForward	GRU	[3]
DR (%)	89.3	**94.2**	94
FA (%)	6.9	**4.7**	11
HD (%)	82.4	**89.5**	83

hyper-parameters model, which outperforms the proposed detector in [3] by a 6.5% increase. Moreover, the FA is improved by almost 57% and the DR is improved by 0.2%. The results indicate that an improvement in the electricity theft detection performance can be achieved using RNN architecture along with the genetic-based hyper-parameters tuning.

7 Conclusions and Future Research

7.1 Conclusions

In this chapter, we proposed two deep neural models for electricity theft detection. Unlike existing research that uses shallow machine learning models such as SVM, our models use deep feed-forward and deep recurrent neural networks. In addition, an analysis on using either a customer-specific detector trained for each customer or a generalized detector trained on the data from multiple customers was carried out. Moreover, an extensive analysis for optimizing the hyper-parameters of the proposed detectors was conducted. More specifically, sequential grid search, random selection, and multi-objective genetic algorithm. Load profiles comprising the electricity consumption of 3, 639 real residential customers over two years are the basis for the experiments. Our extensive performance evaluations including a comparison with a benchmark detector used in the literature have demonstrated that our detectors are efficient in detecting electricity theft cyber attacks. In particular, we draw the following conclusions as an answer to the questions presented earlier in this chapter.

- First, should we build a customer-specific or generalized detectors? generalized detectors are more robust against contamination attacks and does rely on a specific customer historical data. In addition, as shown in the results the sacrifice in the performance is not that much between customer-specific and generalized detector.
- Second, for the architecture of the detector should it be a simple feed-forward or recurrent neural network (RNN)? We recommend the use of the GRU recurrent detector over the feed-forward detector. The recurrent model exploits the time series feed-forward nature of the customers' energy consumption and can learn complex patterns of customers consumption which yields to better detection performance as shown in the results.

- Third, which algorithm is better for hyper-parameters tunning: sequential, random, or genetic-based optimization? One of our main contributions in this work is to express the hyper-parameters optimization problem as a two objectives optimization problem, where the aim is to maximize the detection rate and minimize the false alarm rate and to use multi-objective genetic algorithm NSGA-II to solve it which approaches our third question. Genetic-based optimization for hyper-parameters is the much more complex and time-consuming for training over sequential and random search. However, the mutual impact "interaction" of hyper-parameters on each other is considered while exploring the search space. In addition, NSGA-II offers leverage over blind search where the detector performance is considered while exploring the search space which improves the detection performance. Despite this fact, sequential and random search optimization methods also offered reasonably good performance.

7.2 Future Research

Many different improvements, inspections, and experiments can build on the proposed work. In the future, we plan to apply the proposed detectors on different real datasets. Besides, more neural networks architectures should be investigated, where new types of hyper-parameters should be considered. Another area of investigation is the genetic-algorithm used for the hyper-parameters tuning process; many multi-objective genetic algorithms exist in the literature each also has its parameters that should be selected and studied carefully.

Acknowledgements This publication was made possible by NPRP grant # NPRP9-055-2-022 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Buevich M, Zhang X, Schnitzer D, Escalada T, Jacquiau-Chamski A, Thacker J, Rowe A (2015) Short paper: microgrid losses: when the whole is greater than the sum of its parts. In: Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, pp 95–98. ACM
2. Viegas JL, Esteves PR, Melício R, Mendes VMF, Vieira SM (2017) Solutions for detection of non-technical losses in the electricity grid: a review. Renew Sust Energ Rev 80:1256–1268
3. Jokar P, Arianpoo N, Leung VCM (2016) Electricity theft detection in AMI using customers' consumption patterns. IEEE Trans Smart Grid 7(1):216–226
4. Singh SK, Bose R, Joshi A (2017) Entropy-based electricity theft detection in AMI network. IET Cyber-Phys Syst Theory Appl 3:99–105
5. Antmann P (2009) Reducing technical and non-technical losses in the power sector. World Bank, Washington, DC
6. IBM (2012) Energy theft: incentives to change. Technical report

7. CBC Electricity theft by B.C. Grow-ops costs 100M a year. Online https://www.cbc.ca/news/canada/british-columbia/electricity-theft-by-b-c-grow-ops-costs-100m-a-year-1.969837. Last accesed Nov 2017
8. Pickering P E-meters offer multiple ways to combat electricity theft and tampering. Online https://www.electronicdesign.com/meters/e-meters-offer-multiple-ways-combat-electricity-theft-and-tampering. Last accesed Nov 2017
9. Abaide AR, Canha LN, Barin A, Cassel G (2010) Assessment of the smart grids applied in reducing the cost of distribution system losses. In: 2010 7th International Conference on the European Energy Market (EEM). IEEE, pp 1–6
10. Nizar AH, Dong ZY, Wang Y (2008) Power utility nontechnical loss analysis with extreme learning machine method. IEEE Trans Power Syst 23(3):946–955
11. Nagi J, Yap KS, Tiong SK, Ahmed SK, Nagi F (2011) Improving SVM-based nontechnical loss detection in power utility using the fuzzy inference system. IEEE Trans Power Delivery 26(2):1284–1285
12. Ramos C, de Sousa A, Papa J, Falcao X (2011) A new approach for nontechnical losses detection based on optimum-path forest. IEEE Trans Power Syst 26(1):181–189
13. Angelos E, Saavedra O, Cortés C, de Souza A (2011) Detection and identification of abnormalities in customer consumptions in power distribution systems. IEEE Trans Power Delivery 26(4):2436–2442
14. Lin C-H, Chen S-J, Kuo C-L, Chen J-L (2014) Non-cooperative game model applied to an advanced metering infrastructure for non-technical loss screening in micro-distribution systems. IEEE Trans Smart Grid 5(5):2468–2469
15. Amin S, Schwartz GA, Cardenas AA, Sastry SS (2015) Game-theoretic models of electricity theft detection in smart utility networks: providing new capabilities with advanced metering infrastructure. IEEE Control Systems 35(1):66–81
16. Zhou Y, Chen X, Zomaya AY, Wang L, Hu S (2015) A dynamic programming algorithm for leveraging probabilistic detection of energy theft in smart home. IEEE Trans Emerg Top Comput 3(4):502–513
17. Liu Y, Hu S (2015) Cyberthreat analysis and detection for energy theft in social networking of smart homes. IEEE Trans Comput Soc Syst 2(4):148–158
18. Jindal A, Dua A, Kaur K, Singh M, Kumar N, Mishra S (2016) Decision tree and SVM-based data analytics for theft detection in smart grid. IEEE Trans Ind Inf 12(3):1005–1016
19. Bhat RR, Trevizan RD, Sengupta R, Li X, Bretas A (2016) Identifying nontechnical power loss via spatial and temporal deep learning. In: Proceedings of 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp 272–279
20. Zhan T-S, Chen S-J, Kao C-C, Kuo C-L, Chen J-L, Lin C-H (2016) Non-technical loss and power blackout detection under advanced metering infrastructure using a cooperative game based inference mechanism. IET Gener Transm Distrib 10(4):873–882
21. Tariq M, Poor HV (2016) Electricity theft detection and localization in grid-tied microgrids. IEEE Trans Smart Grid
22. Xiao F, Ai Q (2017) Electricity theft detection in smart grid using random matrix theory. IET Gener Transm Distrib
23. Dineshkumar K, Ramanathan P, Ramasamy S (2015) Development of ARM processor based electricity theft control system using GSM network. In: 2015 International Conference on Circuit, Power and Computing Technologies (ICCPCT). IEEE, pp 1–6
24. Ngamchuen S, Pirak C (2013) Smart anti-tampering algorithm design for single phase smart meter applied to AMI systems. In: 2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). IEEE, pp 1–6
25. Khoo B, Cheng Y (2011) Using RFID for anti-theft in a chinese electrical supply company: a cost-benefit analysis. In: Wireless Telecommunications Symposium (WTS). IEEE, pp 1–6
26. Henriques HO, Barbero APL, Ribeiro RM, Fortes MZ, Zanco W, Xavier OS, Amorim RM (2014) Development of adapted ammeter for fraud detection in low-voltage installations. Measurement 56:1–7

27. Devidas AR, Ramesh MV (2010) Wireless smart grid design for monitoring and optimizing electric transmission in india. In: 2010 Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM). IEEE, pp 637–640
28. Zheng Z, Yang Y, Niu X, Dai H-N, Zhou Y (2018) Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids. IEEE Trans Ind Inf 14(4):1606–1615
29. Devices A (2010) Single-phase multifunction metering IC with di/dt sensor interface. MARUTSU, Norwood
30. Bandim CJ, Alves JER, Pinto AV, Souza FC, Loureiro MRB, Magalhaes CA, Galvez-Durand F (2003) Identification of energy theft and tampered meters using a central observer meter: a mathematical approach. In: Transmission and Distribution Conference and Exposition, 2003 IEEE PES, vol 1. IEEE, pp 163–168
31. Lo C-H, Ansari N (2013) Consumer: a novel hybrid intrusion detection system for distribution networks in smart grid. IEEE Trans Emer Top Comput 1(1):33–44
32. Faria LT, Melo JD, Padilha-Feltrin A (2016) Spatial-temporal estimation for nontechnical losses. IEEE Trans Power Delivery 31(1):362–369
33. Nagi J, Yap KS, Tiong SK, Ahmed SK, Mohamad M (2010) Nontechnical loss detection for metered customers in power utility using support vector machines. IEEE Trans Power Delivery 25(2):1162–1171
34. Irish Social Science Data Archive. Online https://www.ucd.ie/issda/. Last accesed Nov 2017
35. He H, Bai Y, Garcia EA, Li S (2008) Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of IEEE International Joint Conference on Computational Intelligence, pp 1322–1328
36. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
37. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
38. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(Feb):281–305
39. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
40. Eschenauer H, Koski J, Osyczka A (2012) Multicriteria design optimization: procedures and applications. Springer Science & Business Media, New York
41. Pennington J, Schoenholz S, Ganguli S (2017) Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In: Advances in Neural Information Processing Systems, pp 4788–4798

Using Convolutional Neural Networks for Classifying Malicious Network Traffic

Kyle Millar, Adriel Cheng, Hong Gunn Chew, and Cheng-Chew Lim

Abstract As the reliance on the Internet and its constituent applications increase, so too does the value in exploiting these networking systems. Methods to detect and mitigate these threats can no longer rely on singular facets of information, they must be able to adapt to new threats by learning from a diverse range of information. For its ability to learn complex inferences from large data sources, deep learning has become one of the most publicised techniques of machine learning in recent years. This chapter aims to investigate a deep learning technique typically used for image classification, the convolutional neural network (CNN), and how its methodology can be adapted to detect and classify malicious network traffic.

Keywords Convolutional neural networks · Deep learning with GPUs · Malware classification and detection · Analysis and similarity

This research is supported by the Australian Government Research Training Program (RTP) Scholarship and the Commonwealth of Australia as represented by the Defence Science & Technology Group of the Department of Defence. The experiments described herein were enabled by the supercomputing resources provided by the Phoenix HPC service at the University of Adelaide.

K. Millar (✉) · H. G. Chew · C.-C. Lim
School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, Australia
e-mail: kyle.millar@adelaide.edu.au; honggunn.chew@adelaide.edu.au; cheng.lim@adelaide.edu.au

A. Cheng
School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, Australia

Cyber and Electronic Warfare Division, Defence Science & Technology Group, Adelaide, Australia
e-mail: adriel.cheng@adelaide.edu.au; adriel.cheng@dst.defence.gov.au

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_5

1 Introduction

The interest in deep learning over traditional machine learning algorithms originates from its ability to generate complex representations of the input data without the need to hand-select a feature set. This process of making complex decisions directly from the source of information has labelled deep learning an 'end-to-end' algorithm [1]. Often, conventional learning techniques may inadvertently discard useful knowledge if subjected to a feature selection process based on the heuristics of a human operator. Deep learning aims to draw new inferences about a data source, in our case Internet Protocol (IP) network traffic, that might not have been used otherwise.

This new wave of deep learning research has been largely popularised by the field of image classification. The ubiquity of deep learning in image classification was sparked in 2012 when a deep convolutional neural network (CNN) won the ImageNet visual recognition challenge, almost halving the current error rate at the time [2]. Since then, almost all subsequent submissions to the ImageNet challenge have been made using a variation of these CNN learning techniques [3].

Convolutional neural networks are a variant of artificial neural networks which were designed to leverage insights from locally connected features in the input. Although largely used in the field of image classification, CNNs have been successfully applied to other application areas such as, natural language processing [4–7], audio classification [8], and signal processing [9].

Convolutional neural networks have recently gained interest in Cybersecurity and network management research. They have been largely employed to detect and classify the diverse range of traffic seen on present-day networks. Convolutional neural networks have shown promising results in this field, able to perform both statistical-analysis, by interpreting traffic metadata features [10, 11], and facilitate payload-analysis by finding patterns in the message content of the traffic [1, 12, 13]. Despite the potential of CNNs, there has yet to be an in-depth study on the principles of designing and implementing a CNN for use in network traffic classification.

The aim of this chapter is to introduce the theory behind CNNs with an emphasis on why its distinct properties make it well suited to detect evasive malicious network traffic. In particular, the contributions of this chapter are as follows:

- We outline three benefits of utilising CNNs and examine how these properties could benefit or hinder network traffic classifications in a Cybersecurity context.
- We introduce a novel way of representing network traffic to a deep learning classifier inspired by natural language processing, termed Flow-Image.
- We compare three different CNN architectures including a novel architecture aimed at exploiting the known properties of a TCP/IP packet.
- We explore the structure of network traffic and propose how CNNs could be employed to create a classifier that is robust to certain malicious evasion techniques.

This chapter assumes the reader has prior knowledge and background in neural networks and their terminology; otherwise, we refer the reader to [14] for an in-depth introduction to this topic.

The remainder of this chapter is as follows. Section 2 introduces the key concepts and advantages behind a CNN implementation. The advantages of a CNN based implementation are then explored in terms of their application to malicious network traffic classification in Sect. 3. Section 4 outlines how network traffic has been represented to CNNs in related literature, providing an additional novel representation used to address the issues found with existing methods. Section 5 examines three CNN architectures and details how they can be used for network traffic classification.

Section 6 provides a brief summary of the experimental setup used for analysis in this chapter. Section 7 presents experimental results and describes how the different properties of CNNs can affect the classification of malicious network traffic. The chapter concludes in Sections 8 and 9 including suggestions for future research.

2 Convolutional Neural Networks

The underlying assumption of a CNN is that information in the input can be locally correlated [15]. For example, when reading a sentence, it is not just the individual letters that are relevant but the relationship between them. This relationship is often rapidly diminished by distance, with the correlation between letters in one word having little to no effect on the words that follow. Finding local correlations is often much more efficient than analysing the entire input at once. In artificial neural networks, these local correlations can be enforced by using local receptive fields. **Local receptive fields** restrict the inputs to a node to a small localised region of its input. Figure 1 depicts how a local receptive field can be used to restrict the input of a node to a subsection of the previous layer.

There are several advantages to utilising local receptive fields. Firstly, the entire input does not have to be evaluated at once. This is particularly important when working with large input sizes where, in classical deep learning techniques such as a fully-connected neural network, the size of the neural network scales significantly with the number of input fields used. Large neural networks result in larger hardware requirements and can cause overfitting if there are too few samples to train on [16].

The second advantage occurs when there is translation or repetition of identifying features in the input. An example of this advantage is prevalent in object detection tasks where an object should be just as easily detected no matter where in the image it occurs. Once a local correlation is identified, local receptive fields act as a search function, finding that same correlation regardless of its location in the input; therefore, if a local correlation is either shifted or repeated, no additional complexity needs to be added to the neural network to identify it. This is known in artificial neural networks as **weight-sharing**. Depending on the application, weight-sharing can vastly reduce the size of the neural network needed and can help increase its ability to generalise to new data [16].

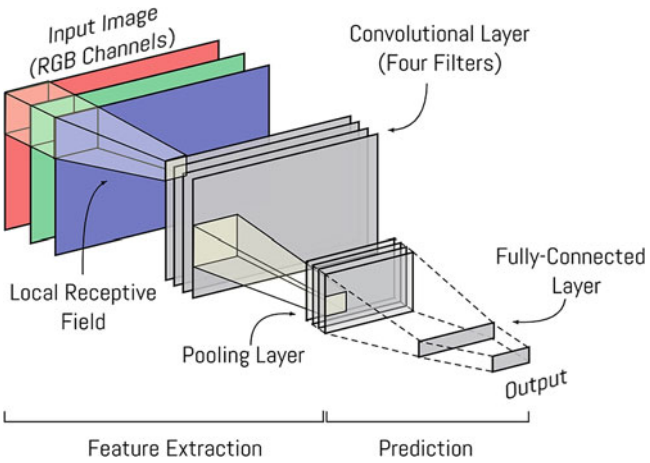


Fig. 1 The structure of a typical CNN used for image classification. The convolutional and pooling layers form the feature extraction stage while the fully-connected layers form the prediction stage

The third advantage of local receptive fields is that they are more robust to localised noise. In many classification tasks the input is likely to suffer from a factor of noise. A system that evaluates solely based on the overall input is likely brittle to these occurrences; however, it is more likely that only a part of the input is affected. Utilising local receptive fields can help localise the effect of noise to just the regions in which it occurs [5].

Convolutional neural networks are simply neural networks that enforce the use of local receptive fields. This is achieved using two specialised layers, the **convolutional** and **pooling** layers. A typical CNN is made up of two stages. The first stage utilises convolutional and pooling layers to identify local correlation within the input and allow for translation invariance. This stage can be thought of as learning and extracting features from the input. The second stage utilises fully-connected layers to combine the features found in the previous stage into an overall prediction of which class the input most likely belongs to. Figure 1 shows a graphical representation of a typical CNN structure used for image classification.

In the remainder of this section, each of these layers are briefly introduced as prerequisites for later sections of the chapter; for a more in-depth explanation of CNNs and their structure, the reader is referred to [15, 17].

2.1 Convolutional Layer

The convolutional layer is the core building block of a CNN. A convolutional layer enforces the idea of local receptive fields by limiting the scope of each node in the layer to a select few nodes in the preceding layer.

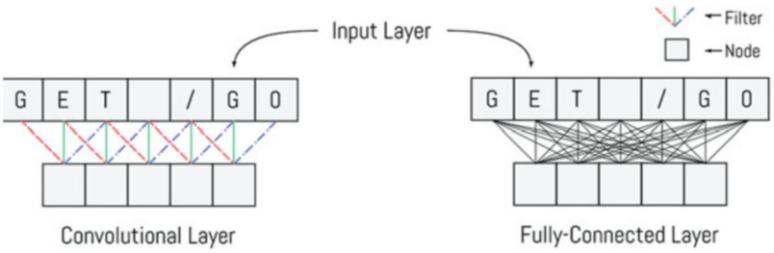


Fig. 2 A comparison of the connections in a convolutional layer versus a fully-connected layer. Each node in the convolutional layer makes the same three connections to the nodes in the input layer, albeit at different locations. This is known as a convolutional layer with a filter size of three. Each node in the fully-connected layer is connected to every node in the previous layer

A feature of local receptive fields stated in the previous section was that of weight-sharing, the ability to detect repeated or translated patterns without adding additional complexity into the network. A convolutional layer implements this concept using a **filter**, a sliding function used to detect the same pattern in different regions of the input. In Fig. 2, a single filter is depicted as it scans across the input for a particular pattern. To search the input for a varying number of patterns, a convolutional layer will typically comprise of many filters, the total output of which is entitled a feature map**.** The **feature map** can be thought of as identifying the location in which each filter found their respective pattern. The convolutional layer in Fig. 1 depicts a feature map comprising of four filters.

The dimensionality of a CNN refers to the dimensions of the filters in its convolutional layers. One-dimensional (1D) and two-dimensional (2D) CNNs are typically seen in research due the magnitude of tasks in which one or two dimensional arrays are used for inputs. 2D-CNNs are typically used in image classification where the two dimensions relate to the spatial coordinates of an image. 1D-CNNs are used for sequential inputs such as text classification. Figures 1 and 2 show a 2D-CNN and a 1D-CNN respectively. It should be noted that the depth of the filter is not considered as a dimension in this terminology. Figure 1 shows a convolutional layer with a filter depth of three, relating to the three input channels of the image (red, green, and blue).

2.2 Pooling Layer

The pooling layer serves two purposes, (i) it reduces the size of the feature map and (ii) provides a level of translation invariance to the features found therein. The most important values of the feature map are where the filters detect their respective patterns. This is identified by strong positive or negative values in the feature map. The larger the absolute number is, the more likely that the filter has detected a pattern in that location. The pooling layer reduces the percentage of low values in the feature map by summarising subregions by their important values.

Max pooling is the most utilised pooling function. It samples subregions in a similar process to that of the convolutional layer; however, instead of finding patterns in the input, it takes the maximum absolute value in that subregion. This increases the density of important values in the feature map while also allowing for slight translation in the locations of the patterns found. Figure 1 depicts how a pooling layer reduces the size of the feature map of the previous layer.

2.3 Fully-Connected Layer

The fully-connected layers used in a CNN are the same as found in a fully-connected neural network. Each node in this layer is connected to every node in the previous layer. This structure allows for the extraction of global relationships from the previous layer. A CNN uses fully-connected layers to combine the individual features found in the convolutional layers into a final prediction of the overall input.

The disadvantage of this layer is that the number of connections is proportional to the number nodes in the previous layer. This can cause significant computational overhead when input sizes are large; therefore, convolutional and pooling layers are often designed to reduce the number of parameters in the input before a fully-connected layer is used.

3 Characteristics of Network Traffic for CNNs

In this section, we examine the three benefits of CNNs for the classification of network traffic in a Cybersecurity context:

- Efficiency on larger input sizes (Sect. 3.1).
- Translation and repetition invariance (Sect. 3.2).
- Robustness to noise (Sect. 3.3).

3.1 Efficiency on Larger Input Sizes

Network traffic is typically characterised by large volumes of traffic. Packets spanning the maximum transmission unit (MTU) of 1500 bytes and network traffic flows using hundreds or thousands of packets are commonly seen in today's networks. It is therefore necessary to consider how to efficiently evaluate such large input sizes when using neural networks.

The **trainable parameters** of a neural network are the parameters that are tuned while training to reduce its error on the desired objective [14]. The most important trainable parameter is the weight of the connections between nodes. Every

connection in the neural network has an associated weight. The value of this weight is adjusted during training to determine the significance of the connection to the overall classification. A CNN can considerably reduce the number of trainable parameters of a neural network by using weight-sharing. Weight-sharing allows a trained weight value to be shared between multiple connections in the same layer.

In the previous section, the relationship between the inputs to a fully-connected layer and the number of connections in the neural network was highlighted. The computational requirements of maintaining such a large number of trainable parameters for each of these connections would rule out many hardware implementations and could lead to overfitting if there are not enough samples to train on [16].

Figure 2 highlights the comparison between the number of trainable parameters in a convolutional layer versus a fully-connected layer. Although there are 15 connections made in the convolutional layer, due to weight-sharing, only three weights require adjusting. In contrast, a fully-connected layer with the same number of nodes would result in 35 adjustable weights.

The reduction in trainable weight parameters as a result of weight-sharing allows a CNN to be much more computationally efficient than other types of neural networks when dealing with larger input sizes. This result is further analysed in Sect. 7.1.

3.2 Translated and Repeated Patterns

The process of scanning for discrete, local patterns enables a CNN to locate features regardless of where they appear in the input. The benefit of this approach has been shown in speech recognition tasks where the varying speed, tone, and inflection of a speaker can change the positioning of these identifying features [16]. Malicious network traffic has been shown to exploit similar properties in the size, sequencing, and timing of packets to obfuscate the identifying features of malicious intent [18].

In allowing for variations in the input, a CNN can be robust to certain evasion techniques. Table 1 shows an example of how different string manipulations could be used to extract the same information. All three commands wish to extract information contained in a file stated as `/_vti_pvt/administrators.pwd`. In order to learn to detect this command, a fully-connected neural network would have to learn each of these representations individually. In contrast, a CNN could reuse the same patterns it has found in its lower levels of abstraction to quickly generalise to new variants.

Table 1 Examples of simple string manipulation [19]

Method	Example
Self-referencing	`/_vti_pvt/./././././administrators.Pwd`
Double slashes	`/_vti_pvt///administrators.Pwd`
Reverse traversal	`/scripts/../_vti_pvt/administrators.Pwd`

Weight-sharing allows a CNN to identify repetition in the input without adding any additional complexity to the neural network. From a network traffic perspective, similar syntax structures and duplicated packets make repeated patterns in data and communication traffic highly prevalent.

3.3 Noisy Inputs

In many classification tasks, the input is likely to suffer from a factor of noise; however, what is noise in terms of malicious network traffic is hard to define. On one hand, randomly generated strings could be injected into a packet to confuse the classifier's decision process; but, on the other hand, if these randomly generated strings are known to be classified as noise, they could be used to hide malicious content.

The use of local receptive fields allows a CNN to classify an input even when certain sections of the input have been obfuscated by noise [5]. This property would allow a CNN to be robust to certain injection attacks where noise is added to confuse a detection system; however, this property could also be exploited. The convolutional layers of a CNN scan the input for identifying features. If the features of malicious intent are not known, it is unlikely that any warnings would be raised; however, a detection system which accounts for the entire input might see a change in the overall input and raise suspicion. A CNN's ability to be robust to noise could therefore both benefit and hinder the classification of malicious network traffic, depending on the evasion techniques utilised.

4 CNN Input Representations

In this section, we investigate how network traffic can be represented to a CNN. In particular, this section evaluates the current methods of representation in related literature and introduces a novel approach that aims to improve the efficacy of using a CNN for network traffic classification.

4.1 Related Work

Network traffic classification typically falls into two categories; (i) statistical-based classification, where the traffic is classified based on a collection of metadata features, such as the number of packets sent; and (ii) payload-based classification, where the traffic is classified based on the distinct signatures found in the message content of the packet [13].

Statistical-based approaches were explored using a CNN in [11]; however, no prior consideration was undertaken as to what the best representation should be to fully exploit the strengths of a CNN. In this paper, we address this directly and strive to utilise the full potential of a CNN.

Wang et al. [1, 12] explored the use of a CNN for payload-based classification in which a 28×28 pixel image was used to represent the first 784 bytes of the payload content of a traffic flow. Each byte was represented by a greyscale pixel using the range from black (represented by the value 0) to white (represented by the value 255). This provides a one-to-one mapping with the 256-value range of a byte field.

Wang et al. showed that a 1D-CNN achieved better classification accuracy than a 2D-CNN on this input representation. Although not stated by the authors, this result is likely to have stemmed from how the second dimension of the image was created. The payload was wrapped to form the second dimension of the image; therefore, this dimension was simply an extension of the first. Analysing a one-dimensional input with a 2D-CNN is likely to reduce the classifier's ability to generalise to new data and may have caused the reduction in accuracy reported.

The value of a pixel in an image is numerical. In greyscale, this value represents a scale between black and white. Representing a byte as a greyscale pixel infers the same scale between its values. For example, implying that a byte value of zero has more similarity to a value of one than it does to ten. While this representation was shown to hold for a test data set in [1, 12], it is not always a true property of the values in a byte for different types of network traffic. Obfuscation techniques of malicious traffic aim to exploit the difference in how a detection system and its intended target process data. Using these differences, a malicious attack can hide its true intention from the detection system while still delivering its malicious payload to its target. A numerical representation of bytes in a detection system could be easily exploited as it is using a property that is non-existent in the intended application.

To address the above issues, a novel approach to network data representation was explored as part of this research. This input representation will be referred to as Flow-Image for the remainder of the paper.

4.2 Flow-Image

Network traffic is fundamentally a conversation between two endpoints; therefore, rather than taking inspiration from image classification as per the related work above, we seek inspiration from natural language processing (NLP).

Text classification is one of the cardinal topics for NLP in which text is classified based on a predefined objective such as sentiment analysis (i.e., identifying whether the writer's opinion is positive or negative) or terminology extraction (i.e., identifying important words in a larger body of text). These methods typically make use of feature generation techniques, such as bag-of-words, to represent language in higher order representations before analysis [7]. This however requires predefined knowledge of the syntax of the language you are classifying.

To classify malicious network traffic, assumptions based on syntactic or semantic structures cannot be made as malicious attacks can exploit these assumptions to circumvent known techniques. Fortunately, CNNs have recently been investigated to perform classification without any predefined knowledge of the subject language [6, 7]. This is achieved by analysing the text at its fundamental level, i.e. its individual characters.

Character-level NLP allows the classifier to learn the complete formation of the text directly from its underlying characters. While this adds additional complexities to the network, it allows for the classification of text when the structure of the language is unknown. Zhang et al. [7] showed that CNNs can be successfully applied to character-level NLP tasks, achieving comparable results to the state-of-the-art techniques for English text classification.

To represent characters to a neural network, a categorical representation must be used as there is no intrinsic correlations between the values of a character. **One-hot-encoding** is a common way to represent categorical information as it is simple to implement. However, the more categories used, the less efficient this method becomes. One-hot-encoding constructs a vector of size n, where n is the number of distinct categories an input value can take. Each entry in this vector is zero except for the entry that corresponds to the category that this vector represents, which is given a value of one. For example, in network traffic, to represent the byte value '0', a vector of size 256 would be constructed with the first entry set to one and the rest zero. One-hot-encoding was shown in [7] to effectively represent 70 different characters to a CNN. This method will be used in Flow-Image to represent the 256 different byte values.

Current implementations of CNNs require a fixed input size. Therefore, the number of bytes analysed in each flow must be set before analysis. Furthermore, to investigate the effect of a 2D-CNN on network traffic classification, the input must also have two representative dimensions. To address these two factors, a network traffic flow was represented in a two-dimensional array as shown in Fig. 3. Each row of this array represented a new packet in the flow with each column representing a

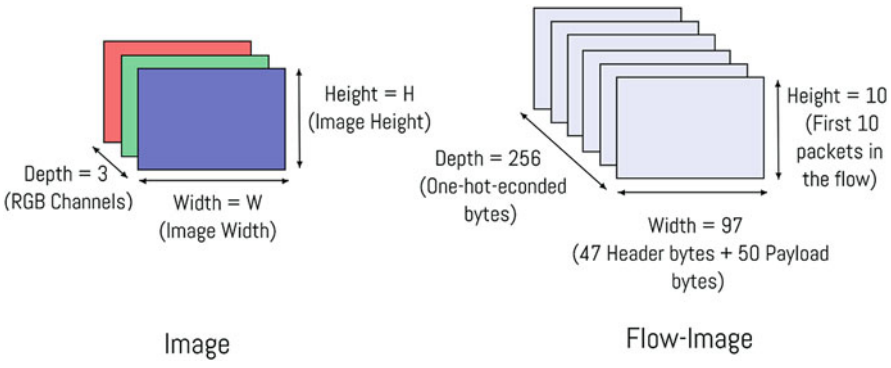


Fig. 3 Representation of an input used for image classification compared to Flow-Image

new byte in the packet. The array was fixed such that 97 bytes of the first ten packets of a bidirectional flow were analysed. Flows that did not meet the required array size were padded with byte values of zero.

Composing the 97 bytes of a packet were 47 bytes from the TCP/IP header and 50 bytes from the payload. The header section was included in the Flow-Image as some attacks are easier to detect by their header sections (i.e., scans and probes), while others are easier to detect by their payload (i.e., worms) [20].

The number of bytes in the TCP/IP header section was limited to 47 as this was found to be the maximum header length for over 99% of the packets in the data set used for analysis (further details of the data set are given in Sect. 6.1). The 47-byte header encompassed an 11-byte IP header (source and destination IP addresses, and TTL fields were removed due to synthetic biases found in the data set) and a 36-byte TCP header.

The payload was truncated to the first 50 bytes of each packet as it has been shown that the classification significance of a byte in the payload decreases the further into the payload it is found [21–23]. The number of flows in the Flow-Image was truncated to ten to balance classification efficiency with the ability to analyse the relationship between packets in a flow.

It should be noted that the assumptions on the size and number of packets to evaluate were based on analysis of our data set. These assumptions may not hold under other circumstances. A malicious attack could easily circumvent detection by sending malicious content outside the limits of the Flow-Image stipulated in this chapter. Before a detection system is deployed, a decision should be made between the efficiency of the system and its robustness to attacks looking to exploit this property.

To represent bytes in a one-hot-encoded vector, a third dimension to this array was added. Figure 3 shows a graphical representation of this Flow-Image in contrast to how a typical image would be represented to a neural network.

5 CNN Architectures

In this section, we explore three CNN architectures and examine how the properties of CNNs can be exploited to improve the detection and classification of evasive malicious network traffic. The configuration selected for each CNN architecture is shown in Table 2.

5.1 2D-CNN

A 2D-CNN is the most common architecture due the abundance of CNNs used for image classification. Images consist of two dimensions which both relate to the same spatial coordinate system; therefore, local correlations that make use of both dimensions must exist. A 2D-CNN is able to exploit this property by scanning for multi-dimensional correlations.

Table 2 Structure of the CNN architectures

Layers	2D	1D	Segmented-CNN	
			Header	Payload
1	Conv[3 × 3]	Conv[1 × 3]	Conv[1 × 47]	Conv[1 × 3]
2	Conv[3 × 3]	Conv[1 × 3]	Conv[1 × 1]	Conv[1 × 3]
3	Pool[2 × 2]	Pool[1 × 2]	Pool[2 × 1]	Pool[1 × 2]
4–5	Conv[3 × 3]	Conv[1 × 3]	Conv[1 × 1]	Conv[1 × 3]
6	Pool[2 × 2]	Pool[1 × 2]	Pool[2 × 1]	Pool[1 × 2]
7	FC[256]	FC[256]	FC[256]	FC[256]
8	Fc[1024]	Fc[1024]	-	Fc[1024]
Output	FC[6]	FC[6]	FC[6]	

Each convolutional layer consists of 256 filters. The output layer consists of a node for each class within the data set, this layer is used to designate the predicted class. Max pooling is used for all pooling layers

Conv[$h \times w$] – Convolutional layer with a filter height and width of h and w respectively

Pool[$h \times w$] – Pooling layer with a filter height and width of h and w respectively

FC[n] – Fully-connected layer with n nodes

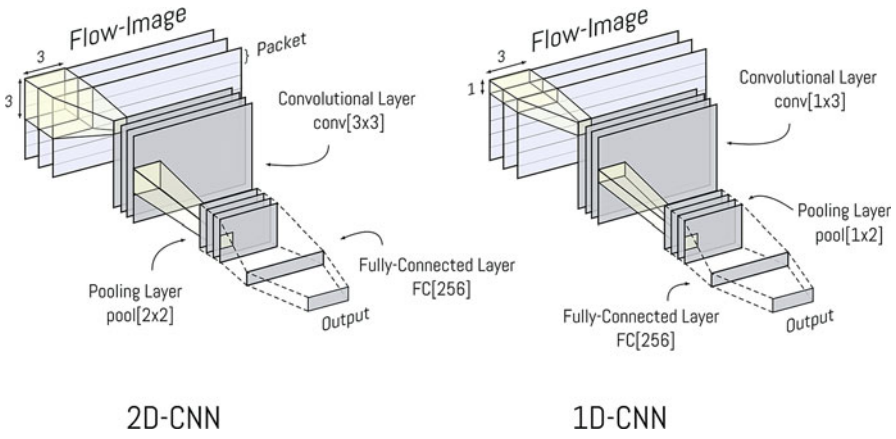


Fig. 4 Structure of a 2D-CNN and 1D-CNN. The filters of a 2D-CNN span multiple packets, while a 1D-CNN's filters only consider the bytes of a single packet. Only the first convolutional, pooling, and fully-connected layers in the 2D-CNN and 1D-CNN are shown

The Flow-Image input has two representative dimensions. The first represents packets in the flow and the second represents bytes in a packet. A 2D-CNN could be utilised to find locally correlated bytes both within the same packet and in adjacent packets. Network traffic typically emulates a call and response within its communication. Most packets are followed by a response or acknowledgement from the other endpoint. Analysing locally correlated packets may yield information into how the two ends of the connection are communicating. Figure 4 depicts the key properties of the 2D-CNN architecture as it evaluates the Flow-Image input.

5.2 1D-CNN

1D-CNNs have been typically used in NLP tasks where information can be represented as a sequential one-dimensional array. Although there are two representative dimensions in the Flow-Image input, the bytes between a packet have a higher degree of correlation than the bytes over multiple packets. A 1D-CNN could be used to exploit this property of network traffic. A diagram of the 1D-CNN architecture is shown in Fig. 4.

5.3 Segmented-CNN

A Segmented-CNN is a novel CNN architecture that aims to capitalise on the distinct properties of the two sections in a TCP/IP packet, the header and the payload. This architecture is inspired by the approach taken by Bromley et al. in their design of a Siamese-CNN [24]. A Siamese-CNN utilises two CNNs to simultaneously analyse two distinct images, the output of which is then compared to determine the similarity between the two. However, instead of measuring the similarity between these two images, this methodology could be adapted to analyse two inputs that relate to the same classification but have their own distinct properties. Figure 5 shows how two CNNs could be used to analyse the distinct sections of a TCP/IP connection.

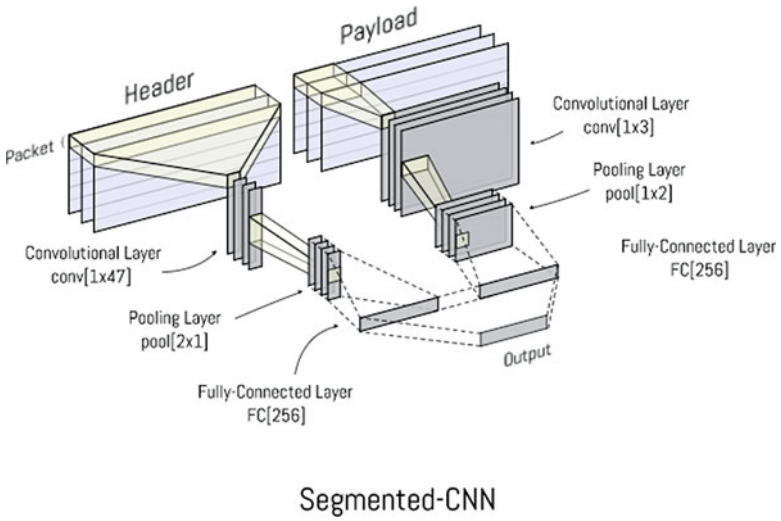


Fig. 5 Structure of a Segmented-CNN showing how the two CNN networks can be individually optimised for the two distinct regions of a packet, the header and payload. Only the first convolutional, pooling, and fully-connected layers of the Segmented-CNN have been shown

The header section of a packet relates to the information needed for it to traverse a network. It has a strict structure that needs to be enforced such that packets can be received intact at the other end of the connection; therefore, there is very little variance in how a header section can be constructed. Furthermore, as most values of this input relate to a distinct property of the packet, there is almost no local correlation within the header section. The classification of the header section of a packet is therefore more suited to a fully-connected neural network; however, multiple headers in a flow are likely to exhibit similar properties and therefore benefit from the utilisation of weight-sharing. To optimise a CNN for these two properties, a one-dimensional filter which spans the entire header section of a packet could be used. At the packet level, this filter acts as a fully-connected neural network; however, the use of a one-dimensional filter allows for weight-sharing between the packets.

The payload is the message content of a TCP/IP packet; therefore, a CNN structure inspired by natural language processing should be used. The 1D-CNN (Sect. 5.2) was selected to be used for the payload section of the Segmented-CNN.

6 Experimental Setup

6.1 Data Set

To evaluate CNNs on network traffic, a data set that contained full packet captures was required. For this reason the UNSW-NB15 data set [25, 26] was chosen as the subject of our experiments. Contained in this data set was a set of nine malicious classes; however, as deep learning requires a large number of samples to generate an accurate prediction, four of the smallest malicious classes were merged into a single class entitled OTHER-MALICIOUS.

The UNSW-NB15 data set was split into three subsets, training (72%), validation (8%), and testing (20%). The malicious classes used in our experiments and their distributions within each subset are presented in Table 3.

Table 3 Distribution of classes from the UNSW-NB15 data set

Malicious classes	Training (72%)	Validation (8%)	Testing (20%)	Total
DOS	2592	290	721	3603
EXPLOITS	18,189	2030	5055	25,274
FUZZERS	13,851	1541	3848	19,240
GENERIC	2727	311	760	3798
RECONNAISSANCE	8397	939	2335	11,671
OTHER-MALICIOUS	1611	188	450	2249
NON-MALICIOUS	36,000	4000	10,000	50,000
Total	83,367	9299	23,169	115,835

A non-malicious class was also included such that the classifier's ability to distinguish between malicious attacks and normal traffic behaviour could be investigated. This non-malicious class was made up of traffic likely to be in abundance on a typical network such as HTTP, P2P, and FTP.

6.2 Hardware and Software

The recent rise of deep learning research has been largely spurred by advancements in the specialised hardware required to utilise it. Previously, the sheer number of mathematical operations within a deep learning algorithm reduced its applicability on computationally or temporally limited applications; however, with the introduction of dedicated graphics processing units (GPUs), many of these operations can now be carried out in parallel.

The University of Adelaide's Phoenix High Performance Computing (HPC) service was used to train the neural networks evaluated in this chapter. Each neural network was trained on a NVIDIA Tesla K80 GPU, with 32GB of memory and an 8-core central processing unit (CPU). The neural networks were implemented using Google's deep learning framework, TensorFlow [27].

7 Experimental Results

To evaluate the effectiveness of Flow-Image and our CNN architectures outlined in Sects. 5.1, 5.2 and 5.3, we conduct malicious network traffic classification experiments using the UNSW-NB15 data set. A fully-connected neural network was also analysed to provide a baseline comparison to the CNN techniques. The fully-connected neural network consisted of three fully-connected layers of 256, 1024, and 7 nodes.

7.1 Architecture Efficiency

One of the key benefits of CNNs is their efficiency on large input sizes. To identify what extent this has on the classification of the Flow-Image input, the training time and neural network size of each architecture were examined.

Figure 6 shows the number of trainable parameters in each neural network to the nearest million. The number of trainable parameters is roughly proportional to the size of the network in memory (shown in Fig. 7). The CNN architectures show between three and ten times reduction in the memory requirements of a neural network. A CNN is therefore better suited for any hardware limited implementations.

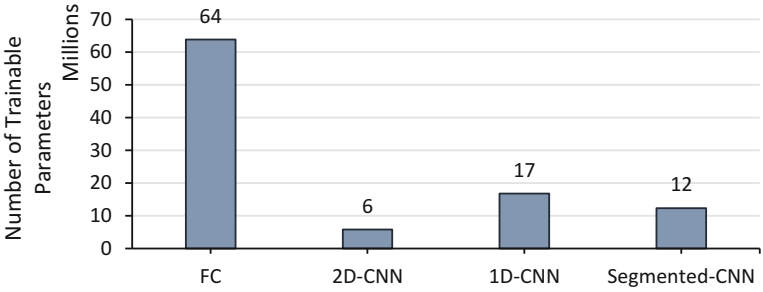


Fig. 6 Number of trainable parameters in each neural network to the nearest million

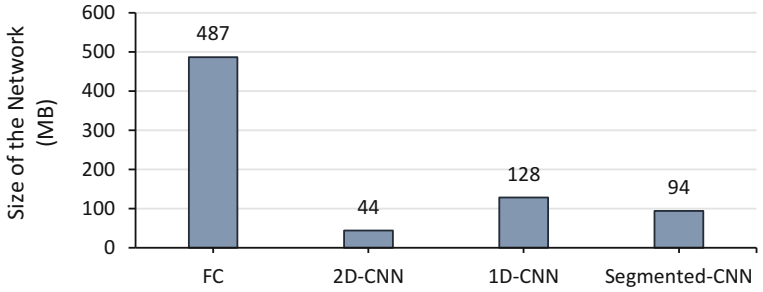


Fig. 7 Approximate size of the neural network in memory

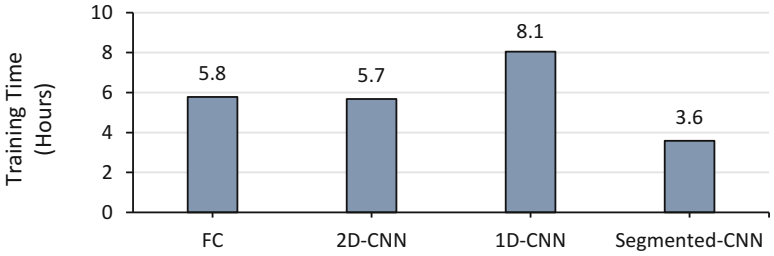


Fig. 8 Training time of each neural network

Figure 8 shows the training time of each neural network. The Segmented-CNN completed training in the shortest amount of time, less than half the training time of the 1D-CNN. The training time of these architectures did not correlate with their number of trainable parameters. This result may be specific to the Flow-Image input representation.

The pre-processing of network traffic to form the Flow-Image input representation allowed for the separation of the header and payload of each packet into distinct sections of the input. The design of the Segmented-CNN therefore may have expediated the training process by continuing this separation of the header

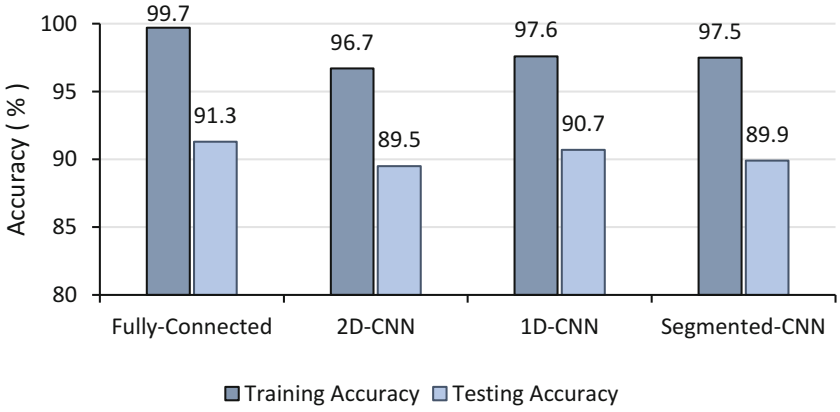


Fig. 9 Training and testing accuracy of different CNN architectures on the UNSW-NB15 data set. A fully-connected neural network was also analysed to provide a baseline comparison to the CNN techniques

and payload sections within the neural network itself. This allowed for two smaller inputs that had a higher degree of similarity to be analysed. This process is akin to that of a divide and conquer approach.

7.2 Classification Performance

The trained classifiers were evaluated on the testing set, the results of which are shown in Fig. 9. All classifiers exhibited similar classification performances on the testing set with the fully-connected neural network achieving the highest classification accuracy overall. The performance of the fully-connected neural network on the testing set suggests that the size of the neural network is sufficient to account for any variation in network traffic in the UNSW-NB15 data set.

The fully-connected neural network achieved almost 100% training classification accuracy on the training set. This result signifies that the classifier was overfitting the training data, supporting our analysis in Sect. 3.1 that even a small fully-connected neural network would likely overfit the Flow-Image input due to the large number of trainable parameters that would be created.

All three CNN architectures shown in Fig. 9 are able to achieve similar test classification performance to the fully-connected neural network albeit with five to ten times fewer trainable parameters. This result suggests that the use of CNNs for network traffic classification would still be recommended over that of a fully-connected neural network when computational resources are limited.

To provide further analysis, the precision and recall values of each class were calculated. **Precision** is the number of correct classifications of a class divided by the number of times that class was predicted. **Recall** is the number of correct

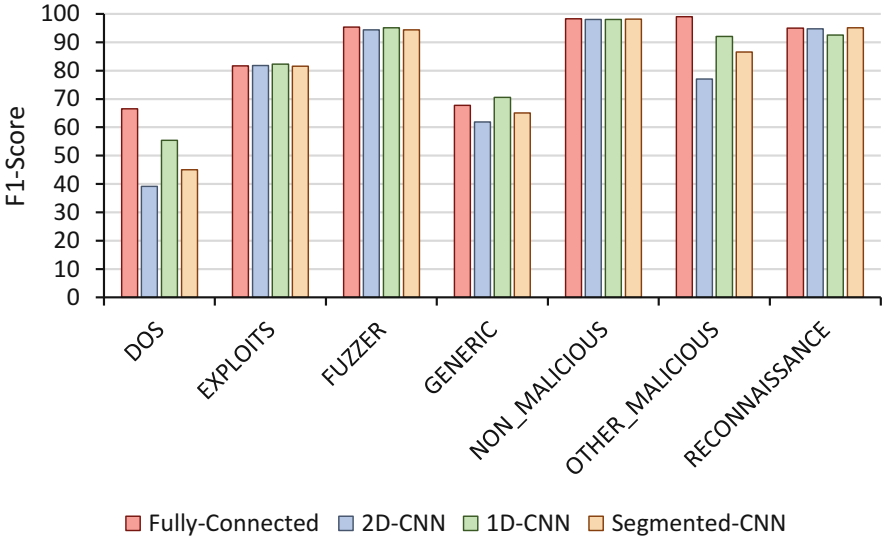


Fig. 10 F1-Score of the individual classes for each classifier on the testing set

classifications of a class divided by the total number of samples of that class. The harmonic average of precision and recall is the F_1-score and is commonly used to compare the performance of a classifier on the individual classes present.

$$F_1\text{-score} = 2\,\frac{(Precision \times Recall)}{(Precision + Recall)} \tag{1}$$

Shown in Fig. 10 is the F_1-score of the individual classes for each classifier on the testing set. From Fig. 10, a pattern in the DOS, GENERIC, and OTHER_MALICIOUS classes was observed. The fully-connected neural network and 1D-CNN achieve the highest accuracies in these classes, followed by the Segmented-CNN and 2D-CNN respectively. This result correlated with the degree of pooling used in each architecture, therefore suggesting that the network traffic flows in the UNSW-NB15 were highly structured. The degree in which the structure of network traffic affects its classification performance is discussed further in the next section.

7.3 Network Traffic Structure

To further investigate how the structure of network traffic affects the classification performance of a neural network, the prediction process of the trained classifiers was evaluated.

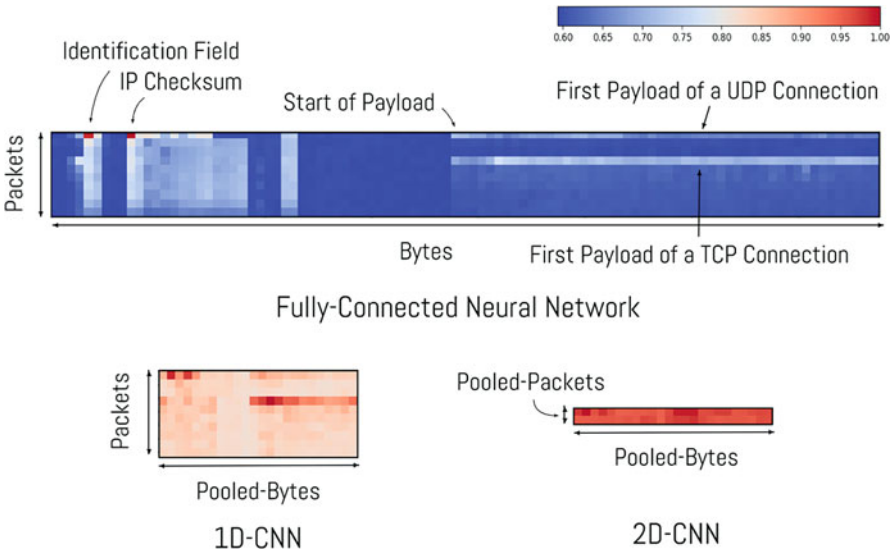


Fig. 11 Three heatmaps showing the important features in the input of the first fully-connected layer of the fully-connected neural network, 1D-CNN and 2D-CNN

To train a neural network, the weights of its connections are adjusted depending on their significance to the overall classification. An estimation of the importance of an input to a neural network can therefore be identified by the summation of the weight of its connections.

The process of identifying significant bytes in a TCP/IP flow by the weight of its connections was explored by Wang [21]. This process will be used to highlight differences in how the fully-connected neural network, 1D-CNN, and 2D-CNN extract information from the Flow-Image input. The Segmented-CNN will not be evaluated within this section as it does not reveal any additional insight into how the predictions of a neural network are made on the Flow-Image. A heatmap depicting the most significant inputs to the first fully-connected layer in each neural network is shown in Fig. 11.

The heatmap of the first layer in the fully-connected neural network indicates the significance of the byte values in the Flow-Image input. From this heatmap, a highly structured pattern is observed; with distinct patterns found both in the order of packets in a flow as well as in the separate header and payload sections of a packet.

Within the header section of the fully-connected heatmap, the utilisation of different header information fields is highlighted. It was shown that the identification field and checksum of the IP header are the two most significant features to the classification of the Flow-Image. These two fields are both unique to their individual packet. It is therefore evident that the neural network is overfitting by learning to identify the individual packets in the training set.

From the payload section of the fully-connected heatmap, it was observed that the neural network identified the first and fourth payload of the flow as most significant to its classification. The first and fourth payload correspond to the first payload content of a user datagram protocol (UDP) and transmission control protocol (TCP) connection respectively. This result therefore suggests that the first payload content of the flow is most significant to the classification of the overall flow. The correlation between the significance of the payload content and its location within the flow has been confirmed by other investigations [21–23]. This result demonstrates a neural network's ability to learn identifying properties of complex inputs without any predefined knowledge of the classification domain.

The heatmap of the first fully-connected layer of the 1D-CNN and 2D-CNN shown in Fig. 11 indicates the significant features learned by the convolutional and pooling layers of their respective neural networks. From these two heatmaps, we observe how the convolutional and pooling layers can be used to increase the density of important features in the input before the utilisation of computationally expensive fully-connected layers.

The 1D-CNN heatmap illustrates how the one-dimensional convolutional and pooling operations will increase the density of important features of an individual packet while preserving the order of packets in the flow. The 2D-CNN further increases the density of important features in the Flow-Image by also applying the convolutional and pooling operations over the packets in a flow.

The input size of the first fully-connected layer in the 2D-CNN is reduced by a factor of 20 when compared to that of the first layer in the fully-connected neural network. This reduction in the number of input values to the first fully-connected layer vastly reduces the size of the overall network as seen in Sect. 7.1. The pooling of packets in the flow however decreases the 2D-CNN's ability to learn identifying features from the order in which packets are sent.

7.4 Transient Features

In the previous section, it was highlighted that a neural network learned to classify the network traffic based on assumptions regarding the order of packets in a flow. In this section, an example of how these assumptions could be exploited by malicious evasion techniques will be investigated.

It was identified that the first payload content of a flow is the most significant to its overall classification; however, this assumption cannot always be relied upon, especially when classifying evasive malicious traffic.

Fragmentation is the process of segmenting the content of a traffic flow into multiple packets. Malicious actors have been shown to evade detection by carefully selecting how fragmentation is performed [19]. A simple way to employ such an evasion technique is by purposely sending packets in the wrong order. A host that receives these packets will utilise the packet's sequence numbers to reassemble them into their correct sequence; however, packet reassembly may not be performed

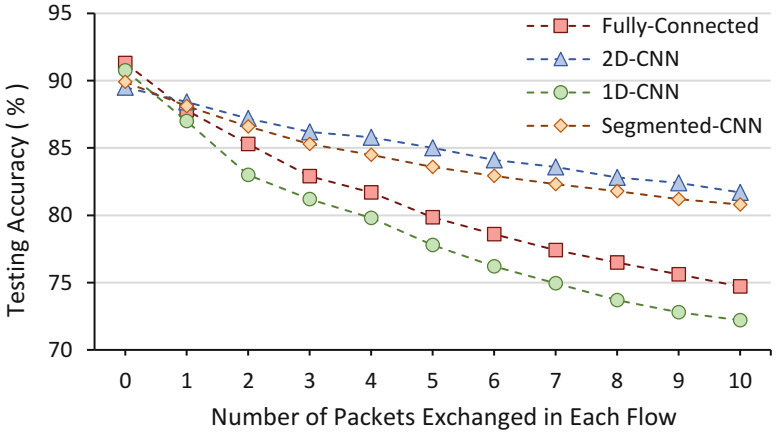


Fig. 12 Accuracy of each neural network on the test set with respect to the number of packets that were exchanged in each flow. Packets were exchanged with the next consecutive packet in the flow

within the intrusion detection system (IDS) itself due to the resultant time delay and computational cost. A malicious actor could therefore change the packet sequence to disguise the overall intent of the traffic flow from the IDS [18].

To simulate the use of a fragmentation evasion technique, the order of packets in the Flow-Image input representation was progressively altered. The accuracy of the trained neural networks on the UNSW-NB15 test set was then re-evaluated. A varying number of packets were chosen for re-ordering to identify how much variance in the order of packets would affect the overall traffic classification. The packets that were chosen for re-ordering were random and simply exchanged with the next consecutive packet within the flow. The accuracy of each neural network on the altered test set is plotted in Fig. 12 with respect to the number of packets that were exchanged in each flow.

The classification performance of each neural network decreases as the order of packets is progressively altered; however, classification degradation is less severe in the 2D-CNN and Segmented-CNN. These two neural networks both employ a pooling operation across multiple packets; thereby reducing their reliance on a specific order of packets to correctly classify the flow.

8 Conclusions

In this chapter, the benefits of using a CNN for malicious network traffic classification were explored. A CNN-based approach was shown to achieve similar classification accuracies to a fully-connected neural network at one tenth of the fully-connected neural network's size. This result confirms that a CNN-based solution would be better suited for hardware limited implementations.

The Flow-Image input representation was presented as a novel way to address the issue of using numerical representation for categorical data as seen in the related literature. A numerical representation of byte values in an IDS could be easily exploited as it classifies traffic using properties that may be non-existent in the intended application.

The Segmented-CNN was proposed as a novel CNN architecture that can exploit the distinct sections of a TCP/IP packet, the header and payload. The Segmented-CNN was shown to achieve a faster training time due the use of a divide and conquer approach.

The structure of network traffic and its effect on the classification performance of the evaluated neural networks was identified using a heatmap of the significant input values. The order of packets in a traffic flow was found to be highly influential in the overall classification of a traffic flow.

An example of a simple detection evasion technique was explored by altering the order of packets in a traffic flow. It was shown that the 2D-CNN and Segmented-CNN were more robust to this evasion technique as they employ a pooling operation across multiple packets; thereby reducing their reliance on a specific order of packets to correctly classify the flow.

9 Future Works

In this chapter, the benefit of a CNN to classify noisy data was introduced; however, in network traffic, it is hard to quantify what is noise and what could be a potential malicious threat. Whether this is a benefit or a potential means of evading a CNN based classification algorithm is a subject for further analysis.

Convolutional neural networks require a fixed input size; however, the diversity of network traffic makes it difficult to define what a suitable input size should be. Furthermore, if this input size is known, malicious actors could implement evasion techniques to place malicious code outside the region of analysis. To address this issue, a recurrent neural network (RNN), which allows for variability in the input size, could be combined with a CNN to create a neural network with the mutual benefits of each approach.

The data set used in this chapter for experimental analysis was synthetic. Further evaluation of neural networks on real network traffic captures is required to determine if the performance of these methods hold under real-life conditions.

One of the key contributions of this chapter proposed that CNNs could be employed to counteract certain known types of evasion techniques; however, deep learning techniques can only learn from the existing data they have access to. Techniques such as polymorphic code, encryption, and variations of character encodings would prove difficult to account for without proper consideration. A more robust training method could be implemented by the utilisation of an adversarial network. Specifically, an additional neural network which is trained to produce

inputs that evade the detection of the first neural network. Adversarial networks could therefore be investigated as a way to train neural networks that are more robust to different evasion techniques.

References

1. Wang W, Zhu M, Wang J, Zeng X, Yang Z (2017) End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: 2017 IEEE international conference on Intelligence and Security Informatics (ISI). IEEE, Piscataway, pp 43–48
2. Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2:1097–1105
3. Russakovsky O et al (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
4. Yoshioka T, Karita S, Nakatani T (2015) Far-field speech recognition using CNN-DNN-HMM with convolution in time. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Piscataway, pp 4360–4364
5. Abdel-Hamid O, Mohamed A-R, Jiang H, Penn G (2012) Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In: 2012 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Piscataway, pp 4277–4280
6. Zhang X, LeCun Y (2017) Which encoding is the best for text classification in Chinese, English, Japanese and Korean? arXiv preprint arXiv:1708.02657
7. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. Adv Neural Inf Process Syst 2015:649–657
8. Hershey S et al (2017) CNN architectures for large-scale audio classification. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Piscataway, pp 131–135
9. Romaszko L (2015) Signal correlation prediction using convolutional neural networks. In: Neural connectomics workshop, pp 45–56
10. Chen Z, He K, Li J, Geng Y (2017) Seq2Img: a sequence-to-image based approach towards IP traffic classification using convolutional neural networks. In: 2017 IEEE international conference on big data (big data). IEEE, Piscataway, pp 1271–1276
11. Zhou H, Wang Y, Lei X, Liu Y (2017) A method of improved CNN traffic classification. In: 2017 13th international conference on Computational Intelligence and Security (CIS). IEEE, Piscataway, pp 177–181
12. Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN). IEEE, Piscataway, pp 712–717
13. Millar K, Cheng A, Chew HG, Lim C-C (2018) Deep learning for classifying malicious network traffic. Presented at the Pacific-Asia conference on knowledge discovery and data mining, Melbourne, Australia
14. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
15. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
16. LeCun Y, Bengio Y (1995) Convolutional networks for images, speech, and time series. Handb Brain Theory Neural Netw 3361(10)
17. Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning
18. Marpaung JAP, Sain M, Hoon-Jae L (2012) Survey on malware evasion techniques: state of the art and challenges. In: 2012 14th International Conference on Advanced Communication Technology (ICACT). IEEE, Piscataway, pp 744–749

19. Del Carlo C (2003) Intrusion detection evasion: how attackers get past the burglar alarm. SANS Great Lakes, Chicago
20. Wang K, Stolfo SJ (2004) Anomalous payload-based network intrusion detection. In: International workshop on recent advances in intrusion detection. Springer, pp 203–222
21. Wang Z (2015) The applications of deep learning on traffic identification. Black Hat USA
22. Aceto G, Dainotti A, Donato WD, Pescape A (2010) PortLoad: taking the best of two worlds in traffic classification. In: 2010 INFOCOM IEEE conference on computer communications workshops. IEEE, Piscataway, pp 1–5
23. Smit D, Millar K, Page C, Cheng A, Chew HG, Lim C-C (2017) Looking deeper – using deep learning to identify internet communications traffic. Presented at the Australasian Conference of Undergraduate Research (ACUR), Adelaide
24. Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R (1994) Signature verification using a "Siamese" time delay neural network. Adv Neural Inf Process Syst *6*:737–744
25. Nour M, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Military Communications and Information Systems Conference (MilCIS). EEE, Piscataway
26. Nour M, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 Sata set and the comparison with the KDD99 data set. In: Inf Secur J: Glob Perspect, pp 1–14
27. Martín A et al (2016) TensorFlow: a system for large-scale machine learning. OSDI 16: 265–283

DBD: Deep Learning DGA-Based Botnet Detection

R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran, Mamoun Alazab, and Alireza Jolfaei

Abstract Botnets play an important role in malware distribution and they are widely used for spreading malicious activities in the Internet. The study of the literature shows that a large subset of botnets use DNS poisoning to spread out malicious activities and that there are various methods for their detection using DNS queries. However, since botnets generate domain names quite frequently, the resolution of domain names can be very time consuming. Hence, the detection of botnets can be extremely difficult. This chapter propose a novel deep learning framework to detect malicious domains generated by malicious Domain Generation Algorithms (DGA). The proposed DGA detection method, named, Deep Bot Detect (DBD) is able to evaluate data from large scale networks without reverse engineering or performing Non-Existent Domain (NXDomain) inspection. The framework analyzes domain names and categorizes them using statistical features, which are extracted implicitly through deep learning architectures. The framework is tested and deployed in our lab environment. The experimental results demonstrate the effectiveness of the proposed framework and shows that the proposed method has high accuracy and low false-positive rates. The proposed framework is a simple architecture that contains fewer learnable parameters compared to other character-based, short text classification models. Therefore, the proposed framework is faster

R. Vinayakumar (✉) · K. P. Soman
Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: r_vinayakumar@cb.amrita.edu

P. Poornachandran
Centre for Cyber Security Systems and Networks, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India

M. Alazab
Charles Darwin University, Casuarina, NT, Australia
e-mail: mamoun.alazab@cdu.edu.au

A. Jolfaei
Department of Computing, Macquarie University, Sydney, NSW, Australia
e-mail: alireza.jolfaei@mq.edu.au

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*, Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_6

to train and is less prone to over-fitting. The framework provides an early detection mechanism for the identification of Domain-Flux botnets propagating in a network and it helps keep the Internet clean from related malicious activities.

Keywords Botnet · Deep learning · Domain name generation · Malware · Cybercrime · Cyber security · Domain-flux · Keras embedding

1 Introduction

Over the past decade, the number of malicious activities have significantly increased [1–6, 13]. Today, perpetrators use malicious codes to infect a large collection of computers, named, botnet, and use a Command and Control (C&C) server, named, botmaster, to remotely control botnets for running malicious activities, such as Distributed Denial of Service (DDoS) attacks, spam emails, scareware, and spyware. A possible mitigation strategy is to blacklist the botmaster, to whom the botnets try to contact. However, to evade being blacklisted, perpetrators change their domain names frequently through Domain Name System (DNS) poisoning [7]. A commonly used method for C&C location resolution and fail-over resilience is fluxing, which mainly has two types: IP flux and Domain-Flux [8]. In this work, we mainly focus on Domain-Flux services, through which the botmaster changes the domain name that has to be mapped into the IP address of the C&C server frequently. This is done using a Domain Generation Algorithm (DGA), which generates domain names randomly on a large scale for registration. Botnets use various DGAs for domain name generation. For example, Conficker, Torpig, Kraken, and Murofet [9].

A method to combat DGA generated malware is to monitor DNS requests and detect DGA domain names using a machine learning classification algorithm. The past works on DGA detection can be grouped into two categories: retrospective and real-time detection. Retrospective detection cannot be used in real-time, because they model a large volume of predictions over a large volume of domains [10]. These methods cluster domain names using various statistical tests, such as Kullback-Leibler divergence testing [12], and contextual information, such as HTTP headers to further improve the performance. These methods are computationally expensive. On the contrary, real-time detection methods make the classifications using domain names without any additional contextual information. Although such approaches are fast, they cannot perform as accurate as the retrospective methods.

The existing real-time based methods for DGA detection are based on feature engineering and the commonly used features are entropy, string length, vowel to consonant ratio, and n-gram. Features are fed into the machine learning classifier for classification (the commonly used classifier is the random forest). In [10], the authors proposed a retrospective approach for DGA detection based on automatic feature extraction. Recently, the application of deep learning architectures, such as a Long Short Term Memory (LSTM) method, has been considered for DGA analysis [11]. Such approaches outperform the classical machine learning classification in terms of accuracy.

The deep learning architectures including LSTM performed well in various cyber security applications compared to classical machine learning algorithms [44], [45].

In this chapter, we propose a scalable deep learning DGA-based botnet identification framework, named, DBD, which uses a Convolutional Neural Network with a Long Short Term Memory (CNN-LSTM) pipeline to detect DGA domain names. Along with DBD, different character-based, short-text classification models are evaluated over a large labeled data set, which consists of a collection of algorithmically generated and legitimate domains gathered from our lab based network activities. Experimental results confirm the effectiveness of our framework in identifying DGA-based botnets, as well as the compromised systems within the deployed network. DBD utilizes TensorFlow [14], that is, a distributed framework for learning very large scale deep neural networks. Once the framework classifies a particular domain as a DGA domain, the framework assumes that the host is infected and analyses its resolved domain list for finding the corresponding C&C server.

The remainder of the chapter is organized as follows: Sect. 2 describes the background information on DNS, Keras embedding, and deep learning architecture. Section 3 describes the past literature on DGA detection. Section 4 proposes a scalable deep learning DGA-based botnet identification framework. Section 5 elaborates on the experiments and observations. Finally, Sect. 6 concludes the chapter.

2 Background and Preliminaries

2.1 Botnet

A botnet is a network of compromised Internet connected computers that are ruled by a bot master remotely through the command and control (C&C) channel. An infected computer in a network is named bot. A bot master uses the bot to host malicious activities. Mostly, the size or structure of botnet can vary. However, they follow the same stages in their lifecycle [15]. To adapt to the new technologies, the botnet evolves with the bot master. The C&C channel provides a communication point between the bot master and the bot to transmit data among them. The C&C communications are as follows:

- A bot master contacts botnets by issuing a command;
- Based on the command, a botnet performs its activities; and
- Botnet forwards the results of performed activities to the bot master.

Botnet can be detected only when the location of the C&C server is identified. In addition, the bot master will not be able to control the botnet without establishing a reliable connection between C&C servers and compromised computers. Primarily, there are three types of C&C architectures based on the way the communication is used, that is, centralized, decentralized, and hybrid. In a centralized botnet architecture, the bot master manages all the active bots in a botnet from a centralized C&C server. In other words, the bot communicates with a C&C server for all

operations, such as receiving and answering commands. The centralized botnet architecture uses a hierarchical, star topology. The Internet Relay Chat (IRC) and Hyper Text Transfer Protocol (HTTP) are key protocols in a centralized architecture. This type of architecture is easier to control, since it has only one central point. However, the design of a centralized architecture is complex. Also, message latency and survivability are short and chances of failure is more compared to other architectures. Decentralized architectures or peer-to-peer architectures contain more than one C&C servers to manage all active bots in a botnet. Each bot acts as a C&C server as well as clients. The detection of the botnet which uses a decentralized architecture is a difficult task, as compared to the centralized architecture. This is due to the use of peer-to-peer protocols. The design of decentralized architecture is complex, while its message latency and survivability is higher and the chances of failure is less compared to other centralized architectures. Hybrid architecture is a combination of centralized and decentralized architecture. The detection and monitoring tasks in a hybrid architecture is more difficult than centralized and decentralized architectures. However, the design of the hybrid architecture is simpler compared to other architectures.

2.2 Domain Name System

Domain Name System (DNS) is an application protocol in the Internet infrastructure, which has a distributed data base that cross-references domain names to their corresponding Internet Protocol (IP) addresses and vice versa. DNS maintains a hierarchy to control its distributed data base, and it is comprised of a root level, top level domains, second level domains, sub-domains, and hosts. Domain names consist of a Top Level Domain (TLD) and a Second Level Domain (SLD), which are separated by a dot point. For example, in YouTube.com, com is a TLD and YouTube is a SLD. The DNS hierarchy is divided into zones which are controlled by a zone manager. Each node in the DNS hierarchy has a label, which includes information related with the domain name.

A hierarchy of DNS is named domain namespace, which is similar to an eDirectory with an inverted tree structure. Organizations can use their own domain namespace to create private networks. These private networks are not visible on the Internet. A simple DNS hierarchy is shown in Fig. 1. A domain name composed of one or more sections or sub domains is named a label. A path from a sub-domain to a root is named a fully qualified domain name. A DNS hierarchy can have a maximum number of 127 levels. These labels are separated by dots. Each label can contain 0 to 63 characters. A label of length 0 is assigned to the root zone. The full domain name length can contain 253 characters. A domain name redirects the end-user request to a particular source where the information exists in the DNS name space. All the information of domain names is stored in the DNS server in the form of a record, named, resource records.

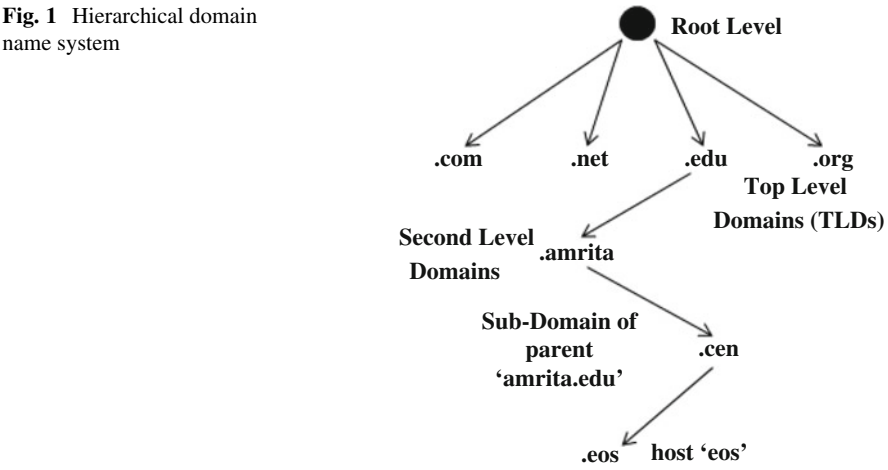


Fig. 1 Hierarchical domain name system

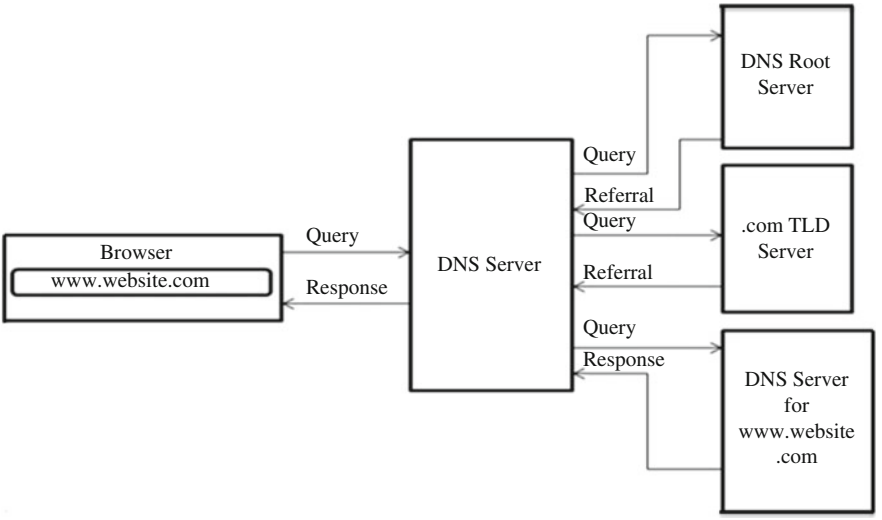


Fig. 2 Recursive DNS query

Recursive and non-recursive (or iterative) are two types of DNS servers. Non-Recursive DNS servers act as the Start of Authority (SoA); they answer queries inside the governed domains without querying other DNS servers even if the Non-Recursive DNS server cannot provide the requested answer. Recursive DNS servers, an example of which is shown in Fig. 2, respond to queries of all types of domains, by querying other servers and passing the response back to the client. Recursive DNS servers usually suffer from DDoS attacks, DNS cache poisoning, unauthorized use of resources, and root name server performance degradation. Furthermore, query to DNS servers are usually not encrypted [16]; thus, it is possible to detect the websites that are being browsed.

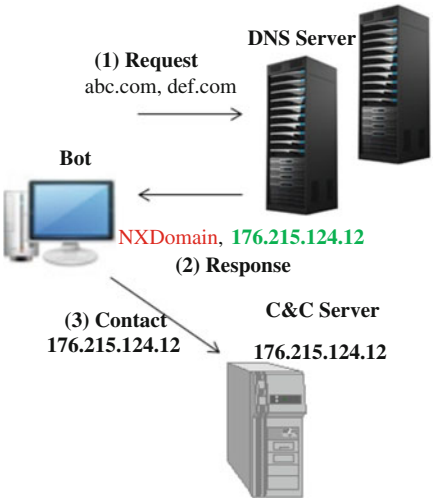


Fig. 3 Diagram of domain-flux attacks

2.3 Domain-Flux and Domain Generation Algorithms

Domain fluxing is a technique in which fully qualified domain names linked with the IP address of C&C servers are frequently changed to keep the botnets in operation. To carry out this operation, the bot master makes use of a Domain Generation Algorithm (DGA) to generate domain names on a large scale, which can bypass blacklisting and heuristics methods for detecting DGA domain names. Figure 3 shows the flow diagram of domain-flux attacks. The bot uses two domains, abc.com and def.com. The domain abc.com is not registered and receives an NXDomain response from the DNS server. def.com is registered and hence it is able to contact the C&C server. In Domain-Flux attacks, botnets contact their botmaster with randomly generated domain names using a hit and trial method. In most cases, this would generate several Non-Existent (NX) response queries for domains which have no IP addresses (NXDomains). Monitoring DNS traffic is an important task and it helps to detect the botnet using a DNS analysis. This method is more efficient than the static and binary analysis of malware. This is mainly because these methods require much less time to reverse engineer the binaries to assign a signature [37].

Analysis of DNS queries helps to detect DGA generated domains and it helps to trace the botnet and block the communication point between the botmaster and C&C server in a timely manner. Detection of DGA generated domain names through analysis of DNS data has various advantages. For example, the DNS protocol includes only a small amount of traffic in the entire network, which makes it appropriate for analysis even in large scale networks. In addition, the DNS traffic is normally cached which reduces the traffic. Some of the domain features, such as the Autonomous System (AS) number and the domain owner, can be added to DNS traffic features to further increase the detection rate of DGA domain names. Furthermore, the analysis of DNS queries helps to identify attacks in early stages or even before they occur.

A DGA generates a large number of domain names using a random seed, which can be a date, a number, or any random characters. To construct domain names, the DGA generator uses a combination of bitshift, xor, divide, multiply, and modulo operations to generate a sequence of characters, which follow a certain distribution, such as a normal or a uniform distribution model. DGA generators normally use different seeds quit frequently. For instance, the seeds may change within a period of one day. This makes the blacklisting strategies inefficient since DGAs keep creating differing domain names. However, DGA generated domain names contain unique statistical properties, which are different from legitimate domain names. As explained in Sect. 4, our proposed detection method would utilize these properties to detect DGA domain names.

2.4 *Keras Embedding and Convolutional Neural Network with Long Short-Term Memory*

Keras embedding converts positive integers (indexes) into continuous or dense vector representations of fixed sizes. These indexes direct to a particular character in a vocabulary. Dense vector representation is a character embedding matrix which is calculated using

$$(nbchar, vocabsize) \times (vocabsize, embeddim) = (nbchar, embeddim), \quad (1)$$

where $nbchar$ denotes the number of unique characters, $vocabsize$ denotes the size of vocabulary, and $embeddim$ denotes the size of the vector space in which characters will be embedded. The resultant matrix of the embedding layer are of the shape $(vocabsize,\ embeddim)$ which are passed as input to the deep learning architectures. The weights in embedding layer are initialized using a Gaussian distribution. The Keras embedding layer collaboratively works with the other deep learning layers to optimize the weights during the backpropagation.

Given a matrix representation M of Keras embedding, the CNN captures the features from the character representation. CNN is composed of a 1D convolution, a pooling, and a fully connected layer. The 1D convolution contains a filter that slides over a sequence and detects features from different locations. To reduce the feature dimension, maxpooling 1D is applied. These new features are fed into LSTM. This procedure is mathematically represented as

$$H^{CNN} = CNN_{char}(M), \quad (2)$$

$$h_t = LSTM(e_t, h_{t-1}). \quad (3)$$

where $e = H^{CNN}$, each row is a time step for LSTM, h_t is a hidden layer representation in time step t. LSTM works on each row of H^{CNN} with hidden

vectors from previous time-step to produce embedding for the subsequent time-steps. Finally, the successive feature representation of LSTM is passed into the fully connected layer, which contains a non-linear activation function that gives 0 or 1. The output 0 indicates a legitimate domain and the output 1 indicates a DGA generated domain name.

3 Related Works

Pleiades designed the first method for detecting DGA-based domains, which does not require the reverse engineering of the bot malware [10]. In [18], Woodbridge et al. applied a Long Short-Term Memory (LSTM) method for detection and classification of DGAs. Woodbridge et al.'s LSTM network is composed of an embedding layer, an LSTM layer and a logistic regression. Woodbridge et al. compared their detection method with a random forest classifier and a feature-less hidden markov model [10]. In all experiments, the LSTM method outperformed in comparison to the hand-crafted features and featureless methods. In [19], the authors discussed various issues, including the accuracy and the false alarm rate of the existing DGA detection methods, and proposed a new way to label the DNS traffic data. They used this labeled data to evaluate the performance of featureless methods CNN and LSTM with Keras embedding and hand crafted features with conventional machine learning classifiers. In [20], the authors studied the efficiency of RNN on a large number of malware samples with a total of 61 malware families.

In [21], the authors discussed the performance of deep learning architectures for DGA detection and categorization on a combination of data set, which was collected from publically available data sets and real time OSINT DGA feeds. They showed detailed experimental analysis of each of the deep learning architectures. Their experiments showed that deep learning architectures perform better compared to random forest classifier. In [22], the authors discussed methods of collecting DNS logs inside Ethernet LAN. They evaluated the performance of recurrent structures using two different data sets. One data was collected from public sources and the other was collected from the real-time DNS traffic. This was done to effectively find out a robust machine learning model, which can be deployed in real time to monitor the DNS traffic. In [23], the authors proposed a scalable framework which provides a method to collect DNS traffic data at the ISP level.

In [24], the authors proposed a unique deep learning architecture, which can detect both DGA and malicious Uniform Resource Locator (URL). The unique architecture proposed in [24] has been evaluated using DNS, URL and Email data analysis for various types of testing data sets [25]. The results are compared with other deep learning architectures and logistic regression with bigram text representation. Curtin et al. [27] proposed an RNN based approach for DGA domain detection. The approach uses a smashword score which measures the closeness in respect to English words. Tran et al. [28] discussed the importance of cost-sensitive approach along with LSTM for DGA botnet detection.

In [29], the authors applied various ImageNet models for DGA analysis. They described how to transform the pretrained weights of the ImageNet database and make the database usable for DGA data. In [30], Mac et al. studied various conventional machine learning classifiers, deep learning architectures, and the combination of deep learning architectures and conventional machine learning for DGA analysis. In a hybrid network, deep learning architecture obtained optimal features and the conventional machine learning was used for classification. In [31], Yu et al. discussed the efficiency of various character-based models for DGA analysis. The models are Endgame [18], Invincea [32], CMU [33], MIT [35], NYU [36].

To model short character strings (URLs, file paths, or registry keys), Saxe and Berlin proposed a three-layer CNN network with Keras embedding [32]. Saxe and Berlin method contains 1,024 neurons and it uses a $ReLU$ activation function. To speed up the model training and prevent from over fitting, Saxe et al. used batch normalization and regularization techniques. In [33], Dhingra et al. proposed a representation and classification method for Tweeter posts. Dhingra et al. used a bidirectional Gated Recurrent Unit (GRU) to learn feature representation of Twitter data. The tweets are tokenized into a stream of characters and each character is represented using one hot character encoding. These one-hot representations are mapped into a character space and are passed into the bidirectional GRU. It contains both forward and backward GRUs, which facilitate the learning process of the sequence of characters in the domain name. A $softmax$ non-linear activation function is used for tweet classification. In [36], Zhang et al. proposed the NYU model for tweet classification. They combined a stack of CNN layers with a LSTM layer with word-based text representation. Since stacked CNN layers would impose over-fitting, Zhang et al. suggested the use of a minimum number of CNN layers, that is, one. They evaluated the efficacy of deep learning models using the conventional text representation methods, such as bag-of-words, ngrams with Term Frequency-Inverse Document Frequency (TF-IDF). In [35] Vosoughi et al. proposed a novel tweet representation based on a Convolutional Neural Network (CNN), that is, commonly used in the field of image processing [34], LSTM and hybrid of CNN and LSTM.

Details of various character-based, short-text classification algorithms are given in Table 1. All algorithms use a character-level embedding as their first layer to transform the domain names into numeric vectors, and this is followed by a deep learning architecture for optimal feature extraction and classification. Details of deep learning based DGA detection algorithms are given in Table 2.

Table 1 Well-known short text classification methods

Name	Architecture	Task
CMU [33]	Bi-directional gated recurrent unit layer	Social media text classification
MIT [35]	Hybrid of stacked CNN layers and LSTM layer	Social media text classification
NYU [36]	Stacked CNN layers	Text classification

Table 2 A short review on deep learning based DGA domain detection

Reference	Method	Data source	Text representation
Woodbridge et al. [18]	LSTM	*legitimate:* One million domains names from Alexa *DGA domains:* OSINT DGA feed from Bambenek Consulting	Keras embedding
Yu et al. [19]	CNN and LSTM	*legitimate:* One million domains names from Alexa and Farsight Security DNS data stream *DGA domains:* DGArchive and Farsight Security DNS stream	Keras embedding
Lison et al. [20]	Architectures related to recurrent structure	*legitimate:* One million domains names from Alexa, Statvoo, and Cisco *DGA domains:* DGArchive and Bambenek	Keras embedding and one-hot encoding
Vinayakumar et al. [21]	CNN and recurrent structures	*legitimate:* One million domains names from Alexa and OpenDNS *DGA domains:* OSINT DGA feeds	Keras embedding
Vinayakumar et al. [22]	RNN and LSTM	*legitimate:* One million domains names from Alexa, OpenDNS, and Ethernet LAN DNS stream *DGA domains:* OSINT DGA feeds and Ethernet LAN DNS stream	Keras embedding
Vinayakumar et al. [23]	CNN and recurrent structures	*legitimate:* One million domains names from Alexa and real time DNS stream from the Internet Service Provider (ISP) level *DGA domains:* real time DNS stream from the ISP level	Keras embedding
Yu et al. [31]	CNN, RNN, LSTM, CNN and RNN, CNN and LSTM, Bidirectional RNN, Bidirectional GRU	*legitimate:* One million domains names from Alexa *DGA domains:* Bambenek	Keras embedding and One-hot encoding
Mohan et al. [24]	CNN and CNN-LSTM	*legitimate:* One million domains names from Alexa and OpenDNS *DGA domains:* OSINT DGA feeds	Keras embedding
Vinayakumar et al. [25]	CNN-LSTM, CNN and CNN-RNN	*legitimate:* One million domains names from Alexa and OpenDNS *DGA domains:* publically available DGA algorithms, 360-DGA, OSINT DGA feeds, DGArchieve and Ethernet LAN DNS stream	Keras embedding

(continued)

Table 2 (continued)

Reference	Method	Data source	Text representation
Mac et al. [30]	LSTM, CNN and LSTM, Bidirectional LSTM	*legitimate:* One million domains names from Alexa *DGA domains:* Bambenek	Keras embedding and One-hot encoding
Curtin et al. [27]	RNN, LSTM, and generalized likelihood ratio test	*legitimate:* One million domains names from Alexa and OpenDNS *DGA domains:* simulated domain names from publically available DGA algorithms, DGArchive, Andrey Abakumov's DGA repository on Github, Johannes Bader's DGA implementations, and Sinkholed domains collected from public WHOIS registration	One-hot encoding and TF-IDF
Tran et al. [28]	Cost-sensitive LSTM	*legitimate:* One million domains names from Alexa *DGA domains:* OSINT DGA feed from Bambenek Consulting	Keras embedding

4 Proposed DGA Detection: Deep Bot Detect

Since DNS traffic is not encrypted, an analysis of DNS traffic would reveal botnets that utilize DGAs. Generally, the presence of DGAs can be detected through

- Analysis of Non-Existent (NX) domain DNS response;
- Analysis of statistical features of DNS queries;
- Analysis of statistical features of DNS responses; and
- Time based analysis of DNS traffic.

The proposed DGA detection method, named, Deep Bot Detect (DBD), is based on an analysis of statistical features of DNS queries. This includes a preprocessing phase, an optimal features extraction phase, and a classification phase. In the preprocessing phase, the feature vectors are extracted from domain names using text representation methods, and then, the optimal features are computed from the numeric vectors using deep learning architectures. Finally, a classification is done using a fully connected layer with a non-linear activation function.

The proposed method converts domain names into lower-case, character strings, because domain names are case-insensitive and differentiating between capital and small letters may cause regularization issues; otherwise, the method has to be trained for more numbers of epochs to learn about differing patterns of the domain names.

The detailed configuration of DBD is listed in Table 3, which looks similar to that of [24] and [23]. The proposed method has less number of trainable parameters in comparison with other character based short text classification models. Thus, the DBD framework is faster to train and less prone to overfitting.

Table 3 Detailed configuration of DBD

Layer (type)	Output shape	Param #
Embedding	(None, 91, 128)	6,528
Conv1D	(None, 8,764)	41,024
MaxPooling1	(None, 2,164)	0
LSTM	(None, 70)	37,800
Dense	(None, 1)	71
Activation	(None, 1)	0
Total params: 85,423		
Trainable params: 85,423		
Non-trainable params: 0		

The corpus contains 39 unique characters: -, ., _, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The maximum length of the domain name is 91. If the length of the domain name is less than 91, it is padded with zeros. In DBD, the vector size of Keras embedding is 128, and each character is mapped into a 1×128 vector. This is different from ASCII encoding, and it helps to learn the similarity among characters by mapping the semantics of similar characters to similar vectors. The keras embedding outputs a matrix representation 39×128, where the number of unique characters is 39 and the dimension of embedding space 128. The Keras embedding jointly works with other layer in the deep learning model and obtains an optimal feature representation that composed of similar characters being embedded closer to each other.

In recent days, visualization plays an important role in deep learning based cyber security applications particularly in intermediate feature visualization [46]

To visualize, the 128-dimensional representation is fed into a t-SNE [26] that converts it to a two-dimensional representation, as shown in Fig. 4. The numbers, characters and special characters are appeared in their own cluster. This indicates that the model has captured the unique features to differentiate characters, numbers and special characters.

To conduct a fair comparative evaluation strategy, for all models under study, we considered Keras embedding as the domain name representation method and we set the embedding size to 128. To evaluate the effectiveness of Keras embedding, we used a 3-gram text representation method to mapped domain names. We hashed the features of 3 grams into a vector of length $1 \times 1{,}000$. These vectors are passed into a DNN for optimal feature extraction and classification. The detailed configuration is listed in Table 4. To avoid overfitting and to speed up the training process, dropout and batch normalization is used, respectively, in between the DNN layers. We found both dropout and batch normalization is use full. Following the embedding layer, DBD contains a convolution layer, which contains 64 filters of length 5 with an $ReLU$ activation function. The Convolutional layer includes a maxpooling with a pool length of 4 and an LSTM with 70 memory blocks. This type of hybrid CNN-LSTM model learns the important features and forms a fixed length feature vector representation. Finally, the 128 vector representation is passed to the fully

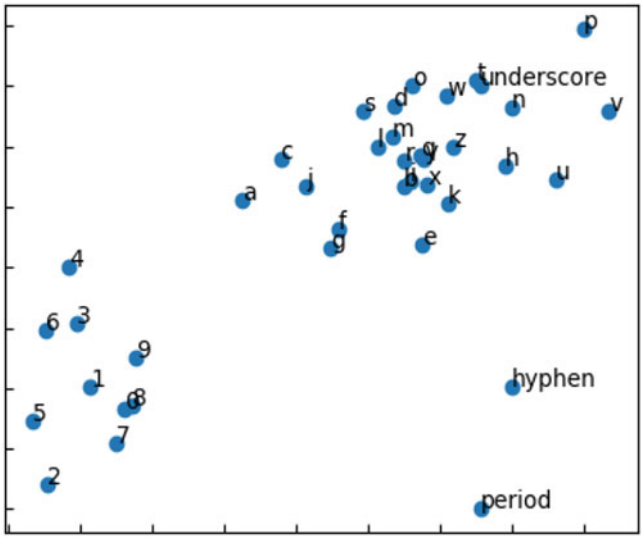


Fig. 4 Keras embedding character vector visualization of DBD model

Table 4 Detailed configuration of DNN

Layer (type)	Output shape	Param #
dense_1 (Dense)	(None, 128)	128,128
batch_normalization_1 (Batch)	(None, 128)	512
activation_1 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 96)	12,384
batch_normalization_2 (Batch)	(None, 96)	384
activation_2 (Activation)	(None, 96)	0
dropout_2 (Dropout)	(None, 96)	0
dense_3 (Dense)	(None, 64)	6,208
batch_normalization_3 (Batch)	(None, 64)	256
activation_3 (Activation)	(None, 64)	0
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 32)	2,080
batch_normalization_4 (Batch)	(None, 32)	128
activation_4 (Activation)	(None, 32)	0
dropout_4 (Dropout)	(None, 32)	0
dense_5 (Dense)	(None, 16)	528
batch_normalization_5 (Batch)	(None, 16)	64
activation_5 (Activation)	(None, 16)	0
dropout_5 (Dropout)	(None, 16)	0
dense_6 (Dense)	(None, 1)	17
Total params: 150,689		
Trainable params: 150,017		
Non-trainable params: 672		

connected layer that contains a $sigmoid$ non-linear activation function, which outputs 0 for legitimate and 1 for DGA domain names. To minimize the loss during backpropogation, all deep learning architectures use a binary cross entropy loss function, defined as

$$loss(pd, ed) = -\frac{1}{N}\sum_{i=1}^{N}[ed_i \log pd_i + (1 - ed_i)\log(1 - pd_i)], \tag{4}$$

where pd is a vector of predicted class label in testing data set, ed is a vector of expected class label, and N is the number of samples.

5 Experiments, Results and Observations

Finding a large, ground truth dataset of domain names for DGA analysis is often difficult. Although many published machine learning based DGA detection solutions exist, there are still many ambiguities in regards to the selection of appropriate machine learning algorithms. This is mainly because most benchmark results are from various data sets, which have differing data distributions and are collected from different resources, such as Alexa [38], OpenDNS [39], 360-DGA, OSINT real time DGA feeds [40], DGArchive [41] and publically available DGA algorithms [42]. In addition, some of the data sets and their associated systems are not publically available for research, due to privacy and security concerns. More importantly, the past DGA analytics have not considered time domain information, which is an important factor to identify zero day vulnerabilities. The majority of security companies make their data sets private, since they do not want to disclose their security flaws.

We evaluated the performance of the proposed method using two corpuses: publically available and private corpus. The publically available corpus (Data set 1) is collected from the publically available DGA algorithms including [42], DGArchive [41], and OSINT real time DGA feeds [40]. The main difference between the two test data sets of Data set 1 is that the Test 1 data set is collected from both the publically available DGA algorithms and real-time OSINT DGA feeds and the Test 2 data set is collected from only real-time OSINT DGA feeds. The private corpus (Data set 2) is collected from the internal network [23]. This data set is entirely collected in a different environment in comparison to the Data set 1. The private data set contains less number of DGA domain names, and it is highly imbalanced. The deep learning architectures are prone to imbalanced data analysis. This requires additional concepts from the field of data mining to handle the imbalanced data more efficiently during learning process [47]. Applying the data mining concepts to handle the imbalanced data can be considered as one of the significant directions towards future works. These different test data sets show how well the trained model on the previous day data is able to detect DGA domain

Table 5 Detailed statistics of the data sets

Data set	Legitimate	DGA generated	Total	Observation dates
Data set 1 (Train)	155,683	115,056	270,739	May 1, 2017–Oct. 15, 2018
Data set 1 (Test 1)	181,837	37,742	219,579	Jan. 1, 2018–Mar. 25, 2018
Data set 1 (Test 2)	867,027	39,946	906,973	Apr. 3, 2018–May 22, 2018
Data set 2 (Test 1)	547,165	27,983	575,148	Feb. 1, 2017–Nov. 15, 2018

names, which occur in the future and how well the train model is able to detect DGA domain names which occurs from different sources. The detailed statistics of data set is reported in Table 5. Legitimate domain names are collected from Alexa [38] and OpenDNS [39]. Each of the test data sets are disjoint. The results of the Data set 1 can be reproduced because the data set is publically available and moreover the nature of real time domain names in Data set 2 makes the models to stays safe from an adversary. To defeat the classifier trained on Data set 2, an adversary has to spend more time in reverse engineering in comparison to the Data set 1.

To avoid possible biases, the real time DNS traffic was captured from an internal network. Passive sensors were deployed in an internal network. The detailed experimental setup and the collection of DNS traffic data set from the internal network is reported in [23]. The DNS query response messages from the DNS servers are collected by passive sensors. Thus, in this work, the DNS traffic was collected from different DNS servers. The sensors capture the network traffic from DNS servers, extract DNS packets and converts into human readable format. This data will be passed to the DNS log collector. The passive sensors could be installed inside the DNS server itself or mirror only the DNS traffic to a server, which is dedicated for DNS passive sensors. The sensors get the data by port mirroring the traffic from the DNS servers present in the deployed network. The logs received from the internal network are passed to the distributed log parser.

All deep learning architectures are implemented using TensorFlow [14] with Keras [43]. Initially, all the models are trained with Data set 1 (Train). During training, the train data set of Data set 1 is randomly divided into 80% training and 20% validation. The validation data helps to monitor the train accuracy across different epochs. The experiments with all deep learning architectures are run till 100 epochs with learning rate 0.001, *adam* optimizer, batch size of 128. To evaluate the performance, the trained model on Data set 1 is tested with the different test data sets, such as Data set 1 (Test 1), Data set 1 (Test 2), and Data set 2 (Test 1). These data sets are disjointed and they are collected from different sources. Most of the models performed better on Data set 1 (Test 1) compared to the test data taken from Data set 1 (Test 2) and Data set 2 (Test 1). An evaluation is accompanied with the ground-truth test data, which has a set of domain names labeled as DGA generated D or legitimate L. The data in Data set 1 (Test 1), Data set 1 (Test 2), and Data set 2 (Test 1) are collected from various sources and times to examine the detection capability of the trained models.

Table 6 Detailed test results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Data set 1 (Test 1)				
Endgame [18]	99.0	96.3	97.9	97.1
NYU [36]	98.7	95.3	96.9	96.1
MIT [35]	98.5	94.8	96.4	95.6
CMU [33]	98.2	92.5	97.6	94.9
Invincea [32]	98.8	96.4	96.5	96.5
3-gram with 5-layer DNN	96.5	89.9	89.9	89.9
DBD	97.8	91.5	95.9	93.7
Data set 1 (Test 2)				
Endgame [18]	98.8	79.8	96.1	87.2
NYU [36]	98.7	79.2	94.6	86.2
MIT [35]	98.6	77.2	95.5	85.4
CMU [33]	97.8	67.5	96.7	79.5
Invincea [32]	98.9	82.9	94.2	88.2
3-gram with 5-layer DNN	96.6	57.0	90.7	70.0
DBD	97.6	65.9	93.9	77.5
Data set 2 (Test 1)				
Endgame [18]	98.6	81.1	93.3	86.8
NYU [36]	98.5	80.5	92.3	86.0
MIT [35]	98.4	78.3	93.1	85.1
CMU [33]	97.7	69.2	94.3	79.8
Invincea [32]	98.8	84.3	91.8	87.9
3-gram with 5-layer DNN	96.4	58.8	88.6	70.7
DBD	97.5	67.8	92.1	78.1

We used $Accuracy \in [0, 1]$, $Precision \in [0, 1]$, $Recall \in [0, 1]$, and $F1 - score \in [0, 1]$ standard metrics, which are based on True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Detailed results are reported in Table 6. TP represents the number of DGA generated domain name samples correctly identified as DGA, TN represents the number of legitimate domain name samples correctly identified as legitimate, FP represents the number of legitimate domain name samples misclassified as DGA, FN denotes the number of DGA generated domain name samples misclassified as legitimate. The metrics $Accuracy$, $Precision$, $Recall$ and $F1 - score$ are defined as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

$$Precision = \frac{TP}{TP + FP}, \tag{6}$$

$$Recall = \frac{TP}{TP + FN}, \tag{7}$$

and

$$F1 - score = 2 \times \left(\frac{\text{Pr}ecision \times \text{Re}call}{\text{Pr}ecision + \text{Re}call} \right). \tag{8}$$

Two most commonly used diagnostic tools to identify the interpretation of the binary classifiers are Receiver Operating Characteristic (ROC) and the precision-recall curves. Generally, the ROC curves are used when the samples of each class are balanced; otherwise, precision-recall curves are used.

To generate precision-recall curves, we estimated the trade-off between the $Precision \in [0, 1]$ and $Recall \in [0, 1]$ across varying threshold in the range of [0, 1]. Most commonly, the area under the curve (AUC) is used to compare the ROC and precision-recall curves. AUC, as the name indicates, just the area under the ROC curve. To generate ROC, we estimated the trade-off between the true positive rate $TPR \in [0, 1]$ (Eq. 7) on the vertical axis to false positive rate ($FPR \in [0, 1]$) on the horizontal axis across varying threshold in the range of [0, 1], where

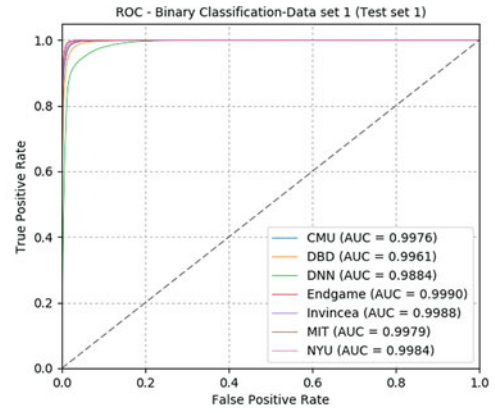
$$TPR = \frac{TP}{TP + FN}, \tag{9}$$

and

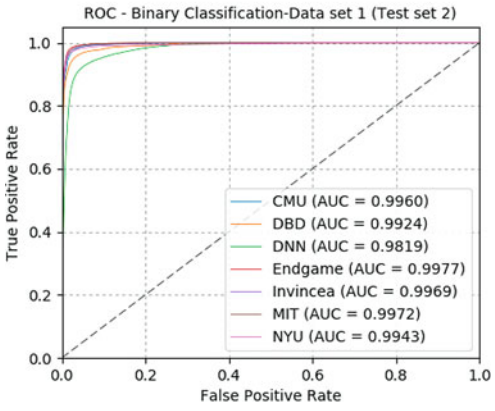
$$FPR = \frac{FP}{FP + TN}. \tag{10}$$

We set the threshold for both ROC and precision-recall curves to 0.5 without fine-tuning for low FPR. Low FPR is an important measure in DGA detection, particularly in applications where there is a live stream of queries in a DNS server. If the FPR is high, the system may block legitimate traffic in real-time systems. To highlight the differences in methods, FPR is adopted on a log scale.

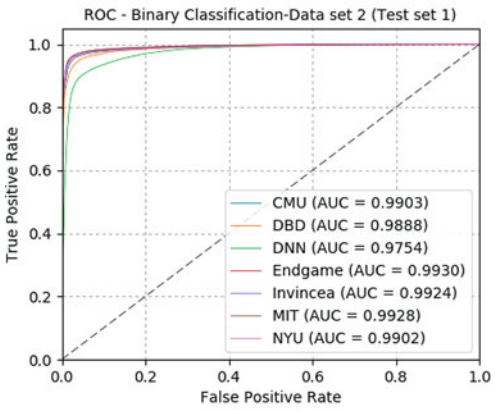
The ROC and the precision-recall curves of various models are depicted in Figs. 5 and 6. These curves are obtained using different test data. A high score of precision and recall indicates a more accurate detection. The results of various tests are averaged. Among all 3 data sets, the CMU model shows the lowest FPR, as reported in Table 7. There exists a clear variation in the TPR that the models achieve against that low FPR. All the models are able to catch DGA domain names with a TPR of 70–80 range on Data set 1 (Test 1), 90–98 range on Data set 1 (Test 2) and Data set 2 (Test 1). However, our proposed method, even though the threshold parameter has not been fine-tuned, shows a better performance in regards to AUC (more than



(a)

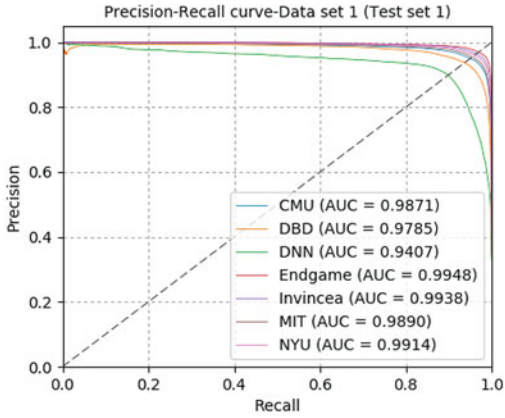


(b)

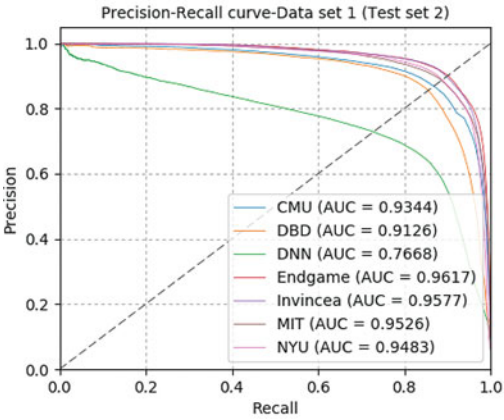


(c)

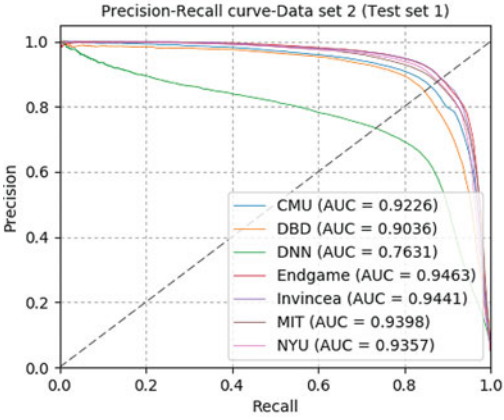
Fig. 5 ROC curve for (**a**) Data set 1 (Test 1), (**b**) Data set 1 (Test 2), and (**c**) Data set 2 (Test 1)



(a)



(b)



(c)

Fig. 6 Precision-Recall curve for (**a**) Data set 1 (Test 1), (**b**) Data set 1 (Test 2), and (**c**) Data set 2 (Test 1)

Table 7 Results on test data sets. TPR and FPR are w.r.t. a threshold 0.5

Model	TPR (%)	FPR	AUC (%)
Data set 1 (Test 1)			
Endgame [18]	72.046	0.246	99.76
NYU [36]	81.353	0.348	99.61
MIT [35]	83.404	0.339	98.84
CMU [33]	70	0.078	99.9
Invincea [32]	75.82	0.276	99.88
3-gram with 5-layer DNN	84.849	0.209	99.79
DBD	86.871	0.383	99.84
Data set 1 (Test 2)			
Endgame [18]	96.079	0.542	99.6
NYU [36]	96.842	0.568	99.24
MIT [35]	96.962	0.574	98.19
CMU [33]	93.597	0.346	99.77
Invincea [32]	96.374	0.543	99.69
3-gram with 5-layer DNN	93.418	0.487	99.72
DBD	96.755	0.573	99.43
Data set 2 (Test 1)			
Endgame [18]	93.947	0.511	99.03
NYU [36]	95.910	0.555	98.88
MIT [35]	96.137	0.557	97.54
CMU [33]	90.732	0.311	99.30
Invincea [32]	94.723	0.519	99.24
3-gram with 5-layer DNN	90.758	0.445	99.28
DBD	95.908	0.565	99.02

0.97) in both ROC and precision-recall curves. Similarly, the precision-recall curves shown in Fig. 6 contain larger AUCs.

Endgame [18] model has performed well in comparison to all other models in all 3 different types of testing data sets. The performance obtained by all models on Data set 1 (Test 1) is higher compared to other data sets. Moreover, the performances of all models on Data set 1 (Test 2) is higher compared to Data set 2 (Test 1). This is mainly due to the differing statistics of datasets. Both Data set 1 (Test 1) and Data set 1 (Test 2) are collected from publically available sources and Data set 2 (Test 1) is collected from a private environment. Thus, the patterns in Data set 1 (Test 1) are entirely different from those of other data sets.

The performances obtained by all models have marginal differences in terms of accuracy, the area under the receiver operating characteristic (ROC) curve and precision-recall (AUC). The methods under study used Keras embedding as the text representation method, and they performed better compared to the 3-gram text representation method with DNN. This is mainly because the Keras embedding has the capability to learn the sequential representation of characters in domain names.

6 Conclusion

Perpetrators use various families of domain generating algorithms to hide the location of their C&C servers and maintain the efficiency of their botnets. To this end, we proposed a method to detect such algorithms, which disguise servers from being blacklisted or shut down. Our proposed method uses a deep learning architecture with Keras embedding, and it has the capability to detect the newly generated DGA based domain names or the variants of existing DGA generated domain names. The proposed method does not rely on the use of contextual information, such as NXDomains, domain reputation, and feature engineering. The method analyses DNS data recursively and discovers the C&C server with the hosts it has infected. The proposed method provides DNS administrators a defensive capability to detach the rapidly growing botnets at a premature stage and lessens their adverse impacts. Our method is highly scalable and is capable of handling Ethernet LAN data with low false positive rates. We evaluated the performance of our method over various well-known character-based, short-text classification models. We used 3-gram with DNN network for our evaluations. We found that most of the models under study have marginal differences in terms of accuracy. In terms of training speed and computational cost, the DBD method is more efficient compared to other models. Our deep learning based solution is very efficient as it captures the most significant features. Furthermore, we showed that it would be difficult for perpetrators to craft malwares, which can avoid our detection method.

Acknowledgements This work was supported by the Department of Corporate and Information Services, Northern Territory Government of Australia, the Paramount Computer Systems, and Lakhshya Cyber Security Labs. We would like to thank NVIDIA India, for the GPU hardware support to research grant. We are also grateful to Computational Engineering and Networking (CEN) department for encouraging the research.

References

1. Broadhurst R, Grabosky P, Alazab M, Bouhours B, Chon S (2014) An analysis of the nature of groups engaged in cyber crime. Int J Cyber Criminol 8(1):1–20. Available at SSRN: https://ssrn.com/abstract=2461983
2. Alazab M, Venkatraman S, Watters P, Alazab M, Alazab A (2012) Cybercrime: the case of obfuscated malware. In: Georgiadis CK, Jahankhani H, Pimenidis E, Bashroush R, Al-Nemrat A (eds) Global security, safety and sustainability and e-Democracy. e-Democracy 2011, ICGS3 2011. Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, vol 99. Springer, Berlin/Heidelberg
3. Alazab M, Broadhurst R (2016) Spam and criminal activity (2015). Trends and issues in crime and criminal justice, No. 526, Australian Institute of Criminology. https://aic.gov.au/publications/tandi/tandi526
4. Ahmad U, Song H, Bilal A, Alazab M, Jolfaei A (2018) Secure passive keyless entry and start system using machine learning. In: Wang G, Chen J, Yang L (eds) Security, privacy, and anonymity in computation, communication, and storage. SpaCCS 2018. Lecture notes in computer science, vol 11342. Springer, Cham

5. Vinayakumar R, Alazab M, Jolfaei A, Soman KP, Poornachandran P (2018) Ransomware triage using deep learning: twitter as a case study. In: International Conference on Cyber Security and Communication Systems. Springer, Melbourne
6. Alazab M (2015) Profiling and classifying the behaviour of malicious codes. J Syst Softw 100:91–102. Elsevier
7. Antonakakis M, Perdisci R, Dagon D, Lee W, Feamster N (2010) Building a dynamic reputation system for DNS. In: USENIX Security Symposium, pp 273–290
8. Ollmann G (2009) Botnet communication topologies. Retrieved 30 Sept 2009. Available at http://www.technicalinfo.net/papers/PDF/WP_Botnet_Communications_Primer_(2009-06-04).pdf
9. Wang TS, Lin HT, Cheng WT, Chen CY (2017) DBod: clustering and detecting DGA-based botnets using DNS traffic analysis. Comput Secur 64:1–15
10. Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, Dagon D (2012) From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX Security Symposium, vol 12
11. Allamanis M, Barr ET, Devanbu P, Sutton C (2018) A survey of machine learning for big code and naturalness. ACM Comput Surv 51(4):81
12. Joyce JM (2011) Kullback-Leibler divergence. In: Lovric M (ed) International encyclopedia of statistical science. Springer, Berlin/Heidelberg, pp 720–722
13. Tang M, Alazab M, Luo Y (2017) Big data mining in cybersecurity. IEEE Transactions on Big Data. https://ieeexplore.ieee.org/document/7968482
14. Abadi M et al (2016) Tensorflow: a system for large-scale machine learning. In: OSDI, vol 16, pp 265–283
15. Silva SS, Silva RM, Pinto RC, Salles RM (2013) Botnets: a survey. Comput Netw 57(2): 378–403
16. Jolfaei A, Vizandan A, Mirghadri A (2012) Image encryption using HC-128 and HC-256 stream ciphers. Int J Electron Secur Digit Forensics 4(1):19–42
17. Ahluwalia A, Traore I, Ganame K, Agarwal N (2017) Detecting broad length algorithmically generated domains. In: International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. Springer, Cham, pp 19–34
18. Woodbridge J, Anderson HS, Ahuja A, Grant D (2016) Predicting domain generation algorithms with long short-term memory networks. arXiv preprint arXiv:1611.00791
19. Yu B, Gray DL, Pan J, De Cock M, Nascimento AC (2017) Inline DGA detection with deep networks. In: IEEE International Conference on Data Mining Workshops, pp 683–692
20. Lison P, Mavroeidis V (2017) Automatic detection of malware-generated domains with recurrent neural models. arXiv preprint arXiv:1709.07102
21. Vinayakumar R, Soman KP, Poornachandran P, Sachin Kumar S (2018) Evaluating deep learning approaches to characterize and classify the DGAs at scale. J Intell Fuzzy Syst 34(3):1265–1276
22. Vinayakumar R, Soman KP, Poornachandran P (2018) Detecting malicious domain names using deep learning approaches at scale. J Intell Fuzzy Syst 34(3):1355–1367
23. Vinayakumar R, Poornachandran P, Soman KP (2018) Scalable framework for cyber threat situational awareness based on domain name systems data analysis. In: Big data in engineering applications, pp 113–142
24. Mohan VS, Vinayakumar R, Soman KP, Poornachandran P (2018) Spoof Net: syntactic patterns for identification of ominous online factors. In: IEEE Security and Privacy Workshops, pp 258–263
25. Vinayakumar R, Soman KP, Poornachandran P, Mohan VS, Kumar AD (2019) ScaleNet: scalable and hybrid framework for cyber threat situational awareness based on DNS, URL, and Email data analysis. J Cyber Secur Mobil 8(2):189–240
26. Maaten LVD, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
27. Curtin RR, Gardner AB, Grzonkowski S, Kleymenov A, Mosquera A (2018) Detecting DGA domains with recurrent neural networks and side information. arXiv preprint arXiv:1810.02023

28. Tran D, Mac H, Tong V, Tran HA, Nguyen LG (2018) A LSTM based framework for handling multiclass imbalance in DGA botnet detection. Neurocomputing 275:2401–2413
29. Feng Z, Shuo C, Xiaochuan W (2017) Classification for DGA-based malicious domain names with deep learning architectures. In: 2017 Second International Conference on Applied Mathematics and Information Technology, p 5
30. Mac H, Tran D, Tong V, Nguyen LG, Tran HA (2017) DGA botnet detection using supervised learning methods. In: Proceedings of the Eighth ACM International Symposium on Information and Communication Technology, pp 211–218
31. Yu B, Pan J, Hu J, Nascimento A, De Cock M (2018) Character level based detection of DGA domain names. In: IEEE World Congress on Computational Intelligence, pp 4168–4175
32. Saxe J, Berlin K (2017) eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint arXiv:1702.08568
33. Dhingra B, Zhou Z, Fitzpatrick D, Muehl M, Cohen WW (2016) Tweet2vec: character-based distributed representations for social media. arXiv preprint arXiv:1605.03481
34. Jolfaei A, Mirghadri A (2011) Substitution-permutation based image cipher using chaotic henon and baker's maps. Int Rev Comput Softw 6(1):40–54
35. Vosoughi S, Vijayaraghavan P, Roy D (2016) Tweet2vec: learning tweet embeddings using character-level CNN-LSTM encoder-decoder. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 1041–1044
36. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. In: Advances in neural information processing systems, pp 649–657
37. Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-nimeh S, Lee W, Dagon D (2012) From throw-away traffic to bots: detecting the rise of DGA-based malware. In: Proceedings of the 21st USENIX Conference on Security Symposium
38. Does Alexa have a list of its top-ranked websites? Available at Alexa: https://support.alexa.com. Accessed 09 May 2018
39. OpenDNS domain list. Available at OpenDNS: https://umbrella.cisco.com/blog. Accessed 09 May 2018
40. OSINT DGA feeds. Available at OSINT: https://osint.bambenekconsulting.com/. Accessed 09/05/2018
41. DGArchive. Available at DGArchive: https://dgarchive.caad.fkie.fraunhofer.de/. Accessed 15/06/2018
42. DGA Algorithms. Available at Github: https://github.com/baderj/domain generation algorithms. Accessed 01/05/2018
43. Gulli A, Pal S (2017) Deep learning with Keras. Packt Publishing Ltd, Birmingham
44. Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S (2019) Deep learning approach for intelligent intrusion detection system. IEEE Access
45. Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Venkatraman S (2019) Robust intelligent malware detection using deep learning. IEEE Access
46. Venkatraman S, Alazab M (2018) Use of data visualisation for zero-day malware detection. Security and Communication Networks 2018:13. Article ID: 1728303. https://doi.org/10.1155/2018/1728303
47. Ebenuwa S, Sharif MS, Alazab M, Al-Nemrat A (2019) Variance ranking attributes selection techniques for binary classification problem in imbalance data. IEEE Access

Enhanced Domain Generating Algorithm Detection Based on Deep Neural Networks

Amara Dinesh Kumar, Harish Thodupunoori, R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran, Mamoun Alazab, and Sitalakshmi Venkatraman

Abstract In recent years, modern botnets employ the technique of domain generation algorithm (DGA) to evade detection solutions that use either reverse engineering methods, or blacklisting of malicious domain names. DGA facilitates generation of large number of pseudo random domain names to connect to the command and control server. This makes DGAs very convincing for botnet operators (botmasters) to make their botnets more effective and resilient to blacklisting and efforts of shutting-down attacks. Detecting the malicious domains generated by the DGAs in real time is the most challenging task and significant research has been carried out by applying different machine learning algorithms. This research considers contemporary state-of-the-art DGA malicious detection approaches and proposes a deep learning architecture for detecting the DGA generated domain names.

A. D. Kumar (✉) · H. Thodupunoori
Department of Electronics and Communication Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India

R. Vinayakumar · K. P. Soman
Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: r_vinayakumar@cb.amrita.edu

P. Poornachandran
Centre for Cyber Security Systems and Networks, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India

M. Alazab
Charles Darwin University, Casuarina, NT, Australia
e-mail: mamoun.alazab@cdu.edu.au

S. Venkatraman
Melbourne Polytechnic, Prahran, Australia
e-mail: sitavenkat@melbournepolytechnic.edu.au

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_7

This chapter presents extensive experiments conducted with various Deep Neural Networks (DNN), mainly, convolutional neural network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional Long Short-Term Memory (BiLSTM), Bidirectional Recurrent Neural Network (BiRNN) and CNN-LSTM layers deep learning architectures for the binary class and multi-class detection. An extensive study of the performance and efficiency of the proposed DGA Malicious Detector is conducted through rigorous experimentation and testing of two different datasets. The first dataset consists of public sources and the second dataset is from private sources. We perform a comprehensive measurement study of the DGA by analyzing more than three Million domain names. Our experiments show our DGA Malicious Detector is capable of effectively identifying domains generated by DGA families with high accuracy of 99.7% and 97.1% for the two datasets respectively. A comparative study of the deep learning approaches shows good benchmarking of our DGA Malicious Detector.

Keywords Domain generation algorithm (DGA) · Cybersecurity · Malware · Botnet · DNS · Deep learning

1 Introduction

Botnets are one among the most destructive hazards on the Internet [1–4]. A botnet is a collection of compromised computers that have been attacked by malware. They respond to instructions from a remote computer controlled by a botmaster through command and control (C&C) servers, to send bulk spam, make DDoS attacks, install other malicious code (eg fake anti-virus software) and steal sensitive information, like harvested passwords and credit card and bank accounts, to be used or sold [3]. Recent botnets are using fluxing techniques to avoid detection, which is posing challenging research opportunities in this domain [5]. Traditionally, a remote access malware depends on a fixed domain or IP address, which can be quickly blocked by security filters. To overcome this, recent hackers adopt Domain Generation Algorithms (DGA) to periodically generate a large number of pseudo domain names that can be used to switch to a new domain at regular intervals, thereby serving as rendezvous points with their C&C servers. This dynamic changing of the domain for the C&C server is also called Domain Fluxing or Fast Fluxing.

In the past, a DGA malware family called Kraken appeared in 2008 and later that year the Conficker pushed the DGA tactic into notoriety in the recent years. DGA algorithms have been employed in various malware families for periodic generation of domain names for communicating with C&C servers. In Crytolocker malware uses DGAs to generate domain names, which is usually initiated with a random seed. The generated domains are distinguishable with the benign domain names and can be identified using the lexical features [5, 6] and through neural networks [7–10]. Later, advanced DGAs started to appear in malwares such as Suppobox

[11], which are dictionary based. They use a dictionary for generating domains by concatenating words [12, 13]. Other notable examples of DGA-based botnets (another name DGA-bots) are Bobax, Kraken, Sinowal (also known as Torpig), Srizbi, Conficker [14], and Murofet.

Our research focuses on analysing the DNS traffic which accounts for only a small part of all the network traffic. We studied the traffic of botnets as they have to send DNS queries for the domain name resolution to the IP address of the C&C servers. Each botnet creates the domain name in a different way and the botnet queries the DNS server for domain resolution. The number of domains generated by the attackers will be small and they generate a large amount of DNS traffic. Some of the complexities of this problem are given below:

(i) Filtering and blacklisting the domains require significant time to reverse engineer the botnet code;
(ii) Despite reverse engineering the botnet code, the attacker may update the DGA algorithm;
(iii) The domains have to be detected irrespective of the type of botnet or the malicious software generating them by analysing the DNS queries.

In all the above cases, both blacklisting and filtering mechanisms completely fail to detect new type of domain names generated by DGA or variants of DGA generated existing domain names [15]. In order to address these problems, machine learning methods were adopted [16, 17]. Most of the existing methods relied on feature engineering [18, 19]. This is considered to be a daunting task as it requires extensive domain knowledge. Moreover, the amount of data generated in this process is huge and hence using the featureless models were at an advantage as their performance increases with the increase in data. Recently, featureless approach for DGA analysis was proposed [5] and another research demonstrated a better performance using long short-term memory (LSTM) networks [20]. These deep learning architectures have been largely used in various cyber security applications [38, 39, 40]. Since then several researchers have introduced various deep learning architectures for DGA analysis, and the performance of various deep learning architectures for DGA malicious detection as well as categorization have been reported [8]. Proposals of DGA analysis using deep learning architectures inside an Ethernet LAN and at the internet service provider (ISP) level are found in literature [9, 21]. However, there is a scarcity of research that provides an independent evaluation and benchmarking of these deep learning architectures adopted for DGA analysis. Hence, to fill this gap, in this work the performances of various deep learning architectures are evaluated for DGA analysis. Most importantly, in this work, the datasets are collected from both public sources as well as real time environments. In order to train and test the datasets, in addition to random split method, time split method is used as it helps to achieve zero day malware detection. Overall, the main objectives of this work are:

1. To perform a comparative analysis of the performance of various deep learning architectures in detecting the DGA generated domain names;

2. To perform benchmarking using experiments on two independent datasets collected from (i) public sources and (ii) private real time derived binary and multi classes;
3. To propose a novel architecture called DGA Malicious Detector for implementing DGA domain detection.

The structure of the chapter is organized as follows. Section 2 provides state-of-the-art approaches related to the study. Section 3 details the proposed architecture. The dataset description is given in Sect. 4. Section 5 describes the experiments and results and finally Sect. 6 gives the conclusion and future work.

2 Related Work

Cybercriminals use various techniques to obscure the domain names for C&C communication. Some of the botnets implement fluxing methods to avoid perceptions by masking the domain IPs. This is attained by using fast flux/domain flux to boost the sustainability. These approaches are extensively implemented for initiating pharming, phishing and malware distribution.

2.1 Fast Flux

Fast Flux (FF) is a usual method to avoid botnet recognition. Mapping multiple IP addresses with one single domain name is a fast process implemented by using IP fast fluxing, which is a more advanced technique to block the C&C server. Networks which implement FF approach are termed as fast fluxing network (FFN). Time to live (TTL)s and large IP pools also show the same characteristics as FFNs whether they belong to suspicious or legal FF networks. Fast flux methods fall under two main categories: single flux and double flux.

In single flux method, a domain could be determined to access a variety of IPs in varying time stamps. As an example, let us consider how a user can access the same domain two times within a short time period. In the first instance, the user queries to the DNS server, which determines that the matching IP address is IP-1 (e.g. 11.11.11.11). With the IP-1, the user gets the access to the fluxing agent FA-1, which switches the request to the actual server" mothership". The request is processed and responded by the mothership to FA-1. Finally, the user gets response from the FA-1. After some time, the user gets access to the same domain again. However, since IP-1 has expired because of the shorter TTL of IP addresses, the user requires to query the reconstructed IP address of the same domain at another time. Now, the DNS replies to the user with another IP address IP-2 (e.g. 22.22.22.22). Later, the user will make use of the updated address IP-2 to connect to the updated flux agent FA-2 which switches the user to the" mothership".

Double-flux is a more revolutionary way of botnet identification. Recurrent switching of the flux agents and the registration in DNS servers is followed in this method. Besides their individual agents, the commanding domain name server too is a part of the fluxing. This method contributes towards another slab of notice within malware networks. Registering and de-registering of the fluxing node takes place in the DNS repeatedly.

Fast fluxing network attack (FFNA) is a technique used by the attackers to maintain their botnets by abusing the above described fast flux technique. Approximately all the weakened computers turn into fluxing agents in this case.

2.2 Domain Flux

Once fluxing is identified, fast flux is the only point where failure occurs due to its unique domain name. This can be addressed by Domain Flux (DF), which is a better and outlived procedure. It has been pointed out by [22] that Torpig, a contemporary botnet program resolved this problem by using domain flux technique.

The process of domain flux has been explained by [22] in their research of taking over Torpig. Generating domains through the domain generation algorithms (DGA) is the base for the idea of Domain flux. Same algorithm will be followed by both the C&C server and its bots and produced by the same value to acquire constantly changing domain names. Bots will attempt to associate with the C&C server and others which are supervised by the botnet master using a list of domain names until one DNS query wins. If the present domain is obstructed by mastery domain, bots will attempt the different ways to calculate other domains by DGA.

The purpose of DF is that the procedure should ensure that all the bots should be able to generate the domains using same kernel [22]. disclosed that calculation of sub-domains by Torpig is done by the use of ongoing week and ongoing year first and does not depend on the ongoing day and attached by Top Level Domain (TLD). The generated domains could be" weekyear.com" or" weekyear.biz", etc. These auto-generated domain names will be used by bots to contact the C&C server. In case of failure, the day information will be used by the bots to calculate the day-to-day domain such as" day.com" or" day.net". The bots will finally attempt to utilize the inflexible domain names in a configuration file as a last resort.

2.3 Non-existent Domain (NX-Domain) Based Detection

The Non-Existent Domain (NX-Domain) based detection method assumes that the DGA domains will lead to a large number of NX Domain traffic, which could be analysed to look for any similarity in the NX domain response rate. Antonakakis [5] proposed a detection technique called Pleiadas which uses both the classification and clustering methods for identifying the domains generated by the DGAs. This method is useful when large number of hosts in a network are compromised with the same DGA bot.

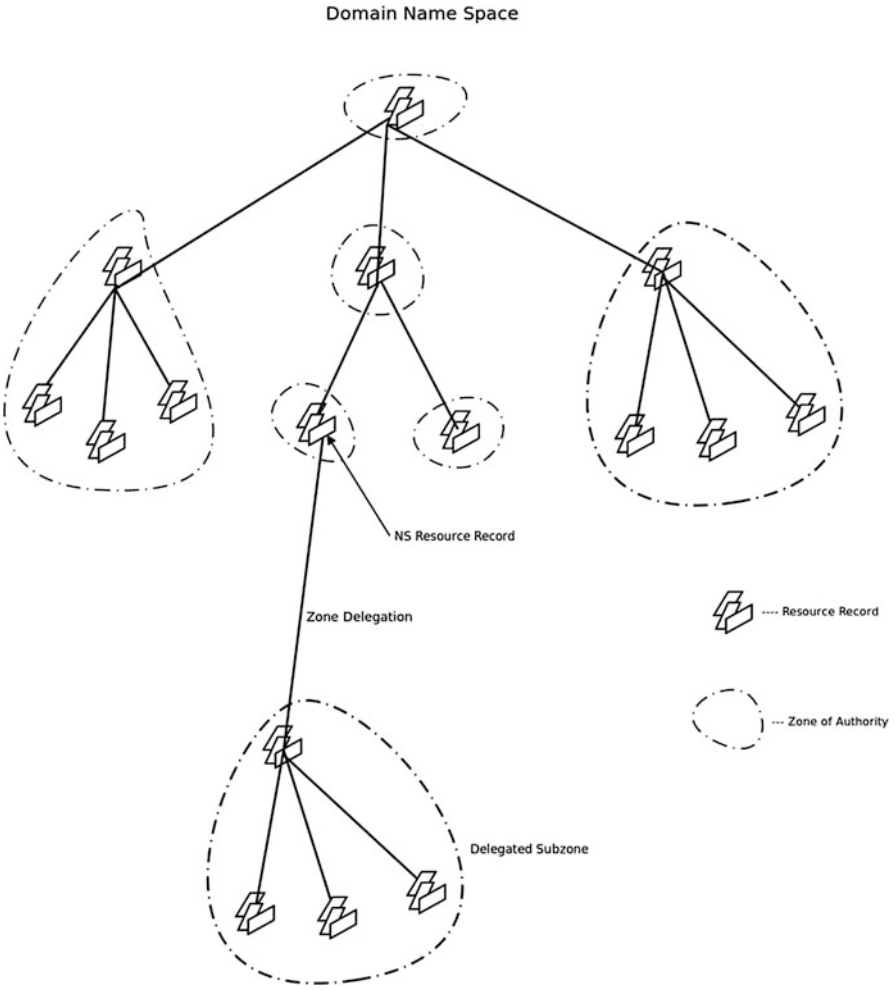


Fig. 1 Hierarchical domain name system

2.4 Domain Name System (DNS)

Domain Name System (DNS) forms an essential part of the internet protocol and maps domain names to IP addresses. Figure 1 shows the hierarchical levels in a DNS. When a website address is typed in the address bar of any browser, the DNS server will search through its database to find a matching IP address with that domain name and verifies if that domain name corresponding to the IP address of the website. Once that is established, the computer will be able to communicate with that web server and retrieve the webpage.

There are two types of DNS servers: Iterative and Recursive servers. Iterative servers or non-recursive servers that are responsible for the queries inside the domain. They will not query other DNS servers even though they cannot give the requested IP address. On the other hand, recursive servers are responsible for the queries of all types of domains which can query other servers [23]. The steps taken by the DNS are as follows [24]:

1. When a website name is typed in the web browser and if the web browser finds the IP address in its own cache memory, it will send the query to the next level which is called a Resolver Server.
2. The Resolver Server, which is an Internet Service Provider (ISP) will then check its own cache memory to find an IP address for the website and if it is not found, it will send the query to the next level which is the Root Server or the Root of the DNS hierarchy.
3. The Root Server directs the Resolver Server to the Top Level Domain (TLD) server for the IP address of the website. There are 13 sets of these Root Servers and they are strategically placed around the world and operated by 12 different organizations. Each set of these Root Servers has their own unique IP address. TLD stores the address information for the top level domains such as .COM .NET or .ORG etc.
4. The TLD will then direct the Resolver Server to the next and final level which are called Authoritative Name Servers. The Resolver Server checks with the Authoritative Name Server for the IP address of the website.
5. The Authoritative Name Server is the final authority responsible for knowing everything about the domain which includes the IP address. When the Authoritative Name Server receives a query from the Resolver Server, it will respond with the IP address for the required website which will be stored in the cache memory so that the above steps need not be repeated again for the same website.

Recursive DNS servers suffer from Distributed Denial of Service (DDOS) attacks. DDOS is a type of cyber-attack on a specific server or network with an intended purpose of disrupting normal operation. The DDOS network achieves this by flooding the target with a constant flood of traffic such as fraudulent request which overwhelms the system causing a disruption or denial of service to legitimate traffic.

3 DGA Detection Method

In this section we summarise related work in respect to existing techniques for DGA domain detection.

3.1 Machine Learning Based Detection Techniques

There is an extensive research literature that uses machine learning techniques directly to detect domains as generated by DGAs. One of the supervised learning methods that called Kullback-Leibler (KL) divergence. The KL-divergence is a non-symmetric technique used to calculate the distance between the probabilities P and Q.

In [25], a technique using support vector machine (SVM) algorithm for binary classification of the malicious domain names has been proposed. SVM consists of a hyper plane which divides the data into two separate classes for classification and it maps low dimension non-linear data inputs (x_i, x_j) into a higher dimension space using a kernel function (k). It can internally use a kernel trick to avoid the explicit mapping that is needed to get a learning algorithm to learn a nonlinear function. Data in n dimensions can be divided by n-1 hyper planes [25] maximizing the margin between the data points and the hyperplane. Typically, the SVM algorithm uses features such as domain name length, entropy, consecutive consonant ratio and proportion of digits in the domain names. Previous work has a combination of 2000 benign URLs and 2000 malicious domain names to form the dataset [26].

In general, machine learning algorithms including SVM algorithms exhibit the major drawback of involving a manual extraction of the features. Identifying the optimal features and feature engineering is a tedious and time-consuming task. However, with deep learning, the features are automatically extracted which saves time and effort, as well as results in higher accuracies for domain name detection [27].

3.2 Deep Learning Based Detection Techniques

Deep learning techniques adopt neural network models such as convolutional neural network (CNN) and recurrent neural network (RNN) architectures that have demonstrated remarkable performance in understanding natural language available in text. In this section, we study popular CNN, RNN and hybrid architectures that have been applied for DGA malicious detection.

3.2.1 RNN Based Architectures

The Long Short Term Memory (LSTM) networks proposed by [20] has been designed specifically for DGA malicious detection which is very close to the original LSTM network. This network consists of an embedding layer, an LSTM layer (128 LSTM cells with default Tanh activation) and a single node output layer with Sigmoid activation. For achieving better convergence results, the method by Adam [28] was adopted as the optimization algorithm, and RMSProp optimizer

was used in [20]. The dropout layer is recently adopted in deep learning to improve the model performance and overcome over-fitting as it serves to break up complex co-adaptations in the neural network. The embedding layer learns to represent each character which occurs in a domain name by a 128-dimensional numerical vector which is different from the original 128 dimensional ASCII encoding. The embedding layer learns by semantically mapping similar characters to similar vectors based on the required classification task. Comparisons of all these models could be done by keeping the dimensionality of the embedding space to be identical.

In other popular models such as the CMU model, the bidirectional recurrent neural network (RNN) processes the input string in two ways: left to right sequence is followed in the Forward layer for processing, while the right to Left sequence is followed in the backward layer. The combined output of the forward layer and backward layer is passed on to subsequent layers. [29] proposes bidirectional LSTMs for character level text processing. [30] proposes a very similar bidirectional Gated Recurrent units (GRUs) and applies in a Tweet2Vec model for tweet classification which predicts the hashtags of tweets.

Recently, [31] presented a measure of complexity for DGA families called the smashword score, to reflects how closely a DGA's generated domains resemble English words. Authors built a machine learning model consisting of recurrent neural networks (RNNs) using the generalized likelihood ratio test (GLRT), and augment these models with a logistic regression model that also includes side information. The combined model provide a good result on DGA families with high smashword score.

3.2.2 CNN Based Architectures

In Convolutional Neural Networks (CNN) by NYU model, the techniques are mainly used for processing the input data with a grid-like topology such as images. [32] applies 1-dimensional CNNs successfully for text classification at the character level and their proposed method is intended to process natural language text effectively. Their proposed architecture consists of 6 stacked CNN layers in which each subsequent layer consumes the output from the previous layer. In comparison with the natural language processing, domain names do not have an internal grammatical structure. Hence, original architectures from [32] results in overfitting of the model with the data. Therefore, we reduced the number of stacked CNN layers to two and decreased the size and the number of filters to suit the problem of domain identification and classification.

Invincea Model [33] proposes a classifier based on CNN that takes an input of generic short character strings and learns to detect the indicators of malicious behaviour. Examples of short character strings are registry keys, file paths or URLs. Instead of stacked layers, the Invincea model implements parallel CNN layers. Instead of a small pooling window, the pooling always happens over the entire domain name. Invincea model detects only the presence or absence of patterns in domain names. It does not retain information on where exactly these patterns occur

in domain name strings. In the Invincea model, the embedding layer is followed by a convolutional layer with 1024 filters, and 256 filters for each of the sizes 2,3,4 and 5. Every filter learns the detection of a soft presence of an interesting soft n-gram where n = 2,3,4,5. The convolutional layer is followed by two dense hidden layers, each with 1024 nodes. The final layer is the single node output layer with Sigmoid activation. We compared all these models, and among them Invincea model has the most expensive architecture.

3.2.3 Hybrid CNN and RNN Based Architecture

The NYU model has stacked CNN layers that are followed by an LSTM layer. The use of multiple stacked CNN layers resulted in the models to overfit with the data. To address this problem, the MIT model [34], an extension of the NYU model was introduced with a reduced hybrid architecture with one CNN layer followed by one LSTM layer.

4 Proposed DGA Malicious Detector Model

Our proposed DGA Malicious Detector Model consists of three phases for DNS traffic collection and pre-processing: (i) Training phase, (ii) Testing Phase and (iii) Deployment phase.

4.1 Training Phase

The training phase employs datasets consisting of the DNS traffic with the domain header captured using the port mirroring method [21]. Initially, data pre-processing and extraction of the domain names are done. The extracted domain names are labelled manually and the training dataset is prepared by saving them in a comma separated values (CSV) file. Various deep learning architectures are trained on the training dataset and the trained deep learning model is tested using two different datasets. The results achieved for the binary and multi-class classification are then analysed for quality assurance using appropriate metrics. The block diagram for both training phase and testing phase is shown in Fig. 2.

4.2 Testing Phase

In the testing phase, two different datasets with different characteristics are used for testing the robustness and reliability of the deep learning architectures

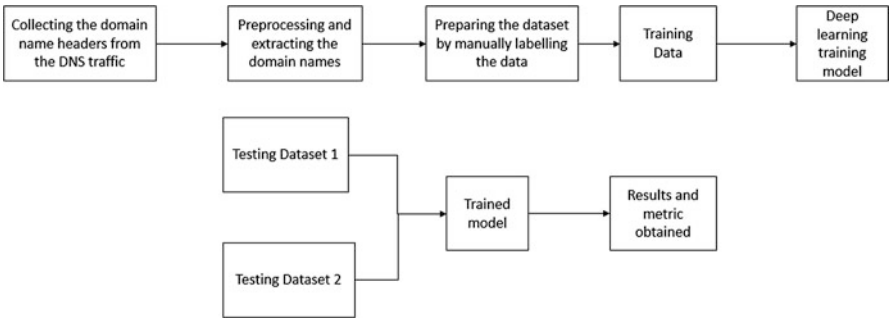


Fig. 2 Block diagram showing training phase and testing phase

independently. The trained deep learning model evaluates the two testing datasets for accuracy, precision, recall and F1 score metrics for both binary class and multi-class DGA malicious detection. The aim is to ensure that the proposed DGA malicious detection model performs well when deployed in a real network even when it is encountered by new domain names that are unseen before.

4.3 Deployment Phase

After the experiments, the best-performing algorithm's pretrained model is deployed for the real-time DNS traffic analysis. Once we feed the domain name it can predict whether the domain name is malicious or benign. If it is malicious, then it is blocked and added to the blacklist, and the benign domain names are allowed and added to the whitelist. The block diagram for the deployment phase is shown in Fig. 3.

5 Data Preparation

There are 2 different datasets used in this research, The detailed statistics of datasets described in Table 1. The first dataset, is a publicly available, the malicious domain names for this dataset are collected using the publicly available DGA algorithms,[1] OSINT feeds[2] and netlab-360.[3] The benign domain names are collected from the

[1] https://github.com/baderj/domain_generation_algorithms

[2] http://osint.bambenekconsulting.com/feeds/

[3] https://data.netlab.360.com/dga/

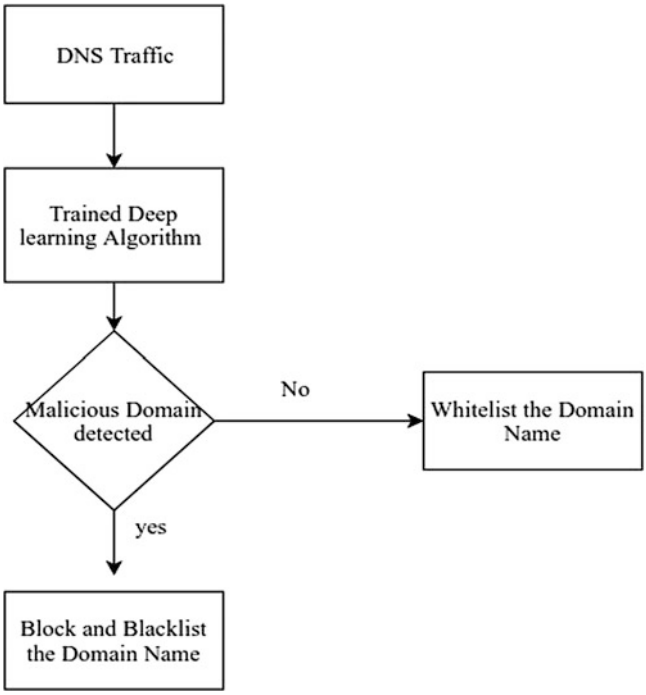


Fig. 3 Overview of DGA detector (deployment phase)

Table 1 Binary class dataset description

Type	Benign domain	Malicious domain	Total
Training	655,683	135,056	790,739
Testing dataset 1	2,349,331	108,076	2,457,407
Testing dataset 2	182	2740	2922

Alexa[4] and openDNSThe second dataset is collected privately within a lab using port mirroring approach.[5] We used 20 DGA algorithms in this experimental study.

The passive sensors were deployed in an internal network. The detailed experimental setup followed during collection of DNS traffic dataset from an internal network is reported in [8, 9, 21, 35, 36]. The DNS query response messages from the DNS servers are collected by passive sensor. Thus, in this work, DNS traffic was collected from different DNS servers. The sensors capture the network traffic from DNS servers, then they extract DNS packets and convert them into human-readable format. This data is passed into DNS log collector. The passive sensors could either install inside the DNS server itself or mirror only the DNS traffic to a server which is dedicated for DNS passive sensors. The sensors get the data by port mirroring

[4] https://support.alexa.com

[5] https://umbrella.cisco.com/blog

the traffic from the DNS servers present in the deployed network. The logs received from an internal network are passed to a distributed log parser. Dataset 2 is used for testing only. To increase the performance in detecting malicious domain names and to find various methods, this dataset is used as part of a shared task of the event, Detecting Malicious Domain Names (DMD 2018). The baseline system is available at the Github hosting service.[6] The detailed description of the baseline system and the submitted system details is discussed in detail by [37].

6 Experimental Analysis

We used various Deep Neural Networks, mainly, convolutional neural network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional Long Short-Term Memory (BiLSTM), Bidirectional Recurrent Neural Network (BiRNN) and CNN-LSTM layers deep learning architectures for the binary class and multi-class detection. An extensive study of the performance and efficiency of the proposed DGA Malicious Detector is conducted through rigorous experimentation and testing of the two different datasets.

6.1 Embedding Layer

A recent NLP techniques use the embedding for converting the words or characters into vector representations. Embedding libraries like word2vec and glove are used in natural language processing (NLP) [32]. By following the similar approach character level domain name encoding has been used. The domain names are tokenised into characters and a dictionary is formed by assigning each character unique id and the dictionary size is 39. Using dictionary, the domain names are transformed into domain name vectors by assigning an index of the character. The domain name vectors are of different length. To convert the domain name vectors into same length, a maximum length 100 used. The domain name vectors which are lesser than the maximum length are filled with 0 s and the domain name vectors which are more than the maximum length are discarded. This is passed as input to embedding layer specifically Keras embedding. This has 3 parameters such as embedding length, dictionary size and maximum length of the sequence. Embedding length is set to 128, dictionary size to 39 and maximum length to 100. Using Gaussian distribution the weights of embedding layer are set, passed into deep learning layers and are tuned along with other deep learning layers during back propagation. Specifically, Embedding layer helps in learning the context and semantics of domain names (Table 2).

[6] https://github.com/vinayakumarr/DMD2018

Table 2 Multi class dataset description

Domain type	Label	Training	Testing 1	Testing 2
benign	0	100,000	120,000	40,000
banjori	1	15,000	25,000	10,000
corebot	2	15,000	25,000	10,000
dircrypt	3	15,000	25,000	300
dnschanger	4	15,000	25,000	10,000
fobber	5	15,000	25,000	800
murofet	6	15,000	16,667	5000
necurs	7	12,777	20,445	6200
newgoz	8	15,000	20,000	3000
padcrypt	9	15,000	20,000	3000
proslikefan	10	15,000	20,000	3000
pykspa	11	15,000	25,000	2000
qadars	12	15,000	25,000	2300
qakbot	13	15,000	25,000	1000
ramdo	14	15,000	25,000	800
ranbyus	15	15,000	25,000	500
simda	16	15,000	25,000	3000
suppobox	17	15,000	20,000	1000
symmi	18	15,000	25,000	500
tempedreve	19	15,000	25,000	100
tinba	20	15,000	25,000	700

6.2 Configuration of Hyper Parameters for Deep Learning Layers

Appropriate tuning of hyper parameters is required for optimal training of the long short-term memory (LSTM), convolutional neural network (CNN), recurrent neural network (RNN), gated recurrent unit (GRU), bidirectional long short-term memory (BiLSTM), bidirectional recurrent neural network (BiRNN) and CNN-LSTM layers of the DGA malicious detection algorithm. In our experimental study, the numbers of memory units for these deep learning architectures are set to 128 and for the CNN model, 64 filters are used with size 3. A dropout layer with a regularization parameter of 0.1 is added for all the architectures for randomly ignoring selected neurons and for preventing co-adaption on training data. Further, the dropout layer prevents any overfitting of the final trained model.

6.3 Classification

Fully connected layer receives the inputs from dropout and primarily performs the classification. In fully connected layer all the neurons of previous layer have connection to every other neuron in next layer. In fully connected layer, we employed Sigmoid as a non-linear activation function for the binary class

which converts the weighted sum into class probabilities for neuron activation. We employed softmax function at the final layer which takes the unnormalised vector input and normalises it into probability distribution over predicted output classes. Binary cross entropy loss function is used for binary classification tasks and categorical cross entropy is used for multi class classification tasks. For all deep learning architectures, the number of epochs used for the binary class are 20 and 50 for multi-class.

7 Results and Evaluations

This section presents results and evaluations of implemented DNN.

We adopt commonly used quality metrics to compare the correctness and accuracy of the classification of our DGA malicious detection model. We use the following terms for determining the quality of the classification models:

True Positive (TP) – the number of data items correctly classified to the positive class.
True Negative (TN) – the number of data items correctly classified to the negative class.
False Positive (FP) – the number of data items wrongly classified to the positive class.
False Negative (FN) – the number of data items wrongly classified to the negative class.

Using these terms, we determine the true positive rate (TPR) and false positive rate (FPR), which are explained below. The ROC curve is obtained by plotting the TPR values against the FPR values and the accuracy is measured by the area under the ROC curve.

Accuracy – is the measure that gives total number of correct predictions made out of the all the predictions made by the model. It is a good measure to use when target classes are nearly balanced in the data.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision – is the measure for ability of the model to give quality positive predictions. It can be interpreted as the probability that positive prediction made by the model is positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall – measure is the ratio of the true positive (TP) divided by the total of true positive (TP) and false negative (FN) predictions.

$$Recall = \frac{TP}{TP + FN}$$

F1-score – measure is given by the harmonic mean between the precision and recall. It can be a better measure when target classes are unevenly distributed.

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

True positive rate (TPR) is a quality metric that reflects the ability of the DGA malicious detection model to detect malicious domain names using the deep learning classifiers. TPR is the ratio between the true positive values and the total of true positive and false negative values.

$$TPR = \frac{TP}{TP + FN}$$

False positive rate (FPR) is a quality metric that reflects the frequency exhibited by the DGA malicious detection model in making a mistake to identify non-malicious domain names as malicious using the deep learning classifiers. FPR is the ratio between the false positive values and the total false positive and true negative values.

$$FPR = \frac{FP}{FP + TN}$$

We conducted an experimental comparison of the performance of the deep learning architectures considered in this research study. The detailed summary of the results obtained is shown in Table 3 for the binary class DGA malicious detection, and in Table 4 for the multi-class DGA malicious detection. From Table 3 it is inferred that CNN-LSTM architecture gives highest accuracy, recall, precision and F1 score for binary class with dataset Testing 1 and RNN architecture gives highest accuracy for binary class with dataset Testing 2. Similarly, from Table 4 it is inferred that LSTM architecture gives highest accuracy for multi-class with public source dataset (Dataset 1) and GRU architecture gives highest accuracy for multi-class with private source dataset (Dataset 2). For training and testing the models, Keras[7] deep learning library with TensorFlow[8] as backend were used on a clustered computing server environment.

[7]Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. https://keras.io/

[8]TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks, https://www.tensorflow.org/

Table 3 Proposed method – performance of models for binary class DGA malicious detection

Model	Accuracy	Recall	Precision	F1 Score	AUC
Binary class with testing dataset 1					
LSTM	0.987	0.787	0.955	0.863	0.9969
CNN	0.968	0.598	0.846	0.701	0.9965
RNN	0.978	0.689	0.93	0.792	0.9909
GRU	0.986	0.779	0.944	0.854	0.9965
BiLSTM	0.987	0.781	0.964	0.863	0.9977
BiRNN	0.98	0.712	0.925	0.804	0.9927
CNN-LSTM	0.988	0.807	0.96	0.877	0.9977
Binary class with testing dataset 2					
LSTM	0.713	0.999	0.694	0.819	0.9545
CNN	0.67	0.998	0.649	0.787	0.9558
RNN	0.783	0.999	0.77	0.87	0.9694
GRU	0.715	0.999	0.696	0.821	0.9558
BiLSTM	0.728	0.998	0.711	0.831	0.9603
BiRNN	0.758	1.00	0.742	0.852	0.9718
CNN-LSTM	0.711	0.999	0.693	0.818	0.9619

Table 4 Proposed method – performance of models for multi-class DGA malicious detection

Model	Accuracy	Recall	Precision	F1 Score
Multi class with testing dataset 1				
LSTM	0.683	0.683	0.683	0.64
CNN	0.661	0.661	0.8	0.707
RNN	0.661	0.661	0.771	0.697
GRU	0.677	0.677	0.822	0.725
BiLSTM	0.662	0.662	0.774	0.695
BiRNN	0.668	0.668	0.772	0.703
CNN-LSTM	0.666	0.666	0.82	0.715
Multi class with testing dataset 2				
LSTM	0.67	0.67	0.678	0.622
CNN	0.671	0.671	0.876	0.717
RNN	0.654	0.654	0.844	0.691
GRU	0.672	0.672	0.896	0.713
BiLSTM	0.663	0.663	0.864	0.702
BiRNN	0.664	0.664	0.845	0.706
CNN-LSTM	0.67	0.67	0.884	0.711

Compared Proposed Method with the other submissions of DMD 2018 shared task in Tables 5 and 6 and the detailed results are available at.[9] The submitted system details of DMD 2018 shared task is given below along with the team name and the

[9] https://nlp.amrita.edu/DMD2018

Table 5 Comparison for binary-class DGA malicious detection [37]

Team name	Accuracy	Recall	Precision	F1 -Score
Binary class with testing dataset 1				
UWT	0.99	0.828	0.966	0.89
DeepDragons CNN- LSTM (proposed)	0.988	0.807	0.96	0.877
CHNMLRG	0.988	0.819	0.944	0.88
BENHA	0.963	0.795	0.199	0.32
bharathibSSNCSE	0.615	0.037	0.311	0.07
UniPI	0.981	0.724	0.919	0.81
Josan	0.989	0.822	0.947	0.88
DeepDGANet	0.976	0.658	0.938	0.773
Binary class with testing dataset 2				
UWT	0.766	0.999	0.751	0.86
DeepDragons RNN (proposed)	0.783	0.999	0.77	0.87
CHNMLRG	0.787	0.999	0.774	0.87
BENHA	0.564	0.974	0.55	0.7
bharathibSSNCSE	0.562	0.956	0.559	0.71
UniPI	0.714	0.999	0.696	0.82
Josan	0.711	0.999	0.692	0.82
DeepDGANet	0.782	0.997	0.769	0.869

Table 6 Comparison for multi-class DGA malicious detection [37]

Team name	Accuracy	Recall	Precision	F1- Score
Multi class with testing dataset 1				
UWT	0.633	0.633	0.618	0.602
DeepDragons LSTM (proposed)	0.683	0.683	0.683	0.64
CHNMLRG	0.648	0.648	0.662	0.6
BENHA	0.272	0.272	0.194	0.168
bharathibSSNCSE	0.18	0.18	0.092	0.102
UniPI	0.655	0.655	0.647	0.615
Josan	0.697	0.697	0.689	0.658
DeepDGANet	0.601	0.601	0.938	0.576
Multi class with testing dataset 2				
UWT	0.887	0.887	0.924	0.901
Deep_Dragons GRU (proposed)	0.672	0.672	0.896	0.713
CHNMLRG	0.674	0.674	0.683	0.648
BENHA	0.429	0.429	0.34	0.272
bharathibSSNCSE	0.335	0.335	0.229	0.223
UniPI	0.671	0.671	0.641	0.619
Josan	0.679	0.679	0.694	0.636
DeepDGANet	0.531	0.531	0.653	0.541

description of proposed method. The detailed description of these models and DMD 2018 shared task baseline system is discussed in detail by [37].

UWT has performed an analysis of several machine learning based approaches with featureful Approach using 28 lexical/linguistic features along with different standard deep learning architectures. Random forest classifier and deep learning based Endgame (LSTM), Invincea (CNN), CMU (LSTM), MIT (CNN + LSTM), and NYU (CNN) architectures. Along with the shared task dataset they have also used Alexa-Bambenek, DGArchive, Real-time stream of DNS data from Farsight Security datasets.

CHNMLRG has used a transfer learning approach by combining CNN with Naive Bayes classifier. Learnt features from the last fully connected layer were used as input for the Naive Bayes classifier.

BENHA has used the frequency of specific characters such as digits, alphabets, spaces and dots as features. In addition to the length of the URL. Two classifiers have been designed based on Support Vector Machines and Multi-layer perceptron (MLP). SVM uses Radial Basis Function (RBF) kernel. MLP uses Adam optimization algorithm and limited-memory BFGS (LBFGS).

bharathibSSNCSE used a LSTM and Bidirectional LSTM deep learning architectures with 128 dimensional numerical vector along with a embedding layer for representing the individual characters in domain names.

UniPI Also used the Bidirectional LSTM architecture with embedding layer consisting of 32, 64, 128, 256 cell sizes.

Josan used two models first one consists of a single layer Bidirectional LSTM network architecture and also a two layer Bidirectional LSTM network. Randomly initialised character embeddings are used during learning phase.

DeepDGANet used a 3 gram representation for transforming the domain names along with feature hashing and applied a deep neural network (DNN) with 5 hidden layers. Dropout of 0.0001 is used for regularisation and batch normalisation is used for speeding up the training process.

From Table 5 we can observe that for the binary class testing dataset 1 architecture CNN-LSTM performed well and on the testing dataset 2 RNN architecture performed well. From this analysis we can infer that sequential models performed well because of the ability to capture the long-term temporal dependencies and relationships between the domain name characters. The code and datasets along with detailed results are available in the GitHub repository.[10]

All the different deep learning architectures were able to give good results for the Binary class classification testing dataset 1 which consists of the domains collected

[10] https://github.com/dineshresearch/Real-Time-Character-Level-Malicious-DomainName-Prediction-Using-Deep-Learning

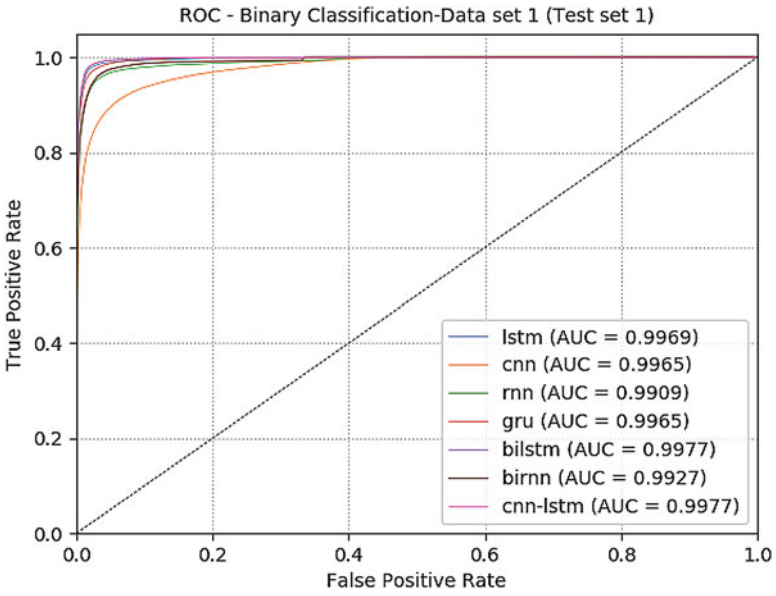


Fig. 4 ROC curve of binary class testing for dataset 1

from the publicly available sources. But comparatively failed to perform well for the testing dataset 2 which consists of the domains collected from the real time internal network traffic. For the zero day DGA detection performing well on this dataset is important.

The ROC curve plots two parameters, TPR and FPR showing the performance of a classification model at all classification thresholds. It signifies the diagnostic ability of the binary classifiers and it shows the ability of the classifier in distinguishing between the classes. It is plotted by the predicted probabilities of the algorithm, namely TPR and FPR. The area under the curve (AUC) gives the performance of different classifiers. BiLSTM and CNN-LSTM have the highest AUC of 0.9977 for Dataset 1 and BiRNN have the highest AUC of 0.9718 for the Dataset 2. Figures 4 and 5 shows the comparison of the AUC and the ROC for Dataset 1 and Dataset 2 of the binary classification for the DGA malicious detection model.

8 Conclusion and Future Work

Detecting malicious DGAs is a challenge and deep learning techniques show promise in this research domain. This research advanced further towards effective detection of malicious domain names generated using DGAs by employing various deep learning architectures. An overview of deep learning approaches provided insights into the problem of detecting malicious domain names accurately. A prac-

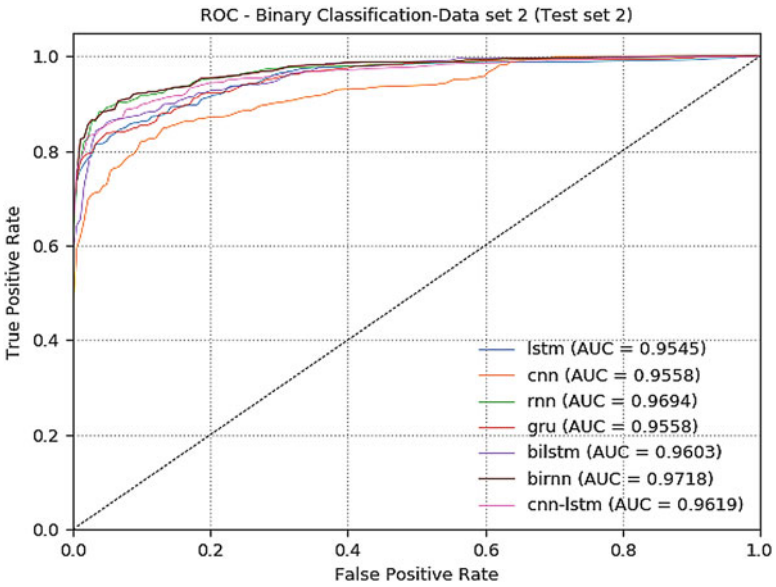


Fig. 5 ROC curve of binary class testing for dataset 2

tical real-time extendable and scalable framework was developed for the malicious domain name detection using popular deep learning models. The advantage of the proposed DGA malicious detection algorithm is its adaptability even when the behaviour of the malware is modified or the malware is updated. The detection is based on the external DNS traffic analysis rather than the internal structure of the malware.

The proposed framework has demonstrated efficiency in the detection of malicious domain names generated using DGAs. It can be deployed to block the bots from external communications and commands from the C&C server. However, its purpose is not to identify the structure and internal behaviour of the malware. Overall, the framework performed well using standard quality metrics and can serve as an external DNS traffic analysis system for detecting the malicious and benign domains. This framework is openly available and the DGA malicious detector can be extended further by applying the big data framework and tools. Future work involves handling huge data by storing and processing the data in a big data platform like Apache Spark and developing a front-end user interface for the real-time visualization of data using dashboards.

Acknowledgement This work was supported by the Department of Corporate and Information Services, Northern Territory Government of Australia, the Paramount Computer Systems, and Lakhshya Cyber Security Labs. We would like to thank NVIDIA India, for the GPU hardware support to research grant. We are also grateful to Computational Engineering and Networking (CEN) department for encouraging the research.

References

1. Broadhurst R, Grabosky P, Alazab M, Bouhours B, Chon S (2014) An analysis of the nature of groups engaged in cyber crime. Int J Cyber Criminol 8(1):1–20
2. Alazab M, Venkatraman S, Watters P, Alazab M, Alazab A (2012) Cybercrime: the case of obfuscated malware. In: Global security, safety and sustainability & e-democracy. Springer, New York, pp 204–211
3. Alazab M, Broadhurst R (2016) Spam and criminal activity, Trends & issues in crime and criminal justice, No. 526. Australian Institute of Criminology, Canberra
4. Alazab M (2015) Profiling and classifying the behaviour of malicious codes. J Syst Softw, Elsevier 100:91–102
5. Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, Dagon D (2012) From throw-away traffic to bots: detecting the rise of DGA-based malware. In: Proceedings of the 21st USENIX conference on security symposium (Security'12). USENIX Association, Berkeley, pp 24–24
6. Schiavoni S, Maggi F, Cavallaro L, Zanero S (2014) Phoenix: Dgabased botnet tracking and intelligence. In: International conference on detection of intrusions and malware, and vulnerability assessment. Springer, Cham, pp 192–211
7. Lison P, Mavroeidis V (2017) Automatic detection of malware-generated domains with recurrent neural models. arXiv preprint arXiv:1709.07102
8. Vinayakumar R, Soman KP, Poornachandran P (2018) Evaluating deep learning approaches to characterize and classify malicious urls. J Intell Fuzzy Syst 34(3):1333–1343
9. Vinayakumar R, Soman KP, Poornachandran P (2018) Detecting malicious domain names using deep learning approaches at scale. J Intell Fuzzy Syst 34(3):1355–1367
10. Vinayakumar R, Soman KP, Poornachandran P, Mohan VS, Kumar AD (2019) ScaleNet: scalable and hybrid framework for cyber threat situational awareness based on DNS, URL, and email data analysis. J Cyber Secur Mobil 8(2):189–240
11. Geffner J (2013) End-to-end analysis of a domain generating algorithm malware family. Black Hat USA
12. Plohmann D, Yakdan K, Klatt M, Bader J, GerhardsPadilla E, A comprehensive measurement study of domain generating malware
13. Pereira M, Coleman S, Yu B, DeCock M, Nascimento A (2018) Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic. In: International symposium on research in attacks, intrusions, and defenses. Springer, pp 295–314
14. Baruch M, David G (2018) Domain generation algorithm detection using machine learning methods. Springer, Cham, pp 133–161
15. Kuhrer M, Rossow C, Holz T (2014) Paint it black: evaluating the effectiveness of malware blacklists. In: International workshop on recent advances in intrusion detection. Springer, pp 1–21
16. Zhauniarovich Y, Khalil I, Yu T, Dacier M (2018) A survey on malicious domains detection through dns data analysis. arXiv preprint arXiv:1805.08426
17. Azab A, Alazab M, Aiash M (2016) Machine learning based botnet identification traffic. In: *Trustcom/BigDataSE/I SPA, 2016 IEEE*. IEEE, pp 1788–1794
18. Yadav S, Reddy AKK, Reddy AL, Ranjan S (2010) Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM conference on internet measurement. ACM, New York, pp 48–61
19. Yadav S, Reddy AKK, Narasimha Reddy AL, Ranjan S (2012) Detecting algorithmically generated domain-flux attacks with dns traffic analysis. IEEE/ACM Trans Networking 20(5):1663–1677
20. Woodbridge J, Anderson HS, Ahuja A, Grant D (2016) Predicting domain generation algorithms with long short-term memory networks. CoRR, abs/1611.00791
21. Vinayakumar R, Poornachandran P, Soman KP (2018) Scalable framework for cyber threat situational awareness based on domain name systems data analysis. In: Roy S, Samui P, Deo R, Ntalampiras S (eds) Big data in engineering applications. Studies in big data, vol 44. Springer, Singapore, pp 113–142

22. Stone-Gross B, Cova M, Cavallaro L, Gilbert B, Szydlowski M, Kemmerer R, Kruegel C, Vigna G (2009) Your botnet is my botnet: analysis of a botnet takeover. In: Proceedings of the 16th ACM conference on Computer and Communications Security, CCS '09. ACM, New York, pp 635–647
23. Bilge L, Sen S, Balzarotti D, Kirda E, Kruegel C (2014) Exposure: a passive DNS analysis service to detect and report malicious domains. ACM Trans Inf Syst Secur 16(4):1–28. https://doi.org/10.1145/2584679
24. Zhou Y, Li Q-s, Miao Q, Yim K (2013) Dga-based botnet detection using dns traffic. J Internet Serv Inf Secur 3(3/4):116–123
25. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. (2003) A practical guide to support vector classification
26. Wang Z, Jia Z, Zhang B (2018) A detection scheme for dga domain names based on svm. In: Proceedings of the 2018 international conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018). Atlantis Press
27. Arel I, Rose DC, Karnowski TP et al (2010) Deep machine learning-a new frontier in artificial intelligence research. IEEE Comput Intell Mag 5(4):13–18
28. Kingma DP, Adam JB (2014) A method for stochastic optimization. CoRR, abs/1412.6980
29. Plank B, Søgaard A, Goldberg Y (2016) Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. CoRR, abs/1604.05529
30. Dhingra B, Zhou Z, Fitzpatrick D, Muehl M, Cohen WW (2016) Tweet2vec: character-based distributed representations for social media. CoRR, abs/1605.03481
31. Curtin RR, Gardner AB, Grzonkowski S, Kleymenov A, Mosquera A (2019) Detecting DGA domains with recurrent neural networks and side information. In: Proceedings of the 9th ACM conference on data and application security and privacy (in submission to CODASPY'19). ACM, New York
32. Zhang X, Zhao JZ, LeCun Y (2015) Character-level convolutional networks for text classification. CoRR, abs/1509.01626
33. Saxe J and Berlin K (2017) expose: a character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. CoRR, abs/1702.08568
34. Vosoughi S, Roy D (2016) A semi-automatic method for efficient detection of stories on social media. In: Proceedings of the 10th International AAAI Conference on Weblogs and Social Media (ICWSM 2016). Cologne, Germany
35. Vinayakumar R, Soman KP, Poornachandran P, Menon P A deep-dive on machine learning for cybersecurity use cases. In: Gupta B, Sheng M (eds) Machine learning for computer and cyber security: principle, algorithms, and practices. CRC Press, Boca Raton
36. Mohan VS, Vinayakumar R, Soman KP, Poornachandran P (2018) S.P.O.O.F net: syntactic patterns for identification of ominous online factors. In: IEEE symposium on security and privacy workshops. Conference Publishing Services, IEEE Computer Society, Los Alamitos, pp 258–263
37. Vinayakumar R, Soman KP, Poornachandran P, Alazab M, Thampi SM, AmritaDGA: a comprehensive data set for domain generation algorithms (DGAs). In: Big data recommender systems: recent trends and advances, Institution of Engineering and Technology (IET). (In Press)
38. Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S (2019) Deep learning approach for intelligent intrusion detection system. IEEE Access
39. Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Venkatraman S (2019) Robust intelligent malware detection using deep learning. IEEE Access
40. Vinayakumar R, Alazab M, Jolfaei A, Soman KP, Poornachandran P (2018 December) Ransomware triage using deep learning: twitter as a case study. In the 9th IEEE International Conference on Cyber Security and Communication Systems At: Melbourne, Australia. Springer, 2018. (In Press)

Intrusion Detection in SDN-Based Networks: Deep Recurrent Neural Network Approach

Tuan Anh Tang, Des McLernon, Lotfi Mhamdi, Syed Ali Raza Zaidi, and Mounir Ghogho

Abstract Software Defined Networking (SDN) is emerging as a key technology for future Internet. SDN provides a global network along with the capability to dynamically control network flow. One key advantage of SDN, as compared to the traditional network, is that by virtue of centralized control it allows better provisioning of network security. Nevertheless, the flexibility provided by SDN architecture manifests several new network security issues that must be addressed to strengthen SDN network security. So, in this paper, we propose a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) enabled intrusion detection system for SDN. The proposed approach was tested using the NSL-KDD and CICIDS2017 dataset, and we achieved an accuracy of 89% and 99% respectively with low dimensional feature sets that can be extracted at the SDN controller. We also evaluated network performance of our proposed approach in terms of throughput and latency. Our test results show that the proposed GRU-RNN model does not deteriorate the network performance. Through extensive experimental evaluation, we conclude that our proposed approach exhibits a strong potential for intrusion detection in the SDN environments.

Keywords SDN · Software-defined networking · Network security · Network intrusion detection · Machine learning · Deep learning

T. A. Tang (✉) · D. McLernon · L. Mhamdi · S. A. R. Zaidi
University of Leeds, Leeds, UK
e-mail: eltat@leeds.ac.uk; d.c.mclernon@leeds.ac.uk
l.mhamdi@leeds.ac.uk; s.a.zaidi@leeds.ac.uk

M. Ghogho
International University of Rabat, Rabat, Morocco
e-mail: m.ghogho@leeds.ac.uk

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_8

1 Introduction

1.1 Motivation

The current Internet architecture has been developed for nearly three decades and is now becoming an increasingly complex system. It is largely decentralised, autonomous and built from a large number of network devices such as routers, switches and numerous types of middleboxes (e.g., firewall, load balancing, etc.) with several complex protocols implemented on each of them. These network devices are traditionally developed by several manufacturers. Each manufacturer has their own designs, firmware and other software to operate their own hardware in a proprietary and non co-operative way. Consequently, current Internet architecture lacks the agility to respond to ever changing demands and the dynamic nature of modern day applications. Software Defined Networking (SDN) [21] is introduced as a promising architecture, enabling scalability and unprecedented flexibility in the configuration and deployment of network services. The separation of control and data planes provides more flexibility and greater control over the traffic flows. Nevertheless, as highlighted in [13], the SDN architecture also introduces various security issues pertaining to the control plane, the control-data interface and the control-application interface. Recently, as mentioned in [14] and [26], SDN security has become a serious concern and attracted significant interest.

An intrusion detection system (IDS) is one of the most important network security tools. IDSs can be broadly classified into two categories: Signature-based IDS and Anomaly-based IDS. The signature-based IDS uses the signature database of the previous attacks to detect the new attacks. This system gives a low false alarm rate but it fails to detect zero-day attacks. The operational efficiency of the signature-based IDS is strongly coupled with the integrity and freshness of signature information available in databases. Maintaining such databases incurs huge operational overhead and is difficult if not impossible to realize in real-time. The anomaly-based IDS tries to identify the observation that deviates from the baseline model. Thus, this system can detect zero-day attacks better than the signature-based IDS. Various approaches have been proposed for intrusion detection like artificial neural networks (ANNs), support vector machines (SVMs), and Bayesian approaches. However, these techniques have a quite high false alarm rate and associated computational cost as mentioned in [28]. Recently, Deep Learning (DL) has emerged as a new approach and achieved a huge success in computer vision and language processing areas. DL has the ability to process raw data and learn the high-level features on its own, and so DL delivers higher accuracy than traditional machine learning techniques. The flow-based nature of SDNs enables network information acquisition in real-time via the OpenFlow [16] protocol. Consequently, flow-based intrusion detection systems have recently attracted significant attention in the context of SDNs.

1.2 Contribution

Following the trajectory of current research, we believe that deep recurrent neural networks can potentially offer better solutions for implementation of IDS under the context of SDN. Recurrent Neural Networks (RNNs) have shown great success in language modelling, text generating and speech recognition. We believe that the RNN is a powerful technique that can represent the relationship between a current event and previous events and then enhance the anomaly detection rate. In this chapter, a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) is proposed to detect the anomaly traffic traces.

In summary, the major contributions of this chapter are the following:

- We introduce an IDS in the SDN environment using GRU-RNN. To the best of our knowledge, this is the first attempt to use GRU-RNN for intrusion detection in the SDN environment.
- Our GRU-RNN approach yields a detection rate of 89% in the NSL-KDD dataset using a minimum number of features compared to other state-of-the-art approaches. This gives our approach significant potential for real time detection. The GRU-DNN achieves an impressing detection rate of 99% dealing with DDoS attacks in the CICIDS2017 dataset.
- We also evaluate the network performance of our proposed approach in the SDN environment. Then results show that our approach does not significantly degrade the SDN controller's performance.

The remainder of this paper is organized as follows. Section 2 presents a literature review. In Sect. 3, we give a system description. Section 4 presents the intrusion detection performance and network performance analysis. Finally, Sect. 5 concludes the paper and presents future work.

2 Literature Review

2.1 What Is Software Defined Networking?

The idea of programmable networks has been proposed as a way to facilitate the evolution of the conventional network. The idea of programmable networks and decoupled control logic has been around for several years. In general, SDN is built under four principles:

- *Seperation of Control and Data Plane*: these planes must be logically separated and connected via an interface. The control aspect is removed from forwarding devices and delegated to an external entity.

- *Network Programmabitity*: The provision of open API is a core aspect of the SDN architecture. Software and scripts should be able to access, configure, and modify network elements with ease.
- *Network Abstraction*: the view of the network is virtualized for any elements of a higher hierarchy. Services and applications are aware of the state of the whole network, but physical attributes and resources are irrelevant for configurations and computations.
- *Logically Centralized Control*: all forwarding devices of a domain are linked to a controlling entity and are subjected to its enacted policies.

SDN is defined by the Open Networking Foundation (ONF) which was founded in 2011 by Microsoft, Google, Facebook, Yahoo, Verizon and Deutsche Telekom. As of 2015 the organization has more than 150 industry members and receives endorsement by several network equipment vendors such as Cisco, Dell, Brocade and HP. An SDN architecture decouples the network control and forwarding functions enabling the network control to become directly programmable. The separation of the control plane from the data plane lays the ground for the SDN architecture. Network switches become simple forwarding devices and the control logic is implemented in a physical or logical centralized controller. The logical centralized controller dictates the network behaviour and offers several benefits. Firstly, it is simpler and less error-prone to modify network policies through software from a single place without reconfiguring individual devices. Secondly, a control program can automatically react to dynamic changes of the network and thus maintain the high level policies in place. Thirdly, the centralised control logic has global knowledge of the whole network, including the network topology and the state of the network resources, thus giving the flexibility and simplifying the development of more sophisticated network functions. For example, the controller can dynamically adjust flow tables to avoid congestion or apply different routing algorithms to different types of traffic.

The SDN architecture is divided into three layers: infrastructure layer, control layer and application layer.

- Infrastructure layer (Tier-1): This layer consists of the forwarding hardware such as switches/routers.
- Control layer (Tier-2): Network intelligence is installed in software-based logically centralized SDN controllers. The control layer regulates and manages forwarding hardware. The controller is the core of SDN networks. It lies between network devices at the one end and the applications at the other end. The SDN controller takes the responsibility of establishing every flow in the network by installing flow entries on switch devices.
- Application layer (Tier-3): Application and services take advantage of the control and infrastructure layer. Conceptually the application layer is above the control layer and enables easy development of network applications. These applications perform all network management tasks. Some examples of SDN application are load balancers, network monitors, and IDSs.

Table 1 SDN threat vectors

No.	Threat vector
1	Forged or faked traffic flow
2	Attacks on vulnerabilities in switches
3	Attacks on control plane communications
4	Attacks on and vulnerabilities in controllers
5	Lack of mechanisms to ensure trust between the controller and management application
6	Attacks on and vulnerabilities in administrative stations
7	Lack of trusted resources for forensics and remediation

2.2 Security in SDN

The SDN concept was originally designed with significant advantages over the traditional networking. Even though SDN brings significant advantages to network security, it also introduces new targets for potential attacks. The main causes of concern actually lie in SDN's main benefits: network programmability and control logic centralization. These capabilities actually introduce new fault and attack planes, which open the doors for new threats that did not exist before or were harder to exploit. According to Kreutz et al. [14], there are seven main potential threat vectors identified in SDN and summarized in Table 1.

Among these seven threat vectors, number 3, 4 and 5 are not present in traditional networking. They are specific to SDNs as they arise from the separation of the control and data plane and the logical centralization of the controllers. Other vectors were already presented in traditional networking. Threat vector number 5 would have the most severe impact on SDN architecture because it could affect the entire network. The controller emerges as a potential single point of attack. Attackers can attack vulnerabilities of controllers and run several dangerous scripts. Therefore, if there are no security settings to protect the controller, it would be under the control of attackers. The controller provides an interface for SDN applications to manage the network. However, this also give chances for malicious SDN applications to take over the network because of the lack of trust between the controller and SDN applications. Malicious hosts also can cause severe damage to the SDN architecture. Malicious hosts can perform Denial of Service (DoS) attacks to controllers and other hosts or network topology poisoning. DoS attack is one of the most dangerous attacks in SDN. It can be done by flooding the network with a large number of forged packets. These packets would trigger the switches to send a large number of requests to the controller for new flow rules. Therefore the control channel bandwidth and the controller CPU resources will be heavily consumed. As a result, the controller would respond slowly to legitimate requests. At the same time, the switches would also suffer from traffic congestion because the packets could quickly exhaust the memory of the flow table storage in the switches. Compromised switches not only have the same capabilities as the malicious hosts, but they are also capable of performing more dynamic and severe attacks. Firstly, they can be used for traffic eavesdropping.

Both data and control flows passing through the compromised switches can be replicated and sent to the attacker for further processing. Furthermore, the attacker can interfere with the control traffic passing through the compromised switches to perform man-in-the-middle attacks. By doing so, the attacker can act as the controller to some target switches.

2.3 Related Work

Researchers have employed classical machine learning approaches such as SVM, K-Nearest Neighbour (KNN), ANN, Random Forest, etc., for intrusion detection for several years. These proposed methods have achieved various degrees of success while also exhibiting some inherent limitations. Parwez et al. [22] employ K-means and hierarchical clustering to detect anomalies in call detail records (CDRs) of mobile wireless networks data. In [19], Bayesian networks are employed for anomaly and intrusion detection such as DDoS attacks in cloud computing networks. Bang et al. [2] use Hidden semi-Markov models to detect LTE signalling attacks. Principal Component Analysis (PCA) and SVM are also combined in [10] to detect intrusions. PCA is used for dimensional reduction on network data and SVM is used to detect intrusion on those data. These work only focus on traditional network with a large set of features that cannot be applied to SDNs. These classical mechanisms are still employed for intrusion detection in the context of SDN.

Braga et al. [3] present a lightweight approach using a Self Organizing Map (SOM) to detect Distributed Denial of Service (DDoS) attacks in the SDN. This approach based on six traffic flow features gives quite high detection accuracy. In [17], the authors use four traffic anomaly detection algorithms (threshold random walk with credit based rate limiting, rate limiting, maximum entropy and NETAD) in the SDN environments. The experiments indicate that these algorithms perform better in the SOHO (Small Office/Home Office) network than in the ISP (Internet Service Provider) and they can work at line rates without introducing any new performance overhead for the home network.

In the literature, the DDoS attacks are some of the most focused attacks for the SDN context because of their severe effects. The controller is a single point of failure in the SDN architecture. Therefore, if intruders trigger the DDoS attacks on the controller and take control of it, they can also control the whole network. SVM is also a quite popular algorithm for its high detection accuracy. In [12] and [24], SVM is used to detect DDoS attacks quite efficiently. Winter et al. [31] train a one-class SVM with a malicious dataset in order to reduce the false alarm rate. K-Nearest Neighbour and graph theory are combined to classify DDoS attacks from benign flows in SDNs by AlEroud et al. in [1]. Mousavi et al. [18] propose an early DDoS attack detection method against the SDN controller based on the variation of the entropy of the flow's destination IP addresses. They assume that the destination IP addresses are evenly distributed for the benign flows, while the malicious flows are destined for a small amount of hosts. Thus, the entropy drops dramatically when any attack happens. In [20], the authors propose a DL based approach using a stacked

autoencoder (SAE) for detecting DDoS attacks in the SDN. They achieve a quite high accuracy rate and low false alarm rate on their own dataset.

In 2016, we applied a Deep Neural Network (DNN) under the context of SDNs to train and test the NSL-KDD dataset [29]. We obtained a potential accuracy of 75.75% with just six basic features. In this paper, we continue this trend by using GRU-RNN to improve the detection accuracy and reduce the false alarm rate of the system.

3 Methodology/System Description

In this section, the Recurrent Neural Network (RNN) and Gated Recurrent Units (GRUs) are briefly reviewed. Then the architecture of the SDN-based IDS is described in detail.

3.1 Recurrent Neural Networks

A RNN is an extension of a conventional feed forward neural network. In general, a neural network architecture is as shown in Fig. 1.

The RNN is called "recurrent" because it performs the same task for every element of a sequence, with the output being dependent on the previous computations. Mathematically, the hidden states of the RNN are computed as:

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h),\ t = 1, 2, \ldots, T, \tag{1}$$

where $\sigma(\cdot)$ is a non-linearity function, $\mathbf{x}_t$ is an input vector at time t, $\mathbf{h}_t$ is a hidden state vector at time t, $\mathbf{W}$ is an input to hidden weight matrix, $\mathbf{U}$ is a hidden to hidden weight matrix, and $\mathbf{b}_h$ is a vector bias term.

The Backpropagation Through Time (BPTT) algorithm is used for training the RNN. However, BPTT for the RNN is usually difficult due to a problem known as vanishing/exploding gradient [9]. Long Short Term Memory (LSTM) [8] networks and GRUs [5] were proposed to solve this problem and are among the most widely used models in DL.

GRUs are selected in our research because of their simplicity and faster training phase compared to LSTMs [7].

For a GRU the activation $\mathbf{h}_t$ is computed differently from (1) as follows:

$$\mathbf{h}_t = \big(\underline{\mathbf{1}} - \mathbf{z}_t\big)\mathbf{h}_{t-1} + \mathbf{z}_t\tilde{\mathbf{h}}_t, \quad t = 1, 2, \ldots, T, \tag{2}$$

where an update gate $\mathbf{z}_t$ defines how much of the previous memory to keep and $\tilde{\mathbf{h}}_t$ is the candidate activation. The update gate is computed by

$$\mathbf{z}_t = \sigma\big(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}\big). \tag{3}$$

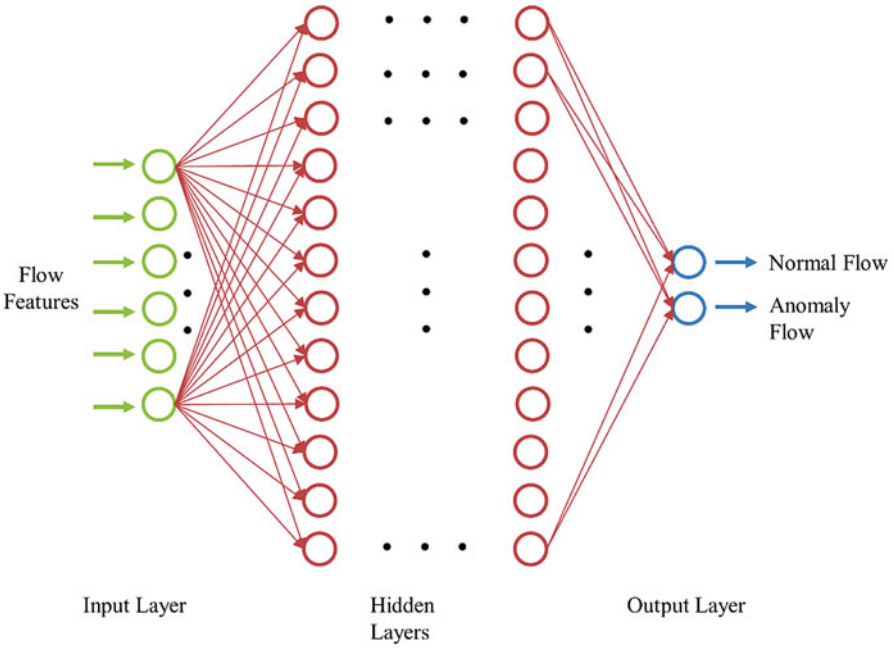


Fig. 1 The deep neural network structure

The candidate activation $\tilde{\mathbf{h}}_t$ is computed by

$$\tilde{\mathbf{h}}_t = \tanh\big(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{h}_{t-1} \odot \mathbf{r}_t)\big), \tag{4}$$

where a reset gate $\mathbf{r}_t$ determines how to combine the new input with the previous activation vector and $\odot$ is an element-wise vector multiplication. The reset gate is computed by

$$\mathbf{r}_t = \sigma\big(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}\big), \tag{5}$$

where $\sigma(\cdot)$ is again a non-linearity function in (3) and (5). Finally, all the $\mathbf{W}$ and $\mathbf{U}$ terms are learned weight matrices.

3.2 System Architecture

As mentioned earlier, the SDN decouples the control plane and data plane from network devices. The data plane, termed as switches, is just simple packet forwarding elements. The control plane can be either a single computer or a group of logically centralized distributed computers. The two entities communicate in order

to exchange network information and manage the whole network. In principle, the SDN network works in the following manner: packets from the internet traverse an SDN switch under the control of the SDN controller. Switches have flow tables for flow rules that contain match-fields, counters and actions that control traffic flows in the network. If the arriving flow does not match any rule in the SDN switch, the switch will send a *packet-in* message with the arriving flow header's information to the controller. The controller sends back a *packet-out* or *flow-mod* message to the switches to instruct them how to process the corresponding flow. Applications that run inside a controller can program the data plane for different purposes such as firewall, load balancer, IDS, etc. This paper focuses on the use of the SDN paradigm as network infrastructure for intrusion detection.

The IDS is implemented as an application on the SDN controller. The SDN-based IDS architecture is described in Fig. 2. It has three main components: Flow Collector, Anomaly Detector and Anomaly Mitigator.

- **Flow Collector**: This module is triggered when a *packet-in* message arrives. It will extract all the flow statistics such as protocol, source and destination IP and source and destination port. This module is also triggered by a timer function to send a *ofp_flow_stats_request* message to switches to request all the flow statistics information. Once the request is received, a *ofp_flow_stats_reply* message, which contains all the aggregated statistics of all flow entries, is sent back to the controller. All the features needed for anomaly detection will be created from these statistics and sent to the Anomaly Detector module.

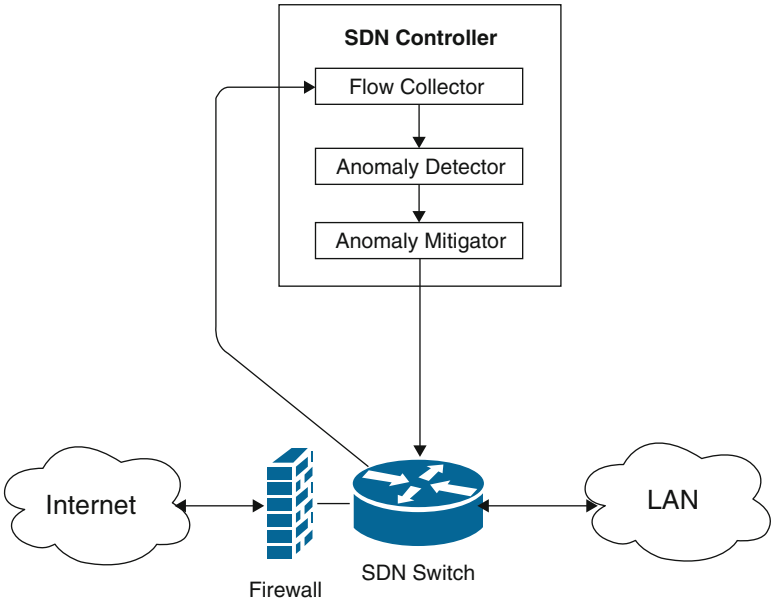


Fig. 2 SDN-based IDS architecture

- **Anomaly Detector**: We have chosen the GRU-DNN algorithm for the core of the Anomaly Detector module in this paper. The anomaly detector module loads a trained model, receives the network statistics and decides if a flow is an anomaly or not.
- **Anomaly Mitigator**: Through the Anomaly Detector's results, the Anomaly Mitigator module can make decisions on what to do with the flow. For example, it is possible to immediately stop the flow in order to prevent possible further attacks or to mirror the traffic to a deep packet inspector to further analyze the flow.

4 Performance Analysis

In this section, we firstly describe all datasets used in our experiment. Secondly, we explain all metrics used to evaluate our model performance. Thirdly, we describe all experimental setups. Detection results are given and compared with other works for a better overview in next section. Finally, the network performance of our GRU-RNN is evaluated and analyzed.

4.1 Datasets

Currently, there are only a few public datasets available for IDS evaluation (i.e., KDD Cup 99, NSL-KDD, DAPRA, and ISCX 2012) and none of them are specified for the SDN architecture. The KDD Cup 99 [11] and NSL-KDD datasets are some of the most popular datasets used in the literature to assess the NIDS performance. As mentioned in [15], the KDD Cup 99 has several inherent problems that makes the classifier fail to deliver a better accuracy. The NSL-KDD dataset [30] is introduced by Tavallaee et al. to solve these problems. However, the above datasets are out of date, lack traffic diversity and feature sets. Recently, the CICIDS2017 dataset [27] was published to deal with these issues. The SDN architecture is still under development, and so the NIDS dataset for the SDN is still quite rare and unpublished. Several researchers still use the conventional dataset to evaluate their approaches. In this paper, the NSL-KDD and CICIDS2017 datasets are chosen to evaluate our particular method.

4.1.1 NSL-KDD Dataset

The NSL-KDD contains 125,973 and 22,544 records in the training set and testing set respectively. Each traffic trace in this dataset has 41 features that are categorized

Table 2 Attacks in the NSL-KDD dataset

Category	Training set	Testing set
DoS	back, land, neptune, pod, smurf, teardrop	back, land, neptune, pod, smurf, teardrop, **mailbomb, processtable, udpstorm, apache2, worm**
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, **xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named**
U2R	buffer-overflow, loadmodule, perl, rootkit	buffer-overflow, loadmodule, perl, rootkit, **sqlattack, xterm, ps**
Probe	ipsweep, nmap, portsweep, satan	ipsweep, nmap, portsweep, satan, **mscan, saint**

Table 3 The NSL-KDD's feature description

Feature name	Description
duration	Length (number of seconds) of the connection
protocol_type	Type of the protocol (e.g. tcp, udp, etc.)
src_bytes	Number of data bytes from source to destination
dst_bytes	Number of data bytes from destination to source
srv_count	Number of connections to the same service as the current connection in the past two seconds
dst_host_same_src_port_rate	Number of connections that were to the same source port

into three types of features: basic, content-based and traffic-based features. Attacks in this dataset are categorized into four categories according to their characteristics. Our IDS is trained by the *KDDTrain+* dataset and tested by the *KDDTest+* dataset. In addition, the *KDDTest+* dataset contains 18 different types of attacks in addition to 22 attack types out of the *KDDTrain+* dataset. Details of each attack category are described in Table 2. Thus, the *KDDTest+* dataset is a reliable indicator to the performance of the model on zero-day attacks as well.

Within the context of SDN, the packet content is not directly accessible in the current OpenFlow protocol. The OpenFlow protocol does not allow us to get all the 41 features in the NSL-KDD dataset. This leads to a reduction of the features for the anomaly detection. So we just focus on the basic features and traffic-based features of the NSL-KDD dataset. Some of the features in these two categories can be easily retrieved from the SDN switches. In our research, a mixed feature set that contains six features from both the basic feature and traffic-based feature set are selected out of 41 features of the NSL-KDD dataset. The selected feature are *(duration, protocol_type, src_bytes, dst_bytes, srv_count, dst_host_same_src_port_rate).* These are SDN related features. Table 3 shows details of these features. They are selected based on their SDN related nature without any feature selection or optimization algorithms.

The NSL-KDD dataset contains both the numerical and symbolic features, and so we will transform the symbolic features into numerical values. After converting, the dataset is normalized into the range of [0–1] by Min-Max scaling as follows:

$$x^{'} = \frac{x - \min(x)}{\max(x) - \min(x)}, x > 0, \tag{6}$$

where $x^{'}$ is the normalized value and x is the original value.

4.1.2 CICIDS2017 Dataset

The CICIDS2017 dataset covers seven types of common attack families (i.e., Brute Force Attack, Heatbleed Attack, Botnet, DoS Attack, DDoS Attack, Web Attack, and Infiltration Attack). In this chapter, we choose a Wednesday dataset focusing on DoS, Heartblead, Slowloris, Slowhttptest, Hulk and GoldenEye Attacks. These types of attacks are on the rise and are a major threat to the SDN architecture.

In this dataset, we use 622,265 and 69,141 records for training and testing sets respectively. We extract a subset of nine features out of 80 features of this dataset for our research. These features have an SDN-related nature and can be extracted easily by SDN controllers. Details of these features can be seen in Table 4. This dataset is also normalized into the range of [0–1] by the Min-Max scaling as with the NSL-KDD dataset.

4.2 Evaluation Metrics

In order to evaluate our proposed approach, Precision (P), Recall (R), F1-measure (F1) and Accuracy (ACC) metrics are used. These metrics are calculated by using

Table 4 The CICIDS2017's feature description

Feature name	Description
Source port	Source port of the flow
Destination port	Destination port of the flow
Protocol	Protocol type of the flow
Flow duration	Duration of the flow in microsecond
Fwd packet length mean	Mean size of packets in forward direction
Flow bytes/s	Number of flow bytes per second
Flow packet/s	Number of flow packets per second
Flow IAT mean	Mean inter-arrival time of packets
Fwd packet/s	Number of forwarded packets per second

four different measures – True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), defined as follows:

- TP: the number of anomaly records correctly classified.
- TN: the number of normal records correctly classified.
- FP: the number of normal records incorrectly classified.
- FN: the number of anomaly record incorrectly classified.

So the four metrics are:
Accuracy (ACC): the percentage of true detection over total traffic records,

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%. \tag{7}$$

Precision (P): the percentage of predicted anomalous instances predicted are actual anomalous instances,

$$P = \frac{TP}{TP + FP} \times 100\%. \tag{8}$$

Recall (R): the percentage of predicted anomalous instances versus all the anomalous instances presented,

$$R = \frac{TP}{TP + FN} \times 100\%. \tag{9}$$

F1-measure (F1): the harmonic of the precision and recall metrics to express the performance of the model,

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \times 100\%. \tag{10}$$

4.3 Experimental Setup

According to our experiments, A DNN with three hidden layers gives best results in all experiment cases. Therefore, we propose a GRU-RNN with three hidden layers. For the training phase, the batch size and epoch number are 100 and 1000 respectively. We use a Nadam optimizer and Mean Squared Error (MSE) function for the model. In addition, we added L1-regularization to our model to prevent over fitting during the training phase. A DNN is also implemented with the same structure as the proposed GRU-RNN. We used Keras [6] to implement our GRU-RNN, DNN, and VanilaRNN models. Scikit-learn library [23] is used to implement the SVM algorithm and measure all the evaluation metrics.

Table 5 Performance metric evaluation for the NSL-KDD dataset

Class name	Precision (%)	Recall (%)	F1-measure (%)
Legitimate	87	89	88
Anomaly	91	90	90

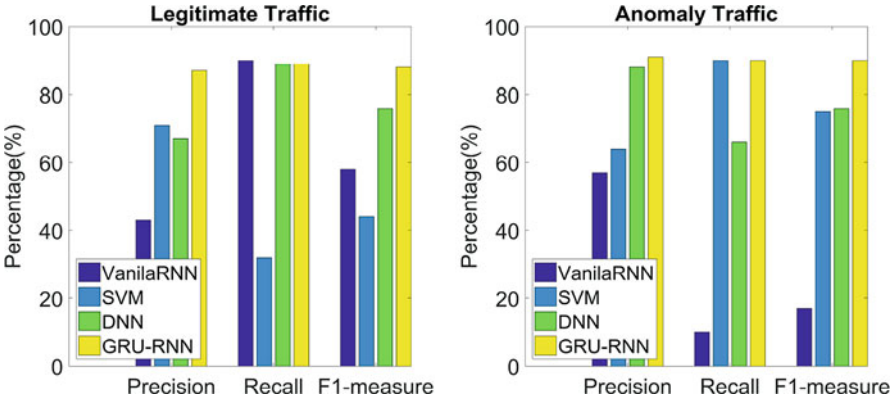


Fig. 3 Performance metric comparison

Table 6 Accuracy comparison with other algorithms

Algorithm	Accuracy
VanilaRNN	44.39%
SVM	65.67%
DNN	75.9%
GRU-RNN (Proposed Model)	**89%**

4.4 Experiment Results

To start with, we present the anomaly detection performance of our proposed model in terms of Precision, Recall, F1-measure and accuracy on the NSL-KDD dataset. Details of the results given in Table 5 show that our GRU-RNN performs well for all the evaluation metrics. Both the legitimate and anomaly traffic traces are detected really well by the GRU-RNN. The detection rates of the legitimate and anomaly traffic traces is 89% and 90% respectively. The anomaly detection accuracy of 90% shows that the GRU-RNN is good at detecting zero-day attacks.

We also compare the performance of our proposed model with other popular algorithms like VanilaRNN, SVM and DNN using the same subset of six features. As we can see in Fig. 3, the GRU-RNN outperforms other algorithms in all the evaluation metrics. The GRU-RNN yields good results for both legitimate and anomaly traffic traces, while other algorithms just work well in only one class.

The results in Table 6 show that our approach outperforms other methods in terms of accuracy. The DNN, coming in second place, shows the potential of the DL approach in anomaly detection. The VanilaRNN gives the worst result compared with its counterpart GRU-RNN.

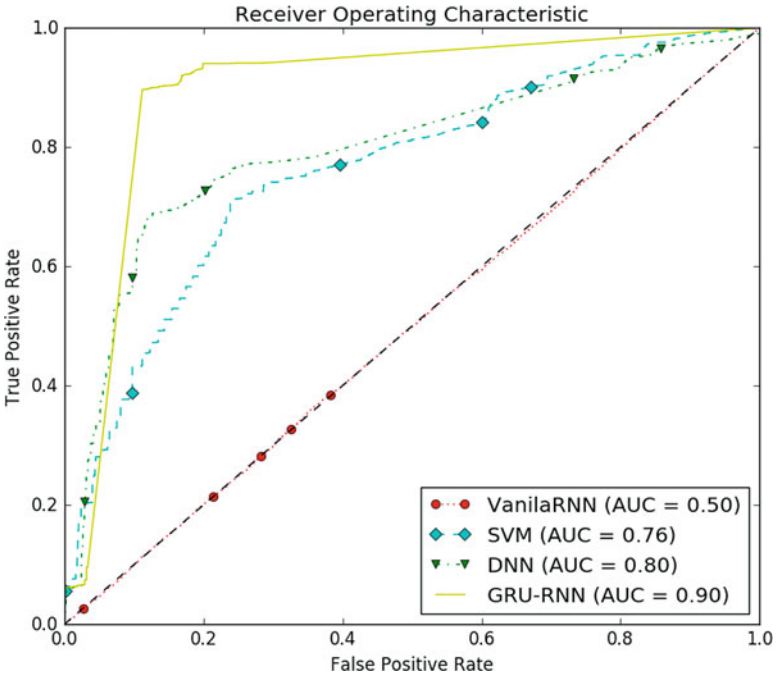


Fig. 4 ROC curve comparison for different algorithms

The Receiver Operating Characteristic (ROC) curve is also presented to evaluate our proposed approach. The ROC curve is created by plotting the false positive rate versus the true positive rate. The area under the ROC curve (AUC) is used to determine which classifier predicts the classes best. The higher the AUC, then the better is the classifier. Figure 4 shows that the proposed GRU-RNN achieves the highest AUC amongst all the tested algorithms with a True Positive Rate of 90% and a False Alarm Rate of 10%. The VanilaRNN gives the worst result as expected. As we can see, the GRU-RNN has a lowest False Positive Rate which is an important factor of the IDS.

The Precision vs Recall curve shows the trade off between Precision and Recall for different thresholds. A high area under the curve represents both high recall and high precision. An ideal system with a high area under the curve will return many results, with all results labelled correctly. As seen in Fig. 5, the GRU-RNN gives us the best results amongst all the algorithms. As the Recall threshold increases, the Precision decreases significantly for all the algorithms, except for GRU-RNN where increases Precision increases to 89%.

Furthermore, we also compared the performance of our proposed model with others in the literature. Our GRU-RNN is compared with other state-of-the-art algorithms like SVM, DNN and NB Tree algorithms. The NB Tree gave the best result in [30]. The results in Table 7 show that our proposed model outperforms all

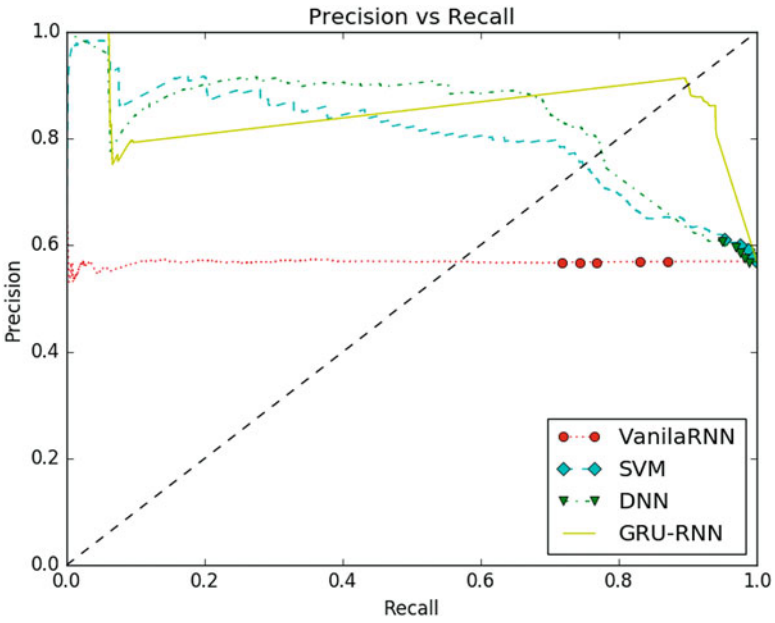


Fig. 5 Precision vs recall curves

Table 7 Accuracy comparison with previous studies

Method	Accuracy
SVM [30]	69.52%
DNN [29]	75.75%
NB tree [30]	82.02%
GRU-RNN (Proposed Model)	**89%**

Table 8 Performance metric evaluation for the CICIDS2017 dataset

Method	Precision	Recall	F1-measure
DNN	97%	97%	97%
ID3 [27]	98%	98%	98%
GRU-RNN (Proposed Model)	**99%**	**99%**	**99%**

the previous methods. Our GRU-RNN performs better than the SVM and NB Tree algorithms that use the whole set of 41 features for training and testing. The GRU-RNN result also indicates a significant improvement in accuracy compared to the basic DNN in our previous work.

For further investigation, we evaluate the GRU-DNN performance as regards detecting DDoS attacks in the CICIDS2017 dataset. We compare the proposed GRU-DNN with DNN and ID3 algorithms. Results of ID3 algorithm are the best achieved from the CICIDS2017 dataset in [27]. Table 8 gives details of our evaluation. As can be seen, the proposed GRU-DNN has better results in all the evaluation metrics compared with the best result from [27]. The DNN yields slightly

lower results than that of the ID3. The proposed GRU-DNN can work well with diverse and complex traffic traces and detect almost all types of DDoS attacks in the CICIDS2017 dataset.

From the above results, the GRU-DNN shows its strong potential in dealing with low-dimensional and raw traffic. Therefore, it is a potential solution for intrusion detection in the SDN paradigm.

4.5 Network Performance Evaluation

In this section, we evaluate the effect of our proposed GRU-RNN on the performance of the POX controller in the SDN environment.

4.5.1 Experiment Setup

The GRU-RNN is implemented as an application written in Python language in a POX [25] controller. Cbench [4] is a standard tool used for evaluating the SDN controller performance. Cbench runs in two modes: *throughput and latency modes*.

- *The Throughput Mode*: a stream of packet-in messages is sent to the controller for a specified period of time to compute the maximum number of packets handled by the controller.
- *The Latency Mode*: a packet-in message is sent to the controller and then waits for the reply to compute the time needed to process a single flow by the controller.

We ran our experiments on a virtual machine having an Intel Core i5-4460 3.2GHz with 3 cores available and 8GB of RAM. The operating system is Ubuntu 14.04 LTS-64bit. The controller performance is tested with a different number of virtual OpenFlow switches emulated by Cbench. The performance of the POX controller running stand-alone is considered as a baseline for our evaluation. We also compare the proposed GRU-RNN algorithm with the DNN algorithm in our previous work [29].

4.5.2 Experiment Results

Throughput evaluation indicates the performance of the controller under heavy traffic conditions. Figure 6 depicts the average response rate of the controller under three testing scenarios. As we can see, both the DNN and GRU-RNN cause overhead on the controller. The DNN algorithm is simpler than the GRU-RNN, and so it gives a slightly better network performance than that of the GRU-RNN. However, the GRU-RNN outperforms the DNN in terms of the detection accuracy. The affect of the GRU-RNN on the controller performance is predictable. The

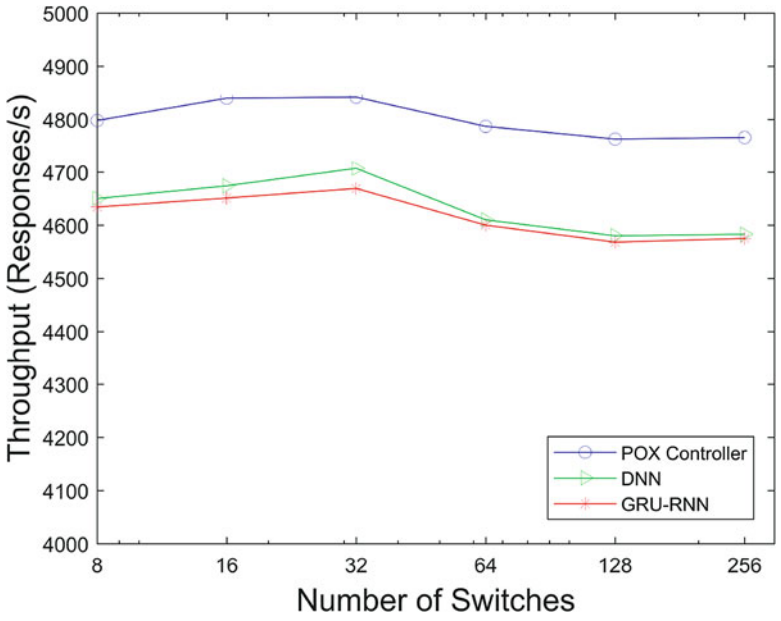


Fig. 6 Throughput evaluation

network throughput decreases slightly when the network size increases from 32 switches to 64 switches. The network performance degrades by about 3.5% when the network size is under 32 switches. When we increase the size to over 64 switches, the throughput drops by about 4%. The overhead on the controller of the GRU-RNN module is unavoidable. The GRU-RNN module has to send several *ofp_flow_stats_request* messages and process *ofp_flow_stats_reply* messages while processing packet-in messages. However, the throughput degradation is quite low and can be improved in the future.

Latency evaluation indicates the length of time that the controller takes to process one single packet. As we can see in Fig. 7, the network latency increases along with increasing the network size. When we increase the network size, the load on the controller is increased as well and this causes the overhead. The GRU-RNN still has the highest overhead amongst all. It takes time for the GRU-RNN to process the *ofp_flow_stats_reply* messages, *packet-in* messages and detect anomaly flows, so the overhead is unavoidable. The overall degradation is about 7% in all cases. This overhead is not significant and can be improved in the future.

All in all, the overhead caused by the GRU-RNN on the SDN controller is quite low, and so our proposed approach has significant potential for real-time intrusion detection in the SDN paradigm. So there is a trade off between performance and security. However, the network performance still can be improved in the future.

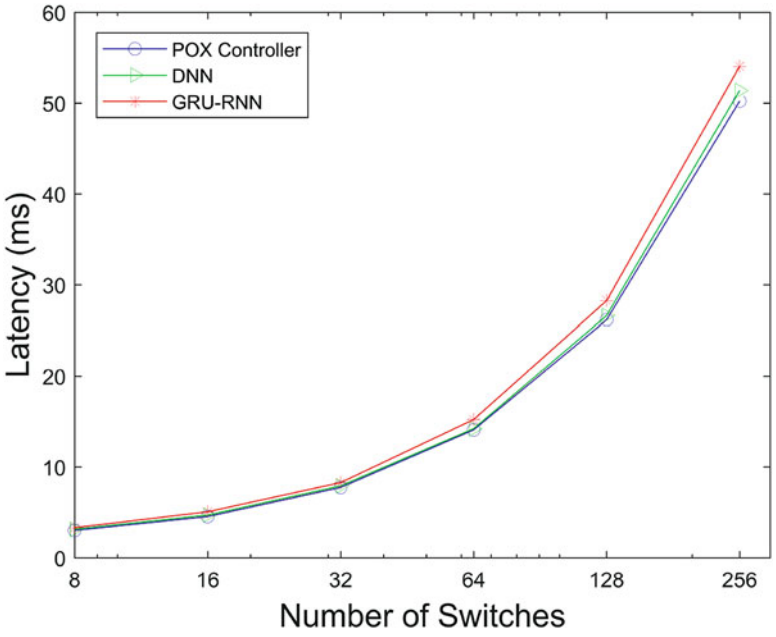


Fig. 7 Latency evaluation

5 Conclusion

This paper has made an attempt to briefly introduce the SDN architecture and its security issues. DL has great potential to be the key technology for intrusion detection in SDN. Despite the recent wave of success of DL in computer vision and language processing areas, there is a scarcity of DL applications in solving SDN security issues. In this chapter, we present an Anomaly-based IDS in the SDN environments using the GRU-RNN algorithm. We show that our proposed approach outperforms other state-of-the-art algorithms with an accuracy of 89% and 99% for the NSL-KDD and CICIDS2017 datasets. Our scheme uses a minimum number of features compared to other state-of-the-art approaches so computational costs can be reduced significantly. In addition, the network performance evaluation showed that our proposed approach does not significantly affect the controller performance. This makes our model a strong candidate for real-time detection. In the future, we will optimize our model and use other features to increase the accuracy and reduce the overhead on the controller. We will also try to extend our research to unsupervised intrusion detection approaches.

References

1. AlEroud A, Alsmadi I (2017) Identifying cyber-attacks on software defined networks: an inference-based intrusion detection approach. J Netw Comput Appl 80:152–164
2. Bang JH, Cho YJ, Kang K (2017) Anomaly detection of network-initiated LTE signaling traffic in wireless sensor and actuator networks based on a hidden semi-markov model. Comput Secur 65:108–120
3. Braga R, Mota E, Passito A (2010) Lightweight DDoS flooding attack detection using NOX/openflow. In: Local Computer Networks (LCN), 2010 IEEE 35th Conference on, IEEE, pp 408–415
4. Cbench (2009) https://github.com/mininet/oflops/tree/master/cbench. Accessed 04 Jul 2018
5. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078
6. Chollet F (2015) Keras. https://github.com/fchollet/keras. Accessed 04 Jul 2018
7. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555
8. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
9. Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J (2001) Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: Kremer SC, Kolen JF (eds) A field guide to dynamical recurrent neural networks. IEEE Press, New York
10. Ikram ST, Cherukuri AK (2016) Improving accuracy of intrusion detection model using PCA and optimized SVM. J Comput Inf Technol 24(2):133–148
11. KDDCup99 (1999) http://kdd.ics.uci.edu/databases/kddcup99/. Accessed 04 Jul 2018
12. Kokila R, Selvi ST, Govindarajan K (2014) DDoS detection and analysis in SDN-based environment using support vector machine classifier. In: Advanced Computing (ICoAC), 2014 Sixth International Conference on, IEEE, pp 205–210
13. Kreutz D, Ramos F, Verissimo P (2013) Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp 55–60
14. Kreutz D, Ramos FM, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S (2015) Software-defined networking: a comprehensive survey. Proc IEEE 103(1):14–76
15. McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by Lincoln laboratory. ACM Trans Inf Syst Secur 3(4):262–294. https://doi.org/10.1145/382912.382923
16. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008) Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 38(2):69–74
17. Mehdi SA, Khalid J, Khayam SA (2011) Revisiting traffic anomaly detection using software defined networking. In: International Workshop on Recent Advances in Intrusion Detection, Springer, pp 161–180
18. Mousavi SM, St-Hilaire M (2015) Early detection of DDoS attacks against SDN controllers. In: Computing, Networking and Communications (ICNC), 2015 International Conference on, IEEE, pp 77–81
19. Nie L, Jiang D, Lv Z (2017) Modeling network traffic for traffic matrix estimation and anomaly detection based on Bayesian network in cloud computing networks. Ann Telecommun 72(5–6): 297–305
20. Niyaz Q, Sun W, Javaid AY (2016) A deep learning based DDOS detection system in software-defined networking (SDN). arXiv preprint arXiv:161107400
21. ONF (n.d) Software-defined networking (SDN) definition. https://www.opennetworking.org/sdn-definition/. Accessed 12 Feb 2018
22. Parwez MS, Rawat D, Garuba M (2017) Big data analytics for user activity analysis and user anomaly detection in mobile wireless network. IEEE Trans Ind Inf 13:2058–2065

23. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
24. Phan TV, Van Toan T, Van Tuyen D, Huong TT, Thanh NH (2016) Openflowsia: an optimized protection scheme for software-defined networks from flooding attacks. In: Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on, IEEE, pp 13–18
25. POX (2009) https://github.com/noxrepo/pox. Accessed 04 Jul 2018
26. Scott-Hayward S, Natarajan S, Sezer S (2016) A survey of security in software defined networks. IEEE Commun Surv Tutorials 18(1):623–654
27. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of Fourth International Conference on Information Systems Security and Privacy, ICISSP
28. Sommer R, Paxson V (2010) Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy, IEEE, pp 305–316
29. Tang T, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M (2016) Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (WINCOM'16), Fez
30. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications
31. Winter P, Hermann E, Zeilinger M (2011) Inductive intrusion detection in flow-based network data using one-class support vector machines. In: New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on, IEEE, pp 1–5

SeqDroid: Obfuscated Android Malware Detection Using Stacked Convolutional and Recurrent Neural Networks

William Younghoo Lee, Joshua Saxe, and Richard Harang

Abstract To evade detection, attackers usually obfuscate malicious Android applications. These malicious applications often have randomly generated application IDs or package names, and they are also often signed with randomly created certificates. Conventional machine learning models for detecting such malware are neither robust enough nor scalable to the volume of Android applications that are being produced on a daily basis. Recurrent neural networks (RNN) and convolutional neural networks (CNN) have been applied to identify malware by learning patterns in sequence data. We propose a novel malware classification method for malicious Android applications using stacked RNNs and CNNs so that our model learns the generalized correlation between obfuscated string patterns from an application's package name and the certificate owner name. The model extracts machine learning features using embedding and gated recurrent units (GRU), and an additional CNN unit further optimizes the feature extraction process. Our experiments demonstrate that our approach outperforms Ngram-based models and that our feature extraction method is robust to obfuscation and sufficiently lightweight for Android devices.

Keywords Android · Malware classification · Convolutional neural network · Recurrent neural network

1 Introduction

This section introduces machine learning approaches for detecting malicious Android applications. Android applications are distributed as APK files, and an APK file consists of various components, including a manifest, a certificate, Java code, native code and resource files. Researchers have proposed various machine

W. Y. Lee (✉) · J. Saxe · R. Harang
Sophos Group PLC, Abingdon, UK
e-mail: william.lee@sophos.com; joshua.saxe@sophos.com; richard.harang@sophos.com

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_9

learning features for APK files [1, 2]; however handcrafted features require domain knowledge, and complex features increase computational costs in both training and inference phases. For example, extracting API sequences or opcode sequences from a DEX file requires significant computational resources [3]. Obfuscated or encrypted malware makes it infeasible to collect features used in the previous work [3].

On the other hand, an application's meta information, such as package names and certificate data, can be collected without requiring any high-cost analysis of APK code. In addition, the application's capability information, such as requested permissions and intent actions can be obtained easily, but the metadata contains the application's key characteristics [4]. Android's manifest files provide the app's package name, permissions and intent actions, and APK's certificate file describes developer information [5]. The information is always available from metadata even if the APK's code is either encrypted or obfuscated.

Another advantage of using package and certificate strings as input features is that they provide interpretability. While deep learning models outperform simple linear models, they are notoriously difficult to explain due to their complexity [6]. The short string data itself can indicate whether a particular application looks benign or malicious. The input data is intuitive and simple, but our method offers efficient and automated feature representation directly from the raw input strings. Our feature extraction method for these strings is not only lightweight but also improves classification performance relative to Ngram-based models.

We discuss the characteristics of Android malware and related work in the following sections. Then, we present a novel malware classification model using multiple short strings from Android's metadata. We also provide an analysis of our experimental results.

2 Android Malware

Millions of new Android applications are released in app stores every year, and threat researchers regularly discover malicious apps in app stores [7, 8]. As a security measure, on the Android platform, each app is required to have a unique ID, a package name and a signature with a developer's certificate [9]. This mechanism ensures that only the applications from the same author can be upgraded on Android devices. It is also not possible to install an unsigned APK. The security policy drives malware writers to generate numerous malware variants using random package names and random certificates.

Together, the package name and certificate owner name provide a unique application ID on the Android platform. Different versions of an application usually contain distinctive code to support new features. However, they share a unique application ID and also use the same permissions and intent actions [10]. Malware variants from a family also exhibit the same characteristics [31]. Their malicious code is frequently obfuscated, but invariant components are app's capability information and the string patterns in the app ID.

Table 1 Applications' package and certificate information

App	Package	Certificate owner
Facebook	com.facebook.katana	C=US, ST = CA, L = Palo Alto, O=Facebook Mobile, OU=Facebook, CN=Facebook Corporation
Instagram	com.instagram.android	C=US, ST = California, L = San Francisco, O=Instagram Inc., CN=Kevin Systrom
malware1	awdawdad.qweqweqwe.awdawdad.werwefwf	C = DE, ST = esttttttttttt, L = htfffffffffffffff, O = drrrrrrrrrrrrrrrrrrr, OU = htttttttttttttttttt, CN = egggggggggggggggh
malware2	com.kbvivk.eavkxrip	C = ha, CN = dwpovc, L = ap, O = dkteeu, ST = mq, OU = wzfhlq

Table 2 Malware apps' package and certificate data

Package	Certificate owner
com.hefcfn.vzdorreb	CN = mtrnuk, OU = sxjkvv, O = iohbaa, L = oe, ST = wo, C = yk
com.ayhvsx.dlnlawpk	CN = zmonlx, OU = bfqpmh, O = sbzknd, L = zh, ST = uv, C = gs
com.uzhwbs.nltzmnun	CN = lgkjju, OU = qoqbyu, O = yqwair, L = gi, ST = cy, C = sk
com.kbvivk.eavkxrip	OU = wzfhlq, O = dkteeu, L = ap, ST = mq, C = ha

While users recognize an application by an app name and icon, the Android platform identifies an application by package name and certificate information. For instance, the Facebook app's name is Facebook, but its package name and certificate data contain additional information. Apps' names can be duplicated, but package names should remain unique in Android.

Table 1 lists package and certificate owner information from some different applications.

Both legitimate apps (Facebook and Instagram) have appropriate names for the package and certificate owner. On the other hand, the malware samples clearly indicate that they have meaningless strings in the package and certificate owner names.

The package name follows a reverse domain form to represent a unique ID. For example, com.facebook.katana is the package name of the Android Facebook app. The Facebook app's certificate owner name contains proper information, including geolocation and a common name for the company. However, the third and fourth ones have suspicious strings. They are malicious applications, and there are thousands of similar ones with similar string patterns.

Table 2 lists the package and certificate information of obfuscated malware variants. The malicious apps do not share any common strings, but request the same permissions and intent actions to do the same malicious activities when they are running. While the package name and certificate are entirely randomized, they often share common patterns in random strings. For example, there are more than thousands of malware variants using the same random patterns.

The variants declare the following permissions and intent actions in the manifest to intercept SMS data from victims' devices, and many malicious samples do the

same [11]. Thus, features extracted from permissions and intent actions are effective indicators for malware detection.

Permissions

- android.permission.SEND_SMS
- android.permission.READ_SMS
- android.permission.WRITE_SMS
- android.permission.GET_TASKS
- android.permission.SYSTEM_ALERT_WINDOW

Intent actions

- android.provider.Telephony.SMS_RECEIVED
- android.intent.action.SIM_STATE_CHANGED
- android.intent.action.SCREEN_ON
- android.intent.action.USER_PRESENT
- android.intent.action.BOOT_COMPLETED

It is not a trivial task to convert the sequential data into useful machine learning features. One common approach is to apply Ngrams against the string data. When the hashing trick is combined with Ngrams, feature extraction from the string data does not require the system to maintain the full vocabulary of Ngrams and thus reduces feature dimensionality significantly. Recurrent neural networks (RNNs), in which the characters are often embedded and the sequence of embedded characters reduced into a single vector, are also well known and widely used to recognize patterns in sequence data.

Our motivation for using package and certificate data as inputs is they are simple but easy to interpret for validation. Learning latent features from the randomized string patterns makes a classification model more robust to detect similar variants. When additional capability information, such as permissions and intent actions, is efficiently combined, the model's classification performance can be improved. Because Android permissions and intent actions unveil apps' behaviors without monitoring app's runtime activities, they play an important role as one of the most discriminating characteristics for malware detection. There are platform-standard and custom permissions and intent actions, so the number of items is unknown. The bag-of-words model is a simple approach to convert string items into a fixed-size vector, but the vector's dimensionality can be determined only after exploring the full data set. The dimensionality is often quite large for random strings. As a realistic alternative, we used the hashing trick for permissions and intent actions. The short strings were hashed, and the indexes were used to increase the corresponding frequency value. Hashing collisions were mitigated with the use of frequency counts instead of binary values.

3 Related Work

This section outlines related machine learning approaches in malware classification.

3.1 Malware Classification

Several machine learning approaches have been proposed to detect malicious Android applications, and they fall into two general categories with respect to feature extraction. Static analysis methods [4, 10, 12], extract features such as permissions, APIs from an application's code or metadata. The work of [13] uses the statistics of APIs as malware features. Some of these features, especially when applied to code, are not generally robust to obfuscated variants. Dynamic analysis methods collect behavioral events at runtime and transform them into features. Dynamic analysis approaches [14, 15], use platform APIs, system calls, file and network access information as input features. These behavioral methods are more resilient to obfuscation. However, dynamic methods are not scalable to address the rapidly growing number of applications. Another weakness is that many attempt to collect malicious activities within a limited time, which can often be trivially avoided [16, 17].

In the past, researchers applied conventional machine learning models with either static or dynamic features. Naive Bayes and random forest models are applied with APIs and metadata [18, 19]. Recently, researchers have demonstrated that deep learning models outperform conventional machine learning models [6]. While classical methods require handcrafted features from domain experts, deep learning makes it possible to discover automated feature extraction from a large dataset [20].

3.2 CNNs

Convolutional neural network (CNN)-based models demonstrate state-of-the-art performance in image recognition problems. The main benefit of CNNs is that feature extraction is automated by applying convolutional and pooling operations on a raw image, which eliminates handcrafted feature engineering processes. In addition, CNNs have been applied to malware classification [3, 6]. In static analysis, some subset of the file is fed into a CNN via an embedding approach. Typically, dynamic analysis captures runtime events, such as API calls, and the sequence data is fed into a CNN.

The work of [3] uses an API's sequence as an input to a CNN layer. The model automatically extracts and learns malicious patterns using raw method calls in the assembly code. While the model can be deployed on various devices, from workstations to IoT devices, their feature extraction cost is more expensive than our lightweight method. The work of [6] improved classification accuracy using deep auto-encoder and CNN layers, but the approach requires a pre-training step using an auto-encoder to extract features. Aside from Android malware problems, [20] proposed a CNN-based model, eXpose, to detect malicious URLs; the model takes raw URLs as input and learns to extract features and classify them. Inspired by the eXpose, our model also extracts features from raw short strings in an app's metadata.

Unlike previous work, which operated on the single input string, we propose a novel approach to leverage multiple input strings and to extract latent patterns in combined data sources using both CNN and RNN layers.

3.3 RNNs

Recurrent neural networks (RNNs) use hidden units to maintain sequential information and learn to classify string data, such as a sequence of characters. Moreover, RNNs are powerful dynamic systems, but they have the problems of vanishing gradients and computational costs. Long short-term memory (LSTM) networks [21] are introduced to solve the issues, and gated recurrent units (GRUs) [22] further reduce training costs. Embeddings are designed to represent the text data's semantic relationship as dense vectors. Character-level embeddings transform short strings into dense vectors, making individual characters' vector representations more relevant to their semantic relationships [20].

Rhodea et al. [16] collected behavioral data for the first few seconds of file execution and used RNN to classify malware, and achieving an accuracy of 0.96. Their assumption was that malicious activity begins rapidly once a malicious file begins execution. However, malicious applications usually wait for certain system events to perform hidden operations. [23] proposed using LSTM with a sequence of Android's permissions, obtaining a classification accuracy of 0.897.

Unlike the previous work, our model operates directly from multiple raw strings and learns both string patterns and relationships between inputs.

4 Methodology

End-to-end machine learning models involve two components, feature extraction and classification, where we jointly extract features and train a model in a single training step. This section describes our feature extraction method and the architecture of neural networks for feature extraction and classification.

Two feature extraction methods were explored to compare classification results. A baseline model uses Ngrams features from package and certificate owner strings, whereas, in our model, the features were automatically generated from an RNN feature extraction layer.

4.1 Ngrams Features

Character-level Ngrams feature extraction is a pragmatic method for short text data. Both the package name and the certificate owner name are short strings, and the

average length of the strings is about 20 characters. As Ngrams features weakly preserve the sequence order in strings, they are known to capture sequence patterns better than orderless bag-of-words or bag-of-tokens methods. Character-level (1,5)-gram features were extracted from each input string data, and then the hashing trick was used to generate fixed size feature vectors.

We selected a hashing bucket size of 128 in order to match the RNN's feature dimensionality. Hashing collisions were handled by increasing the previously taken value of the hashing bucket. We also used feature hashing to extract string features from permissions and intent actions.

4.2 RNN Features

Recurrent neural network (RNN) features are extracted through an embedding and RNN feature extraction unit. From an input string, the extractor produces a 128-dimensional vector, and the vector is sent to the subsequent classification layer. Since RNNs accept time series or sequential inputs, it is necessary to convert input strings into sequential vectors. Embedding is designed to embed input strings into fixed length dense vectors. A raw string is a sequence of characters, and the unit embeds them into a shape of (S, D) real-valued matrix. Each character is mapped to its vector, and those vectors are concatenated into a matrix as the input for the next RNN layer. The embedding layer is optimized during training phase along with other model's parameters resulting in pairs of semantically similar characters being embedded closer to each other [20].

We used a modified method used in the eXpose, with a string length of $S = 40$ and an embedding depth of $D = 32$ for our experiments [20]. For package and certificate owner information, 87 printable characters were selected as valid input characters, and any other characters are replaced with a reserved token. The maximum length of the string was set to 40, covering 95% of the samples. If the input string was less than 40, padding characters were added; otherwise the first 40 characters were selected.

Figure 1 presents an overview of our feature extraction component. The RNN unit received sequential data from the embedding unit and outputs a 128-dimensional vector. The vector was fed to the next classification step. A gated recurrent unit (GRU) [22] was chosen for our RNN unit to improve training speeds without compromising the RNN's classification performance.

Convolutional neural networks are well known for extracting features for image recognition, and one-dimensional CNNs learn to extract sequential patterns from string data. Adding a one-dimensional CNN unit prior to the RNN unit reduces the dimensionality of the vectors; thus, it decreases the training time for the next RNN unit. The CNN's max pooling compresses a vector's dimensionality in half, to (20, 32) from (40, 32), without losing any latent information.

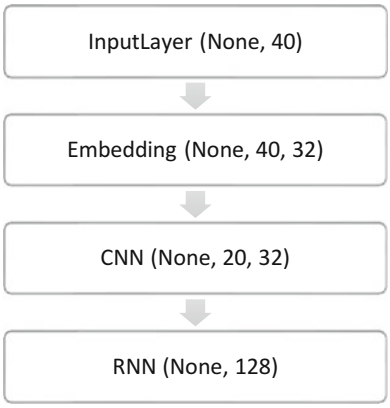


Fig. 1 Feature extraction component

4.3 Classification

Our classification component took any fixed-size vectors from the feature extraction component. When multiple inputs were used, the inputs were concatenated into a vector as the input for the next classification component. The component consisted of three dense layers, as depicted in Fig. 2. Each dense layer comprised of a batch normalization [24] and a dropout [25] unit to improve classification performance and prevent overfitting.

Experimental models were implemented in Keras [26] and tested on a machine with one Nvidia Telsa GPU. An Adam optimizer [27] with a learning rate of 0.001 was applied to optimize the following binary cross entropy loss in all models:

$$-(ylog(p) + (1-y)\log(1-p))$$

where p is the predicted probability of observation and y is a binary indicator of the label data.

5 Experiments and Results

Two million APK samples were collected from VirusTotal [28], and the first seen dates used were between January and December 2017. We randomly selected 20% of the samples are used for validation in our tests as shown in Table 3. Our labeling strategy was to use consensus voting from detection information in VirusTotal. Samples detected by no scanners were labeled as benign, and samples detected by five or more scanners were marked as malicious.

We evaluated the following five experiments by training models for 100 epochs. Each experiment was configured to test the different features in Table 4.

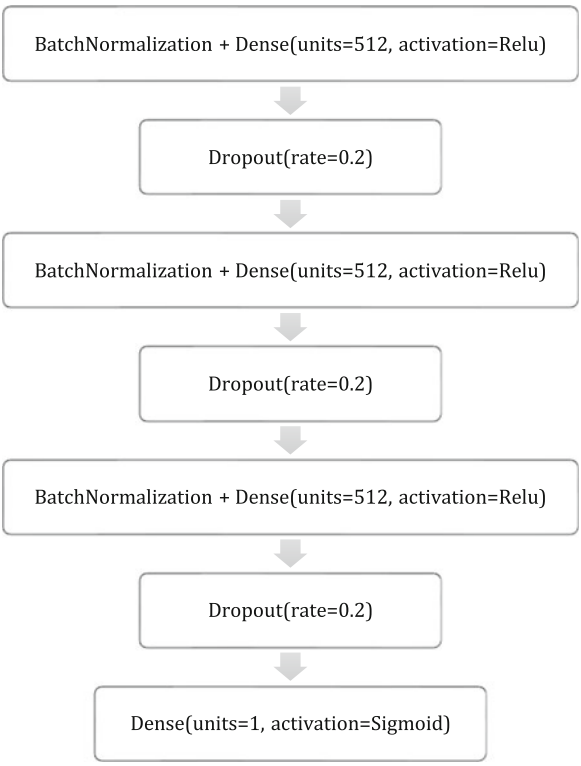


Fig. 2 Classification component

Table 3 Sample set

	Benign samples	Malicious samples
Training set	888,620	1,057,091
Validation set	264,130	222,298

Table 4 Inputs for five experiments

Experiment	Input and feature dimension
Ngram	Package (128)Certificate (128)
RNN	Package (128)Certificate (128)
Ngram-PA	Package (128)Certificate (128)Permissions + intent actions (256)
RNN-PA	Package (128)Certificate (128)Permissions+ intent actions (256)
CNN-RNN-PA	Package (128)Certificate (128)Permissions + intent actions (256)

Multiple input feature vectors were all concatenated into a vector, and the vector was fed into the next classification component. The first baseline model, Ngram received two 128-dimensional vectors from package and certificate owner strings and then concatenated them into 256-dimensional vectors for classification. The second RNN model used RNN features extracted from the raw package and certificate inputs. Ngram-PA and RNN-PA models added additional features from permissions and intent actions, resulting in 512-dimensional vectors. The last one,

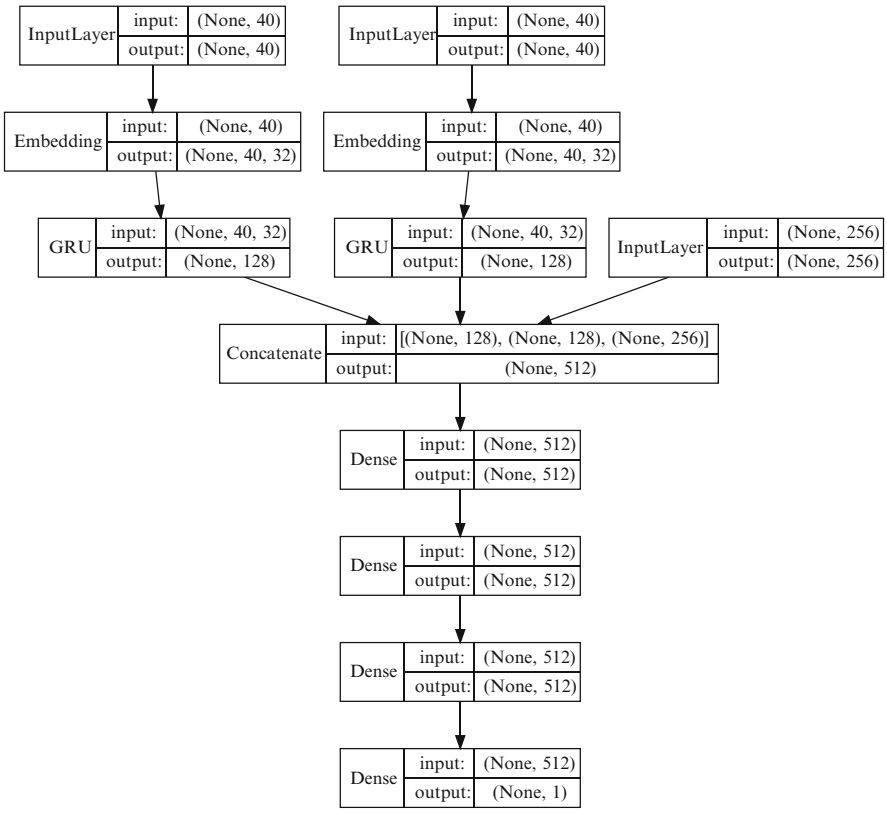


Fig. 3 The architecture of CNN-RNN model

CNN-RNN-PA added an additional CNN unit in the RNN-PA model. Figure 3 depicts the network architecture of CNN-RNN-PA model and the model's batch normalization and activation units are omitted in the diagram. The left and middle inputs are for the package and certificate features and the right input is for the permission and intent action features.

Figure 4 displays the results in receiver operating characteristic (ROC) curves, and Table 5 describes each model's true positive rate (TPR) and false positive rate (FPR). We compared the model's performance at a lower FPR of 0.001. While the Ngram model yielded a TPR of 0.805 at FPR of 0.001, the RNN model achieved a TPR of 0.869, resulting in a gain of 8%. The Ngram-PA dropped its TPR to 0.717, which suggests that simply adding additional input data does not guarantee any gains. However, the RNN-PA and the CNN-RNN-PA improved their TPRs to 0.929 and 0.936, respectively, resulting in a 16% gain against the Ngram model.

Although we did not perform an exhaustive analysis of malware samples, the obfuscated samples in Table 2 were detected by the CNN-RNN-PA model with a highly competent prediction score of 0.999. The number of obfuscated malware has

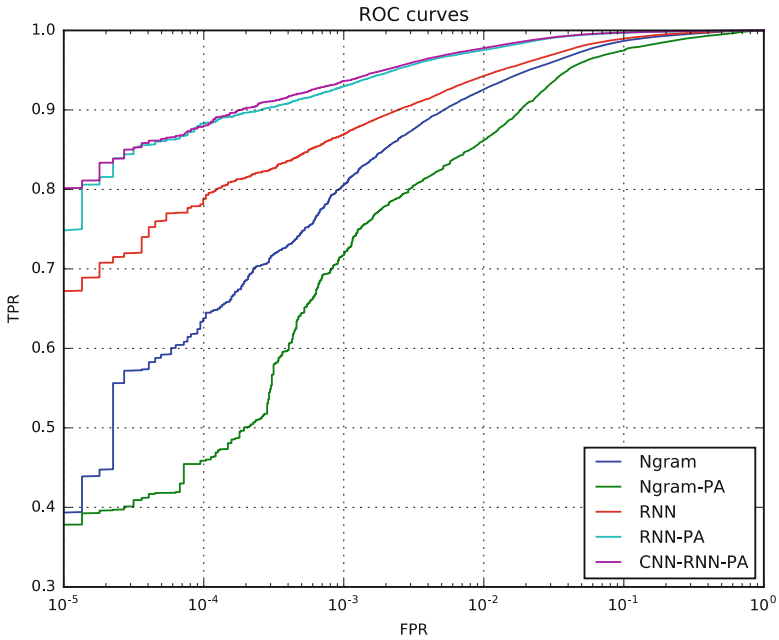


Fig. 4 ROC curves

Table 5 Results in TPR and FPR

	AUC	TPR		
		FPR(0.0001)	FPR(0.001)	FPR(0.01)
Ngram	0.9939	0.637	0.805	0.925
RNN	0.9955	0.788	0.869	0.942
Ngram-PA	0.9882	0.458	0.717	0.862
RNN-PA	0.9985	**0.883**	0.929	0.975
CNN-RNN-PA	**0.9986**	0.880	**0.936**	**0.977**

been increasing, accounting for at least 50% of all malware. The CNN-RNN-PA model achieved an area under curve (AUC) of 0.9986, which strongly supports the idea that the model can classify both obfuscated and non-obfuscated malware with a low FPRs.

We also analyzed the complexity of models by measuring their trainable parameter sizes and training costs in Table 6. As a model's complexity increases, the training time also increases. For instance, the RNN-PA model's complexity increased by 40% in parameter size and 11 times in training time when compared to the Ngram model. On the other hand, the best performing model combined with CNN and RNN reduced its training time by 55% and also achieved the best AUC of 0.9986 with a TPR of 0.936. The CNN layer efficiently compressed the representation of features by 50% by using a max polling unit, but the compressed vectors still preserved latent patterns from raw input strings.

Table 6 Models' complexity

	Number of parameters	Training time per epoch (seconds)
Ngram	663,533	69
RNN	793,793	790
Ngram-PA	794,625	71
RNN-PA	925,889	808
CNN-RNN-PA	932,097	446

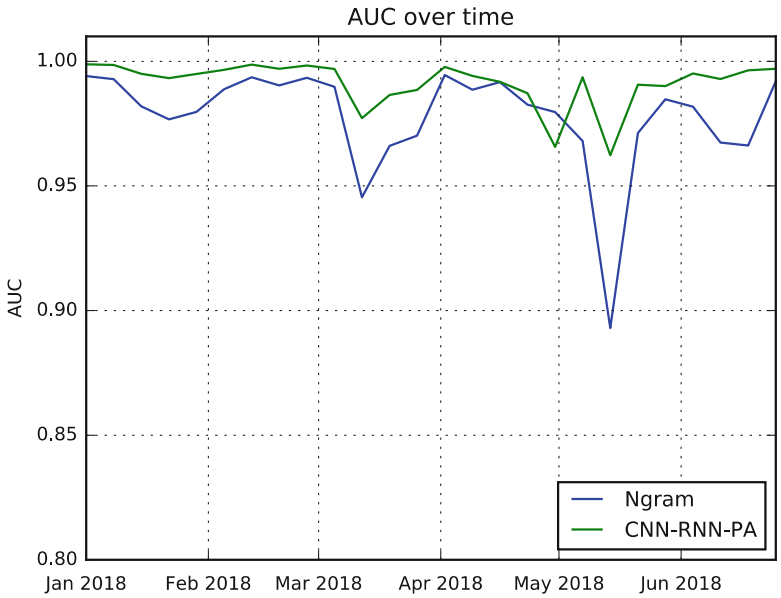


Fig. 5 AUC over time

In order to evaluate the robustness of the models, AUCs were monitored for over 6 months against approximately one million new samples from 2018. Figure 5 displays the time decay results of two models by measuring weekly AUC over time. It is not uncommon that the classification power of machine learning models degrades over time. While there were drops in March and May, the AUC of the RNN-CNN-PA model outperformed the Ngram model and its AUC is increased in June. The time decay results suggest that our models learn to detect unseen malware efficiently.

The runtime performance of the RNN-CNN-PA model was tested on a Nexus-5 device. For the experiment, the Keras model's weights were converted into the TensorFlow format, and TensorFlow mobile version was used for the test. The file size of the model's weights was about 3.8 MB, and the average prediction took about 40 msec for an APK file, which supports the idea that our method is a lightweight and efficient solution for the Android platform.

6 Conclusion

This chapter discussed the characteristics of Android malware and machine learning models for detecting them using different features. The rapidly growing number of obfuscated malware makes it difficult to develop a scalable and robust machine learning model.

Our models jointly learned to extract features from multiple short strings and identify malicious Android applications. The feature representation from our stacked CNN and RNN units improved the model's classification ability to identify obfuscated malware. We also presented how to efficiently combine feature vectors from Android's metadata such as package names, developer names and capability information, in order to optimize learning performance. Our CNN-RNN method improved classification performance by 16% against Ngram features and reduced training time by 50% against an RNN model without any compromise of classification performance. Furthermore, our model proved to be robust to obfuscation and lightweight enough to be deployed on Android devices.

While we explored a concatenation method as fusing multiple features, future work should consider the potential effects of different feature fusion methods. For example, a multiplicative fusion method can be used for combining features from multiple sources [29].

References

1. Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K (2014) Drebin: effective and explainable detection of Android malware in your pocket. In: Network and distributed system security symposium
2. Zhou Y, Jiang X (2016) Dissecting android malware: characterization and evolution. In: Proceedings of the 2012 IEEE symposium on security and privacy
3. Billah Karbab E, Debbabi M, Derhab A, Mouheb D, Android malware detection using deep learning on API method sequences. arXiv preprint, arXiv:1712.08996
4. Peiravian N, Zhu X (2013) Machine learning for android malware detection using permission and API calls. Int Conf Tools Artif Intell:300–305
5. Chen W, Aspinall D, Gordon A, Sutton C, Muttik I (2016) More semantics more robust: improving android malware classifiers. In: Proceedings of the 9th ACM conference on security & privacy in wireless and mobile networks, pp 18–20
6. WangEmail W, Wang Z (2018) Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. J Ambient Intell Humaniz Comput: 1–19
7. Android malware. https://nakedsecurity.sophos.com/2017/11/07/2018-malware-forecast-the-onward-march-of-android-malware/
8. Andriod PHA. https://source.android.com/security/reports/Google_Android_Security_PHA_classifications.pdf
9. Android certificate. https://developer.android.com/studio/publish/app-signing

10. Suarez-Tangil G, Kumar Dash S, Ahmadi M, Kinder J, Giacinto G, Cavallaro L (2017) DroidSieve: fast and accurate classification of obfuscated android malware. In: Proceedings of the seventh ACM on conference on data and application security and privacy. ACM, New York, pp 309–320
11. Android permissions. https://www.researchgate.net/publication/296704790_Permission_Analysis_for_Android_Malware_Detection
12. Aafer Y, Du W, Yin H (2013) Droidapiminer: mining api-level features for robust malware detection in android. In: International conference on security and privacy in communication systems, pp 86–103
13. Huda S, Abawajy J, Alazab M, Abdollalihian M, Islam R, Yearwood J (2016) Hybrids of support vector machine wrapper and filter based framework for malware detection. Futur Gener Comput Syst 55:376–390
14. Kumar Dash S, Suarez-Tangil G, Khan S (2016) DroidScribe: classifying android malware based on runtime behavior. In: IEEE security and privacy workshops. Conference Publishing Services, IEEE Computer Society, Los Alamitos, pp 252–261
15. Chaba S, Kumar R, Pant R, Dave M, Malware detection approach for android systems using system call logs. arXiv preprint, arXiv:1709.08805
16. Rhodea M, Burnapa P, Jonesb K, Early-stage malware prediction using recurrent neural networks. arXiv preprint, arXiv:1708.03513
17. Malik S, Khatter K (2016) System call analysis of android malware families. Indian J Sci Technol 9
18. Ahmed F, Hameed H, Shafiq MZ, Farooq M (2009) Using spatio-temporal information in API calls with machine learning algorithms for malware detection. In: Proceedings of the 2nd ACM workshop on security and artificial intelligence. ACM, New York, pp 55–62
19. Tian R, Islam R, Batten L, Versteeg S (2010) Differentiating malware from cleanware using behavioural analysis. In: International conference on malicious and unwanted software
20. Saxe J, Berlin K, eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint, arXiv:1702.08568
21. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
22. Cho K, Merrienboer Bv, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y, Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint, arXiv:1406.1078
23. Vinayakumar R, Soman KP, Poornachandran P (2017) Deep android malware detection and classification. In: International conference on advances in computing, communications and informatics
24. Ioffe S, Szegedy C, Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint, arXiv:1502.03167
25. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
26. Keras. https://keras.io/
27. Kingma DP, Ba J, Adam: a method for stochastic optimization. arXiv preprint, arXiv:1412.6980
28. Virustotal. https://www.virustotal.com/
29. Liu K, Li Y, Xu N, Learn to combine modalities in multimodal deep learning. arXiv preprint, arXiv:1805.11730
30. Rieck K, Trinius P, Willems C, Holz T (2011) Automatic analysis of malware behavior using machine learning. J Comp Secur 19:639–668
31. Alazab M (2015) Profiling and classifying the behaviour of malicious codes. J Syst Softw, Elsevier 100:91–102

Forensic Detection of Child Exploitation Material Using Deep Learning

Mofakharul Islam, Abdun Nur Mahmood, Paul Watters, and Mamoun Alazab

Abstract A precursor to successful automatic child exploitation material recognition is the ability to automatically identify pornography (largely solved) involving children (largely unsolved). Identifying children's faces in images previously labelled as pornographic can provide a solution. Automatic child face detection plays an important role in online environments by facilitating Law Enforcing Agencies (LEA) to track online child abuse, bullying, sexual assault, but also can be used to detect cybercriminals who are targeting children to groom up them with a view of molestation later. Previous studies have investigated this problem in an attempt to identify only children faces from a pool of adult faces, which aims to extract information from the basic low- and high-level features i.e., colour, texture, skin tone, shape, facial structures etc. on child and adult faces. Typically, this is a machine learning-based architecture that accomplish a categorization task with the aim of identifying a child face, given a set of child and adult faces using classification technique based on extracted features from the training images. In this paper, we present a deep learning methodology, where machine learns the features straight away from the training images without having any information provided by humans to identify children faces. Compared to the results published in a couple of recent work, our proposed approach yields the highest precision and recall, and overall accuracy in recognition.

Keywords Child exploitation · Child pornography · Digital forensics · Intelligence analysis · Deep learning · Cybercrime

M. Islam (✉) · A. N. Mahmood · P. Watters
Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia
e-mail: M.Islam3@latrobe.edu.au; A.Mahmood@latrobe.edu.au; P.Watters@latrobe.edu.au

M. Alazab
Charles Darwin University, Casuarina, NT, Australia
e-mail: mamoun.alazab@cdu.edu.au

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_10

1 Introduction

Categorical age estimation by machine offers substantial extent of challenges and difficulty to the computer vision community due to unavailability of complete knowledge on human visual system. Study revealed many factors include among others; individual's lifestyle, health, race, occupation, cultural background has a significant impact on human ageing process especially on face on top of people's gene. Over the last three decades, much interest have been paid adult face detection only but nothing significant has been contributed on its counterpart. An effective and efficient way that attempts to quantify children's face in ways that agree with human intuition needs to have at least some of the similar cues akin to human visual system in making judgment. Defining such cues certainly will put this research to a considerable amount of challenge. The issue is not clear how to extract features that have discriminative ability for age estimation using skin tone or other facial cues visible in naked eyes.

Deep learning, a sub-type of machine learning, is a new era where machine can learn many layers of perception and depiction to create a sense of data like text, audio, and image leading object detection problems in to the next level. Each layer in the network takes in data from the previous layer, transforms it, and passes it on. The network increases the complexity and detail of what it is learning from layer to layer. Here the network learns straight away from the data—no influence from human at all over what salient features are being learned by the network. Deep learning is a data-hungry scheme requires a huge volume of data to produce accurate result while machine learning accuracy is almost plateaus – reach a state of little or no change.

2 Related Work

Kwon and Lobo [1] initially employed high resolution facial images to categorise facial images into three age groups – senior adults, younger adults or child. Application on a limited image dataset not allowing this to consider as a robust approach in categorical age estimation though authors claimed 100% accuracy. Later, they suggested another classification technique to accomplish the categorical age estimation where they employed a craniofacial development theory and wrinkle analysis [2]. Once more their approach is disadvantaged by inadequate image, which may result in poor performance while applied on large real-world datasets. The wrinkle analysis employed again by Hayashi et al. [3] combined with geometric relationships between visual structures on a human face for categorical age estimation. Their approach is primarily aimed at classifying age into multiple groups at the 5 years intervals which in fact, is an age-group wise classification technique – not exactly a categorical age estimation technique.

Lanitis et al. [4], employed Active Appearance Models (AAM) [5] which combined shape and texture parameters to extract using three classifiers – simple quadratic fitting, shortest distance, and Neural Network classifier to estimate the age. Finally, the age estimation accuracies of these classifiers are compared. An algorithm based on uncertainties of the age labels is suggested by Yan et al. [6] to categorise age-group. An approach combining AAM with Support Vector Machine Regression is suggested by Khoa et al. [7] for categorical age estimation. AAM's parameters are employed initially to perform categorical age estimation (adult and children) followed by an age estimation using an age-determination function based on Support Vector Machine Regression. In fact, AAM's parameter-based classification is not an ideal approach to age categorization in the sense that these appearance data are not exclusively able to express enough cues that are necessary to produce an accurate result in classification. Finally, the authors suggested for further expansion in categorical estimation part, in particular, to develop it as a robust categorical age estimation approach.

Geng et al. proposed an approach where ageing patterns are generated for each person in a dataset of facial images presenting every single subject at different ages [8]. The authors introduced a single sample, a collection of temporal face images for each subject, which is finally projected to a low dimensional space. At testing phase, an unseen face is substituted at multiple orientations in a pattern to indicate age of the subject using the process of minimizing the reconstruction error. Here, the authors suggest methods using unique characteristics of ageing like ageing patterns perform better than the standard classification techniques.

Ageing patterns using manifold learning is demonstrated by Fu and Huang [9], where they employed a manifold criterion based discriminative subspace learning to represent the low-dimensional ageing manifold. The authors suggest age estimation improves significantly when Regression is applied on the ageing manifold patterns. Based on this findings Guo et al. [10] applied a Support Vector Machine Regressor (SVR) to learn the relationship between age and coded face representations. Application of a local SVR trained with only ages within a small interval around the initial age estimate utilizing a refined age estimation using a local SVR is the key aspect of this particular work.

Error-Correcting Output Codes (ECOC) on the fused Gabor and LBP features of a face image are employed by Wang et al. [11] to categorise an individual into one of four possible age groups (child, teen, adult and senior adult). The authors found ECOC combined with AdaBoost or SVM solved the multiclass learning problem like age categorization. FG-NET and Morph datasets are employed to obtain experimental outcomes on its effectiveness and robustness in age categorization and found the algorithm based on the fused features performing better than the one based on Gabor alone or LBP alone.

Our literature review reveals a robust approach for categorical age estimation especially adult and children is still missing – the void which this research is looking to fill.

3 Model Implementation and Training

Someone easily can envisage how much challenges are there to capture all the nuances of child face using only low and high level features like colour, textures, and other hand-crafted descriptors alike while, excluding adult faces. In this paper, a new method of a novel child face detection technic is presented that detects features considered distinctive, compared to the basic low and/or high level features (colour, texture, shape, edge etc.) and other hand-crafted features. The proposed approach leverage knowledge from deep convolutional neural networks to acquire discriminative patterns straight away from the available data. However, it involves a high price tag, the requirement for proper training data incorporating the diverse viewpoints and problem point of view itself. This deep learning approach performs more accurately than existing methods, as demonstrated empirically using low and high level features and other hand-crafted features.

A ubiquitous phrase "a picture is worth a thousand words". That might be true for humans, but can a machine find the same meaning in the images too? Human has got photoreceptor cells in their retina to pick up wavelength of light, but that information doesn't seem to propagate up to human consciousness as a human can't express exactly what wavelength of light he/she is picking up. In a similar manner, a camera captures pixels only. Now, the challenge is how we get human level perception from these pixels.

In this vein, in this paper, we propose a novel approach that allow us to take benefits of deep learning concepts while, at the same time, not having adequate data to train a data-driven solution for child face detection from scratch. The basic features like colour, texture, shape etc.) along with other handcrafted features are in fact, descriptors that are built on a set of cue depending upon the specific feature being used to characterise regions of interest in an image. While on the other hand, deep learning paradigm use Convolution Neural Network (CNN), a set of structured layers of neural network, where each layer acts as a feature extractor resulting quite generic and a specific classification task independent.

One of the great advantage of the Convolutional neural networks (CNN) is that a local understanding of any imagery data is good enough to generate expression on that data resulting fewer parameters that greatly improves the data learning efficiency as well as reduces the amount of data required to train the model. While on the other hand, a fully connected layer (FCN) takes weights from each pixel resulting a larger number of parameter. CNN looks at a small patch of the image at a time to learn the weights rather than looking at each pixel in FCN. In fact, CNNs are a clever way to reduce the number of parameters by reusing the same parameters multiple times instead of dealing with FCN. In CNN model, the number of parameters is independent of the size of the original image.

Our proposed CNN model will learn how to classify images to one of the two candidate categories. In effect, "a picture is worth only one word" out of just two possibilities. The proposed method starts with input images to subject them under a series of convolution, pooling, and local response normalization operations as found to be applied in the literatures.

The proposed network consists of three convolutional layers each of them having a Rectified Linear Units (ReLU) layer, three max pooling layers, one fully connected layer, and one final classification module.

Everything starts in the proposed network with receiving input images and transforming them with a series of convolution, pooling, and local response normalization operations, similarly to many other convolutional neural network models in the [12]. We are primarily aimed at observing the larger objects structures and then gradually looking at the smaller objects. Training images into a convolution layer combined with a Rectified Linear Units (ReLU) layer, where we initially applied a 7×7 kernel filter to capture larger structural facial features like nose, lip, mouth, ear, eye, eye brow, chin etc. followed by a max pooling layer. ReLU applies to maintain the non-linearity of the network with the help of a non-saturating activation function, which increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Max pooling layer applied to perform rescaling or subsampling a convolved output helps reduce the number of parameters that eventually help to not overfit the data. Rescaling or subsampling a convolved output helps reduce the number of parameters, which in turn can help to not overfit the data. The main idea behind this max pooling technique is, it sweeps a window across an image and picks the pixel with the maximum value. Depending on the stride length, the resulting image is a fraction of the size of the original. In our proposed technique we applied a stride length of 2. Another 7×7 kernel filter applied on the output (feature map) from the first convolutional layer to capture more details on the larger facial structure.

We applied similar convolution layers, each having a 5×5 kernel filter, and a 3×3 kernel filter along with a ReLU layer inside and a max pooling layer to incorporate finer details of both the larger and smaller facial objects, each of them followed by a convolution with the same size of kernel filter on the output of the respective convolution. After several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Finally, a softmax classification layer is applied to convert our fully connected layers output into probability with the help of a softmax function that takes the final feature vector and transforms it into a vector of real number in range (0,1). Now our deep neural network model is fully trained to classify face images in to different category.

To address the problem of overfitting, we employed polynomial power of 0.5 alongside a weight decay of 0.01 as a bias term to ensure smooth backpropagation. A bias term and an activation function ReLU are introduced in the convolution layer to maintain its nonlinear behaviour that improves expressiveness. We select a dropout rate of 35%, a learning rate of 0.0005, and a maximum number of 200 epochs while we trained our CNN model. Further, we employ a regularization term in the cost function to combat overfitting. A dropout function is applied to all the layers to minimize overfitting where it takes a randomly selected number of features in that layer while training to make the network redundant and robust to infer output.

4 Experiment Setup, Results, & Discussion

Our proposed approach primarily aimed at developing a robust categorical age estimation tool that can detect adult and child faces automatically. Initially, we have tested our method on a wide variety of facial image databases freely available on the Internet to check its performance and robustness in a real-world scenario. Finally, a more intensive testing has been conducted for performance evaluation on classification accuracy along with an exhaustive performance comparison with few existing classification techniques. Accuracy and finding the correct category are the criteria based on which effectiveness of the proposed approach is measured. We employed a standard set of criteria and procedure to make a judgement on the detection/classification accuracy.

One of the most significant challenges in investigation and implementation of such a sensitive detection system is non-availability of appropriate and authenticated facial datasets on children which is considered to be a major constraint in this particular area of research. For validation purpose, we need datasets on facial images on both adult and children having zero or minimum expression. While freely available datasets are extremely limited, a few authenticated paid databases are available from different research communities. In this paper, we utilise the freely available databases for our experimental evaluation, validation and performance comparison.

Further, non-availability of images having zero or minimum expression is another bottleneck as the majority of the databases are predominantly targeted to a classification based on the facial expression – regardless of adult or child.

Generally, face images in these databases are not ready to use straight away and need pre-processing like cropping, scaling, resizing and enhancement. A vast majority of them aren't suitable for use due to poor resolution and other defects of the original image. Apart from that, the individual photo needs to be cropped from group photos to meet our project's requirement. To run our experiment, children, youths, and seniors have taken into consideration. So, we have selected the required number of images from the aforementioned image databases while maintaining the actual requirement of our purpose. Facial images having neutral or minimum expression are carefully selected though these are hard to get. We have taken extra care in choosing the required facial images from these databases as mentioned above. Further, infants are discarded from the children as they are considered not to be vulnerable as reported by a wide range of literature, article, and online surveys in regards to a binary categorization – either adults or children.

Now, we first outline the datasets and algorithms used in both experiments. We then outline the method for analysing the results.

Face images for training and testing have been sourced from [13–15]. The NCMEC is a publicly available database maintained by an US organization working on missing and exploited children and the FG-NET contains scanned face images from newborns to senescence. The LAG facial database contains 3828 images having images ranging from child/young to adult/old. We have selected 2000 images

Table 1 Facial images taken from different databases

Method	Adult	Children	Total
FG-NET	300	100	400
NCMEC	0	200	200
LAG	700	700	1400

from these databases to run our experiment for comparing our results with that of a couple of recent published work. Table 1 summarizes the number of facial images taken from different datasets.

Corrupted value in the imaging data set is one of the major issues as noise are being introduced due to diverse imaging artifacts like low intensity, intensity inhomogeneity, and shadow. Camera electronics and its lens characteristics play a significant role to embed further noise. More noise means more uncertainty. A low-pass filter is applied to the data set to minimize the overall impact of noises resulted from the inherent imaging artefacts. Apart from these difficulties, other difficulties like intensity inhomogeneity, unclear edges of different segments or objects shadows, shading and highlight effects in image dataset are also present in the data set. We apply adaptive histogram equalization to eliminate these difficulties.

Out of the three datasets as mentioned above, 1000 images have been chosen from each category (adult or child) for our experimental purpose. In order to generate new training and test images our image dataset has been augmented through flipping horizontally or vertically and scaling up or down resulting in 4 times bigger the size of the original dataset.

We replace softmax classification layer as we used during training of our CNN model with a binary classifier in our testing phase as we are primarily aimed at classifying the images into two groups – adult or child. Non-linear Support Vector Machine (SVM) with RBF kernel [16] is applied here to classify the images using the LIBSVM library. A grid search is performed on the training set to estimate the SVM parameter.

Our test image having a size of 256×256 is fed initially into the trained CNN model to classify, where it passed sequentially through a series of CNN layers to extract the features of the input image and finally fed into the SVM classifier following a fully connected layer where extracted features are accumulated in to a feature vector.

Our literature review suggest different authors contributed a good number of face detection techniques already over the decades though not paid too much attention in the area of categorical age estimation, the issue we are dealing with in this particular paper. For quantitative performance comparison, we have to rely on only a couple of literature [7, 17] as no other categorical age estimation approach is currently exists. A popular measure, called Precision and Recall (Eqs. 1 and 2), based on True Positive (TP), False Positive (FP), and False Negative (FN) is applied on both – our experimental results and the results obtained from [7], to evaluate the classification performance of our novel approach. Apart from the Precision and Recall, we compared accuracy (Eq. 3) of our approach with [7, 17].

Table 2 Comparison of performance of our proposed method with other existing methods

Methods	TP	FP	FN	TN
Khoa et al	937	466	263	734
ICSIPA	1119	88	76	1117
Proposed	1163	36	37	1164

Table 3 Comparison of accuracy of our proposed method with other existing methods

Methods	Precision (%)	Recall (%)	Accuracy (%)
Khoa et al.	67	78	70
ICSIPA	93	94	93
Proposed	97	97	97

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{T + N} \tag{3}$$

Out of 8000 face images in our dataset, we have divided the dataset into training and testing sets in to a 70:30 ration each of which representing child and adult face equally. Our CNN model is trained with the training images to classify the test images.

The recognition results were evaluated on a dataset composed of 2400 images (not included in the training set) – 1200 for children, the rest of the 1200 images were adult face images taken from the training set (Tables 2 and 3).

5 Conclusion

We have proposed a novel deep learning-based child face model, where low level features works in tandem with some high level features as extracted automatically by a trained CNN model on facial images to detect child and adult image with maximum accuracy. To the best of our knowledge, this is the first deep learning-based approach which is able to categorize a face image – either a child or adult successfully with maximum accuracy. In doing this, it paves the foundation for research in this particular area. Apart from the categorical age detection, this robust approach will also able to recognise real contents using contextual constraints in images from a wide range of applications like security & surveillance, human body parts identification, dermatological applications and even in marketing – tailoring advertising in shop windows based on the age of shoppers walking past. As a result, an extended avenues and usefulness for research in computer vision domain emerges as an outcome of this particular work. These alternate uses for the outcomes of this

particular work provide extended usefulness and avenues for extended research in cybercrime and computer vision domain. This work can also facilitate large-scale analyses of the prevalence of child exploitation material online [18], which is a precursor to identifying strategies to reduce the demand for this material [19].

References

1. Kwon YH, da Lobo Vitoria N (1993) Locating facial features for age classification. In: Proceedings of SPIE – the International Society for Optical Engineering, vol 2055. SPIE – The International Society for Optical Engineering, Bellingham, pp 62–72
2. Kwon YH, Lobo N (1999) Age classification from facial images. Comput Vis Image Underst 74(1):1–21
3. Hayashi J et al (2001) Method for estimating and modeling age and gender using facial image processing. In: Seventh international conference on virtual systems and multimedia. IEEE Computer Society, Los Alamitos, pp 439–448
4. Lanitis A et al (2004) Comparing different classifiers for automatic age estimation. IEEE Trans Syst Man Cybern B 34(1):621–628
5. Cootes TF et al (2001) Active appearance models. IEEE Trans Pattern Anal Mach Intell 23:681–684
6. Yan S et al (2007) Auto-structured regressor from uncertain labels. In: International conference on computer vision
7. Khoa L et al (2009) Age estimation using active appearance models and support vector machine regression, biometrics: theory, applications, and systems, 2009. BTAS '09. IEEE 3rd international conference. Washington, USA, pp 1–5
8. Geng X et al (2007) Automatic age estimation based on facial aging patterns. IEEE Trans Pattern Anal Mach Intell 29(12):2234–2240
9. Fu Y, Huang TS (2008) Human age estimation with regression on discriminative aging manifold. IEEE Trans Multimed 10(4):578–584
10. Guo G et al (2008) Image-based human age estimation by manifold learning and locally adjusted robust regression. IEEE Trans Image Process 17(7):1178–1188
11. Yan S et al (2009) Synchronized submanifold embedding for person independent pose estimation and beyond. IEEE Trans Image Process 18(1):202–210
12. Krizhevsky A (2012) ImageNet classification with deep convolutional neural networks, In: Advances in neural information processing systems, pp 1097–1105
13. NCMEC, National Center for Missing & Exploited Children (NCMEC). http://www.missingkids.com
14. Face and Gesture Recognition Research Network. http://www.fgnet.rsunit.com/
15. Simon B (2017) Large age-gap face verification by feature injection in deep networks. Pattern Recogn Lett 90. https://doi.org/10.1016/j.patrec.2017.03.006
16. Chang CC, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2 (1):27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
17. Islam M et al (2011) Child face detection using age specific luminance invariant geometric descriptor. In: 2011 IEEE international conference on signal and image processing applications (ICSIPA2011)
18. Watters PA (2018) Investigating malware epidemiology and child exploitation using algorithmic ethnography. In: Proceedings of the 51st Hawaii International Conference on Systems Science (HICSS), Hawaii
19. Watters PA (2018) Modelling the efficacy of auto-internet warnings to reduce demand for child exploitation material. In: Proceedings of the 22nd Asia-Pacific conference on knowledge discovery and data mining workshops

Toward Detection of Child Exploitation Material: A Forensic Approach

Mofakharul Islam, Paul Watters, Abdun Nur Mahmood, and Mamoun Alazab

Abstract With continual advances in Internet capability, in addition to its global and decentralized nature, the Internet along with different social networking sites are experiencing a boom in demand and supply. Recent study found that the social networking sites like Facebook, Twitter, and MySpace are providing a forum for paedophiles to share child pornography. With the advent of sophisticated digital technology, Law Enforcement Agency (LEAs) around the world dealing with child pornography facing real challenge to combat with the technologically-savvy paedophiles. The major challenge in child pornography lies in authentic detection of children face in an image. The main objective of this research is to present a novel framework for a dedicated child face detection tool, where we will use child's face specific contextual contexts and visual cues that are based on new knowledge in terms of features or contexts representatives of child's skin and face. The proposed technique can estimate age categorically – adult or child based on a new hybrid feature descriptor, called *Luminance Invariant & Geometrical Relation based Descriptor* (LIGRD). LIGRD is composed of some low and high-level features, which are found to be effective in characterizing the local appearance in terms of chromaticity, texture, and geometric relational information of few facial visual cues simultaneously. Comparison of our experimental results with that of another recently published work reveals our proposed approach yields the highest precision and recall, and overall accuracy in recognition.

Keywords Child exploitation · Child pornography · Digital forensics · Intelligence analysis · Face detection · Deep learning · Cybercrime

M. Islam (✉) · P. Watters · A. N. Mahmood
Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia
e-mail: M.Islam3@latrobe.edu.au; P.Watters@latrobe.edu.au; A.Mahmood@latrobe.edu.au

M. Alazab
Charles Darwin University, Casuarina, NT, Australia
e-mail: mamoun.alazab@cdu.edu.au

M. Alazab, M. Tang (eds.), *Deep Learning Applications for Cyber Security*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-13057-2_11

1 Introduction

Before the advent of the Internet, clandestine porno magazine or porn-cinemas were the principal means for the production, distribution, exhibition, possessing and consumption of pornographic material. With its arrival, the Internet quickly became the prime medium for the distribution, transmission, and exhibition of such materials. Parallel to the changes in pornographic media, content has also changed, in that an increase in violence and misogyny could be found in magazines, on videotapes and on the Internet. Internet access to the computers and choice of social-networking sites are increasingly a modern day must have for children and young people around the world establishing new cultural norms and becoming mainstream within education. Clear majority of children and young people using the Internet for searching their educational stuff, playing games online, keeping themselves in touch with their social and online friends and as a source of perfectly innocent fun and games for the great majority of the time. In doing so, it is almost certain that at some stage these children and young people will be exposed to material that will cause trauma in their mind and possibly damage them psychologically. The worst part in that is, eventually it will lead them encounter organization or individuals who mean them injury. To follow potentially catastrophic consequences throughout a children's lifetime, he/she needs only one such encounter to go wrong. It is therefore no surprise that the Internet as well as the social networking sites pose a serious threat to our children who are vulnerable, either some or all the time while they are on the Internet. In fact, potential outcome of the Internet use is questionable as a little is understood yet of the potential problems and benefits associated with it, and the overall impact on the society from social benefit point of view that may arise. The risks posed by criminals who attempt to share, exchange, consume and produce child exploitation material, however, are clear, and law enforcement is faced with the difficult task of trying to deal with the sheer volume of material (and offences) in a streamlined and systematic way.

One study in 2011 [1] revealed the Internet usage and population statistics across the globe as shown in Table 1. Another study in 2002 showed that across the whole world the number of children and young people regularly using Internet was running towards the 80 million mark at that time, and this number is increasing geometrically. A recent study showed that 27 million American children between the ages of 2 and 17 are active Internet users, and that is one in five of all US Internet users [2].

Children Internet pornographic statistics in Table 2 shows that onset of viewing pornography starts from early on. The average age of a child's first exposure to pornography is 11. A total of 90% of children ages 8–16 have viewed pornography online. Paedophiles try to exploit children using many character names that appeal to them such as "Pokemon" [1].

According to compiled figures from various news and research organizations, every second, $3*;* 075*:*64 is being spent on pornography. Every second, 28,258 internet users are viewing pornography. In that same seconds, 372 internet users are

Table 1 World Internet usage and population statistics' 2011 [1]

Regions	Population	Users	Penetration (%)
Africa	1.04 bil.	119 mil.	11.4%
Asia	3.88 bil.	922 mil.	23.8%
Europe	816 mil.	476 mil.	58.3%
Middle East	216 mil.	69 mil.	31.7%
N. America	347 mil.	272 mil.	78.3%
L. America	597 mil.	216 mil.	36.2%
Carib. Oceania/Aus	35 mil.	21 mil.	60.1%
Total	**6.93 bil.**	**2.10 bil.**	**30.2%**

Table 2 Children Internet pornography statistics' 2011

Average age of first Internet exposure to pornography	11 years old
Largest consumer of internet pornography	35–49 age group
15–17 year olds having multiple hard-core exposures	80%
8–16 year olds having viewed porn online	90% (most while doing homework)
7–17 year olds who would freely give out home address	29%
7–17 year olds who would freely give out email address	14%
Children's character names linked to thousands of porn links	26 (including Pokémon and action man)

typing adult search terms into search engines. Every 39 min, a new pornographic video is being created in the U.S. In fact, legal pornography is huge but legal business. The pornography industry has larger revenues than Microsoft, Google, Amazon, eBay, Yahoo, Apple and Netflix combined. In 2006, worldwide pornography revenues ballooned to $97:06 billion. Pornographic revenue in Australia alone stands at US$ 2 billion (per capita AU$98:70) in 2006 [1].

Out of these statistics, a significant amount goes to purchasing child pornography, or has a link with adult pornography, if a **Progression Model** as shown in the Fig. 1 is assumed.

Some research suggests that becoming a paedophile is a gradual process that may start with viewing a variety of pornographic material, then gradually looking at the "teenage" sites and others, once the viewer has become desensitised to "legal" material. Over time, such users access more extreme material, in an endless quest for novelty, and these sites eventually point them to younger pictures, and finally, they may end up viewing illegal sites containing child pornography. Looking at ever-younger girls is a steady downward trend for the paedophiles [3], and the content can be extreme: the COPINE scale provides some indication of the extreme themes which are often pursued.

The annual turnover of child pornography businesses has been recently estimated at US billion globally [1]. More $3 billion globally. More than 100,000 web sites

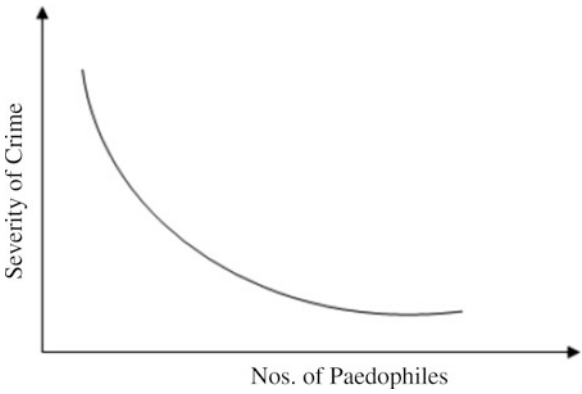


Fig. 1 Children pornography progression model

have been identified as engaging in the primary business of selling child porn to others [4–6].

No doubt, the Internet has wormed its series of blissful tubes throughout our modern lives. It's what people check in the morning, what people look at during the day, what entertains them at night. News, products, business opportunities, conversation, relationships, self-promotion, entertainments – what's not on the Internet. An entire generation is growing up having never known a time without the Internet [7]. All of us will agree on is the ever-increasing dominant role this technology can play in our kid's daily lives, social networking sites in particular. With choices such as MySpace, Twitter, Xanga, Tumblr and the mother of all, Facebook – it's hard to find a teen now who never visited any of these sites. Currently, the most popular one, Facebook that started its journey in 2004 and opened to public in 2006 having 42 million active users, playing a dominant role in reshaping online behaviour and attitude of the modern-day Internet users. Its unparalleled ability to attract users using its virtual mode of communication and capability of providing almost everything an Internet user can be dreamt of, made it as an integral part of what people want and more importantly expect. Advances in the functionality and the usability of systems have provided criminals with opportunities to use – most often misuse technologies in ways in which they were never intended. Although these social networking sites specifically claims themselves as safe site, a series of investigations conducted by US multi-state investigations suggested that the Facebooks filters miserably fall short in blocking child pornography. The Facebook is failing to prevent child predators from posting suggestive and potentially illegal photographs of children on its website, a weekslong investigation by FoxNews.com reveals, despite its claim that its doing all it can to keep paedophile materials from being displayed. Facebook employs content-based filtering system that able to scan automatically few basic keywords most commonly being linked to child pornography as identified by the National Centre for Missing and Exploited Children (NCMEC). But FoxNews.com took two Facebook executives on a click-by-click tour of the Facebook in a lengthy telephonic interview on Oct. 6, 2010 bringing them face-to-face with some of its vile contents

and forcing them to admit that their efforts to block child predators were not working at all. During a 90-min phone interview with Facebook spokesman Simon Axten and the company's chief security officer, Joe Sullivan, FoxNews.com was able to expose the underbelly of Facebook. These executives were stunned viewing the extremity of graphic contents on the site. Even they are unaware of and unable to explain the extremely graphic content on their site [1].

In the said interview, FoxNews.com asked the executives to put the word PTHC in their sites search box and subsequently instructed to click on the first result identified as a public group Page called PTHC, with 197 members. When the executives tried to get access a post there it directed them to a video allegedly made on an 8-year-old boy being sexually abused. The term PTHC stands for Pre-Teen Hard Core – is linked with child sexual exploitation activity and materials that is frequently being used by the paedophiles, and it is on the NCMEC list of keywords. Later, when they are further asked to click on the profile of any of the group's members, the executives were steered into a subculture dedicated to using Facebook to traffic child pornography and to target and interact with children [8].

Recently, a US State Attorney General announced an investigation as the latest scrutiny for Facebook and similar sites, such as MySpace, that have been criticized for not regulating the interactions between adults and minors. In this investigation, undercover investigators were posing as children in online social networking website Facebook, where they were allegedly solicited by sexual predators and the users could easily access pornographic images. The undercover investigators also contacted Facebook and made complain as upset parents, but most often their complaints were ignored. In some cases, Facebook did respond by taking down inappropriate material [9].

It is therefore no surprise that child pornography is increasingly emerging as one of the most devastating artefacts of today's wired and global society. The potential field of research into criminal and deviant behaviour on the Internet is vast though few studies have yet been made to assess the magnitude of its impact on our society. Despite highly controversial and the subject of immense interest, researchers paid less attention to child pornography and its prevention on the internet.

Although weaving cyberspace into the fabric of our society, providing immense benefits to us, it is on the other hand, emerges as an increasingly visible problem to the mankind today. With the advent of sophisticated digital technology Law Enforcement Agency (LEA) around the world dealing with child pornography facing real challenge to combat with the technologically-savvy paedophiles. Recent growth in PCs at home and social-networking on the Internet combined with evolving field of digital technologies added a new dimension to paedophilia, resulting a convenient way to make their business by sharing horrific pictures of children being sexually abused – most often for profit. To combat child pornography, LEA are currently using some techniques that are both inefficient and relatively basic, resulting in a high failure rate of correct child porn image identification that could be used as forensic evidence in court. Rapid and accurate detection and prosecution of child pornographers are currently handicapped due to a lack of appropriate technology – technologically-savvy child pornographers attempt to

outwit law enforcing agencies by using newly developed and sophisticated state-of-the-art digital imaging and video processing facilities, equipment and software.

We propose a novel approach to implement a tool using a new model-based approach for detecting and identifying child face from images based on new knowledge in terms of child skin and face specific features or contexts. Successful implementation of the proposed approach combined with any existing state-of-the-art pornography detection techniques could be used as a dedicated tool to inspect huge volume of confiscate hard drives as anecdotal evidence. The recent growth in PCs at home and work (with Internet connectivity) provides paedophiles a convenient way to conduct their business, by sharing horrific pictures of children being sexually abused – often for profit. In addition to introducing strategies to try and reduce demand, often working against technologically savvy paedophiles, Law Enforcement Agencies (LEAs) around the world have had to deal with larger volumes of child pornography cases. To prosecute child pornography cases, LEAs typically use manual techniques that suffer from being slow, potentially inefficient and error-prone. Rapid and accurate detection (and prosecution) of paedophiles is somewhat handicapped due to a lack of appropriate technology – technologically savvy paedophiles attempt to outwit law enforcing agencies by using newly developed and sophisticated state-of-the-art digital imaging and video processing facilities, equipment and software. Literature suggests that LEA's digital forensic units currently are experiencing tremendous backlogs of cases due to the huge volume of child pornography related cases. A recent study showed that, in the Netherlands alone, the amount of data confiscated by the Police has increased from4.5 to 152 terabytes between 2003 and 2006 [10]. The huge amount of data must be manually inspected to find conclusive evidence for child pornography is a major challenge for law enforcing agencies. This inspection usually must be achieved in a very limited time frame, to keep a suspect and his PC in custody without a specific charge, or due to the statutory limits around the holding of evidence. The task of sifting through confiscated PC disk images to identify child pornography on the other hand is tedious, time consuming, emotionally taxing and difficult. As such, there is a pressing need for an accurate, faster and automated tool to detect and identify child pornographic images effectively regardless of its quality of production.

In addition, the proposed approach can be effectively integrated with any existing pornography detection approach to implement a dedicated real-time application tool to monitor and detect pedophilic activity on the Internet and other social networking via the Internet. For having a better and authenticated detection of pornographic evidence in an image, we can apply a dedicated and robust detection scheme based on pornography specific contextual constraints that are representatives of actual pornographic activity rather than relying on some heuristic skin region contents or ratios and features as employed in the existing techniques. A stochastic approach based on Markov Random Fields (MRF) modelling has been found to be effective in modelling contextually dependent physical phenomena especially in a degraded situation where source images suffer numerous imaging artefacts. Ongoing work on a solution to the problem of real-time pornography detection is undertaken by

Fig. 2 Spanning 4 generations. (Courtesy: The Mail, March 31, 2010)

us, where we employ a novel stochastic vision model based on Markov Random Fields (MRF) prior. The MRF prior encodes the pornography specific contextual constraints to facilitate accurate and authenticated detection and identification of skin regions containing pornographic contents (Fig. 2).

In this work, we are interested in using the computational foundations of vision to help us design machine vision systems with applications to real-time detection and forensic investigation. Our broader research program aims to find solutions for some fundamental questions in computer vision. How can we recognize persons, guns, cars, boats and many other suspicious categories of objects in cluttered pictures? How can we gain knowledge of these classes in the first place? Can we provide the identical skill to computer?

We propose a novel approach to implement a tool for detecting and identifying children's faces from images. We utilize specific facial features or contexts, where low level skin features will detect the skin regions from images. We then exploit some visual cues from human faces to incorporate new knowledge in terms of spatial relationships among the low-level features along the contour of these visual cues.

However, identifying child (or adult) faces from images is a real challenge that has been in the literature with no general solution proposed so far. In many application areas – from identifying the age of actors by the classification review board, through to determining whether a porn image contains children or not – identifying age quickly accurately remains an unsolved problem. Detecting skin in images is a solved problem, but the literature suggests all the existing techniques are general – none of them are adult or child skin specific. A large number of existing state-of-the-run techniques for detecting adult skin are available, especially for nude person identification. In fact, pornography detection is a solved problem now. Therefore, this work is primarily aimed at child face detection to categorize a child face only, which is eventually applied to categorise the subject (whole image)

either into child or adult.. Afterwards, a state-of-the-art pornography detector can be applied on the detected child images (the whole images – not the face only) to detect whether the image contains pornography contents or benign.

However, identifying child (or adult) faces from images is a real challenge that has been studied in the literature with no general solution proposed so far. In many application areas – from identifying the age of actors by the classification review board, through to determining whether a porn image contains children or not – identifying age quickly accurately remains an unsolved problem.

Our hypothesis is that we can exploit children's face specific contextual cues in terms of skin and visual cues on face from images to perform the estimation, particularly where age is ex pressed categorically rather than continuously (child or adult, for example). For detecting children face from pornographic images, we will employ a novel children's facial model based on contextual cues representatives of children's skin and some crucial cues on human face, which is first of its kind. Detecting pornography in images is almost a solved problem now, the reason why the real challenge in child pornography detection lies in the detection of a child skin/face.

The article has five main sections. The Sect. 2 is largely descriptive on the existing pornography detection and skin detection techniques that have close resemblance with this work. Section 3 will describe our proposed research methodology followed by experimental results and discussion in Sect. 4. Finally, conclusion and further future research will be discussed in Sect. 5.

2 Related Work

In general, pornography detection techniques are primarily based on two parts – skin modelling & classification technique and classification of pornographic contents. Skin classification technique comes first before applying pornographic content classifier as the latter is being applied on the skin patches as obtained from the former.

2.1 Existing Works on Child Pornography Protection

To the best of our knowledge, only a couple of literatures published from the University of Ontario Institute of Technology exist that are aimed at child pornography detection explicitly so far. Shupo et al. [11] suggested a network-based detection system that uses a stochastic weak estimator coupled with a linear classifier. The features that were extracted from the images – both training and testing, are some statistical properties from the labelled packets (those that have been stored into the pornographic and benign vectors). Later these vectors as obtained in training stage were fed into the classifier to train the classifier. The estimators they consider are the

stochastic learning weak estimator (SLWE) and the maximum likelihood estimator (MLE). The SLWE is considered to be more accurate in dealing with non-stationary data1. To classify the image the authors employed four different distance functions – Euclidean distance, Weighted Euclidean distance, Variational distance, and Counter distance between two aforesaid feature vectors. The authors gathered a sanitized image dataset on child pornography from Toronto Police, Canada, while the non-child pornography image datasets were gathered using random images obtained from the Internet.

Being inspired by the outcome of the aforesaid work, A. A. Ibrahim extended [12] the research as a research Master project from the same institution, where he employed almost the same methodology but on an artificial image dataset this time. The author did not use any real-world child pornographic image; rather they employed an artificial dataset composed of adult pornographic images collected from the Internet.

2.2 Existing Works on Age Estimation

In age estimation, technology that has been proposed and implemented so far consists of content-based image analysis tools, like some low-level skin features and/or age specific features like wrinkle and other heuristic age features that are, in fact, not based on any adult or children specific facial contextual constraints.

Age estimation from digital facial images initiated by Kwon and Lobo [13], where they employed high resolution facial images to classify into three age groups – babies, young adults or senior adults. Authors reported 100% accuracy in classification but application on a limited database indicates that it is not applicable in real world applications. To accomplish the similar age group classification, they suggested another age classification technique based on craniofacial development theory and wrinkle analysis later [14]. Again, the approach is handicapped by limited dataset and thus very hard to assess its robustness while applied on real world large datasets. Hayashi et al. [15] employed wrinkle analysis along with geometric relationships between significant visual cues on human face to classify age into multiple groups at the 5 years intervals, which in fact, not a categorical age estimation technique rather it is focused on age-group wise classification.

Active Appearance Models have been employed by Lanitis et al. [16], where combined shape and texture parameters are extracted to estimate the age using three classifiers – simple quadratic fitting, shortest distance classifier, and Neural Network and age estimation accuracies of these classifiers are compared. Statistical modelling on ageing patterns (a sequence of personal facial age images) as suggested by Geng et al. [17] primarily based on the assumption that multiple images of different ages are available for each person. Yan et al. [18] suggested an algorithm, where the uncertainty of the age labels is considered in age estimation. Recently, Khoa et al. [19] suggested an approach combining Active Appearance Models (AAM) [20] with Support Vector Machine Regression to classify different

ages. Initially, a classification is performed to classify age categorically (adult and children) using AAM's parameters then age-determination functions based on Support Vector Machine Regression are applied to estimate the age. In fact, classification using only AAM's parameter is not an ideal approach to classify age as appearance data exclusively are not able to provide adequate cues that are necessary to obtain accuracy in classification. This approach basically aimed at predicting specific age rather than categorical estimation like adult or children, the reason why authors recommended for further improvement especially in this part to make it a reliable categorical age estimation approach.

Geng et al. proposed an approach where ageing patterns are generated for each person in a dataset of facial images presenting every single subject at different ages [21]. The authors introduced a single sample, a collection of temporal face images for each subject, which is finally projected to a low dimensional space. At testing phase, an unseen face is substituted at multiple orientations in a pattern to indicate age of the subject using the process of minimizing the reconstruction error. Here, the authors suggest methods using unique characteristics of ageing like ageing patterns perform better than the standard classification techniques.

Ageing patterns using manifold learning is demonstrated by Fu and Huang [22], where they employed a manifold criterion based discriminative subspace learning to represent the low-dimensional ageing manifold. The authors suggest age estimation improves significantly when Regression is applied on the ageing manifold patterns. Based on this findings Guo et al. [23] applied a Support Vector Machine Regressor (SVR) to learn the relationship between age and coded face representations. Application of a local SVR trained with only ages within a small interval around the initial age estimate utilizing a refined age estimation using a local SVR is the key aspect of this particular work.

Error-Correcting Output Codes (ECOC) on the fused Gabor and LBP features of a face image are employed by Wang et al. [24] to categorise an individual into one of four possible age groups (child, teen, adult and senior adult). The authors found ECOC combined with AdaBoost or SVM solved the multiclass learning problem like age categorization. FG-NET and Morph datasets are employed to obtain experimental outcomes on its effectiveness and robustness in age categorization and found the algorithm based on the fused features performing better than the one based on Gabor alone or LBP alone.

Islam et al. [25] present a new framework for capturing facial image patterns that can be applied in categorical age estimation, which is primarily aimed at offering a novel approach to investigate and implement a child face detection technique that can estimate age categorically – adult or child based on a new hybrid feature descriptor. The novel hybrid feature descriptor LIGD (the luminance invariant geometric descriptor) is composed of some low and high-level features, which are found to be highly effective in characterizing the facial patterns in terms of local appearance. In local appearance estimation, chromaticity, texture, and positional information of few facial visual cues can be employed simultaneously.

Literature suggests, a dedicated approach for categorical age estimation especially adult and children is still missing, and the issue is not yet been addressed – the hole which this research is looking to fill.

3 Our Proposed Approach

Our proposed methodology will be based on a new image modelling on human face – both adult and child specific contextual constraints to be extracted from the pornographic images as to be obtained from an existing state-of-the-art pornography detection tool, where we will finally employ a Support Vector Machine (SVM) classifier to classify children and adult face image. The proposed approach is invariant to illumination changes, camera viewpoint, scaling, affine transformation and other imaging factors like shading, makeup and occlusion.

Child skin detection by machine offers considerable extent of challenges and difficulty to the machine vision community as complete knowledge on human visual system is unknown yet. Human ageing process especially on skin is not controlled by people's gene alone but also attributed to many factors like individual's lifestyle, health, race, occupation, cultural background and many more. Nothing significantly has been contributed in this particular area yet though much interest has been paid on adult skin detection over the last three decades that eventually kept the issue still open. Literature suggests that nothing has been attempted yet to address this issue. Automatic identification of children skin process needs at least some of the similar cues being used by human visual system in making judgment, resulting an effective and efficient way that attempts to quantify children's skin in ways that agree with human intuition. Defining such cues certainly will put this research to a considerable extent of challenge. It is still an open problem to extract general discriminative features for categorical age estimation using skin detection regardless of the negative influence of individual differences.

Smoothness of the skin's surface, hydration of the skin, collagen, elastin and glycosaminoglycan content have direct relation with the skin texture. Several key factors have gradual impact on these essential elements of skin. While a children's skin apparently looks like normal adult skin, there is significant differences in terms of both the structural and functional characteristics. A baby has wonderfully soft, smooth skin that differs significantly with adult skin. That lovely smooth skin of childhood undergoes some significant changes when we reach our teens and continues till adult. At this stage body produce an increased amount of hormones. A hormone called androgens starts to produce in both girls and boys to produce male hormones. Under the influence of this androgen the skin produces more oil (sebum), resulting a tendency for acne to develop. In addition, at the same time muscle starts to develop under the thick, soft, smooth and supple skin due to effective functioning of the stratum corneum and becomes visible at late teens, leaving a wavy skin surface all over the body. The anatomical differences between children and normal adult skin relate primarily to differences in the surfaces of the skin [26].

Human skin changes inexorably as people get older – no matter whatever the race, ethnicity, culture, age, and sex is. A young adult's skin is well hydrated, tends to be soft, smooth and supple, and has a natural translucency; while on the other hand, a more mature adult skin tends to function less well due to start of the ageing process, gradually leaving a dry and tend to feel tight skin with a rough texture and dull appearance. Wrinkles start to appear because of a gradual reduction in the water content of the stratum corneum, lipids and sebum, and some degree of photo damage due to sunburn [27]. Apart from the above, the major skin tone problem that occurs with time is the darkening of skin colour due to the skin cell damage from prolonged sunlight exposure. The top most cell layer in our skin absorbs the natural colouring pigment, melanin, allowing gradual darkening of skin colour due to over absorption. Melanin is produced in the basal cell layer of human skin and dead cells on the surface absorb this colouring pigment. Use of cosmetics, soap, shower gel, and body lotion add an extra degree in gradual skin tone deterioration process. A gradual decline in adult skin tone due to numerous factors makes substantial differences with that of children's skin, favouring us to detect children's skin even with our naked eyes. The answer of the question, how people perceive the difference between two objects is still missing as complete knowledge on human visual system is unknown yet. So, our prime focus is to apply at least few of the cues that human visual system applies in distinguishing different objects in our approach to detect children's skin offering an effective and efficient way that attempts to quantify shape and appearance in ways that agree with human intuition.

3.1 Image Pre-processing Stage

Noise embedded in a data set results in some corrupted values for the data set. This was particularly true in face/skin segmentation/classification problem where data sets are usually noisy due to diverse imaging artifacts like low intensity, intensity inhomogeneity, and shadow. In addition, noise attributed to the camera electronics and lens characteristics made the problem harder. More noise means more uncertainty. So, we apply a low-pass filter to our dataset to minimize these noises originated by the aforesaid inherent imaging artefacts. To remove other difficulties like intensity inhomogeneity, unclear edges of different segments or objects shadows, shading and highlight effects in image dataset, we apply adaptive histogram equalization. Adaptive histogram equalization enhances the contrast of the grayscale image by transforming the values using contrast-limited adaptive histogram equalization. Contrast-limited adaptive histogram equalization operates on small regions in the image, called tiles, rather than the entire image. Each tile's contrast is enhanced, so that the histogram of the output region approximately matches the histogram specified by the desired histogram shape for the image tiles. The neighbouring tiles are then combined using bilinear interpolation to eliminate

artificially induced boundaries. The contrast, especially in homogeneous areas, can be limited to avoid amplifying any noise that might be present in the image [28].

3.2 Low Level Feature Generation

Research reveals that accuracy in skin segmentation/classification primarily relies on colour and texture features. Research also found that segmentation based purely on colour is more sensitive to local variation, but provides sharp boundaries [29–31]. On the other hand, segmentation based purely on texture feature results in fuzzy boundaries, but usually homogenous regions. In other words, colour and texture-based segmentation are in combination capable of producing sharp boundaries and homogenous regions. In fact, an individual object can be detected more efficiently with the combined features of colour and texture than either the colour or texture feature used independently. As a result, segmentation with combined colour and texture outperforms others in terms of sharpness and homogeneity [29–32].

One of the primary goals of this proposed model is to find a suitable low-level feature set composed of color and texture features that are able to discriminate skin and non-skin region effectively with maximum accuracy. Although RGB is the most commonly used color space unfortunately, it is not perceptually uniform and not corresponds to the human visual system. Literature suggests *YCrCb*, *CIE-Luv*, *CIE-Lab*, and normalized rgb are most popular choice in skin detection. Similarly, on the other hand, Wavelet Transform, Gabor filters, and MSRAR have been found to be effective in encoding texture details – coarser and finer both while other texture descriptors are not ideal in complex situations like security & surveillance, bio-medical, bio-engineering, material sciences, mining, mineral and metallurgy, where situation demands a more sensitive texture discriminative ability [29–32]. We employ 12 possible combinations of low level feature sets out of these 4 color spaces and 3 texture descriptors on 100 face images from the NCMEC [33] and FG-NET [34] database and then fed these multidimensional feature sets sequentially as an input into a Nave Bayes' classifier to perform segmentation using an adaptive thresholding technique. Since people with different skins have different likelihood, an adaptive thresholding technique is required to achieve the optimal threshold value for each image. Here, we discard the brightness/luminosity components from the color spaces to make the feature sets invariant to illumination.

Classification error is defined as;

$$Classification\ Error : FP\,(\%) + FN\,(\%)$$

Our study (Table 3) clearly demonstrates the superiority of Chroma *YCrCb* + *Wavelet* compared with its other competitor as it outperforms them in all the three metrics (False Positive, False Negative, and Classification Error).

Table 3 Performance comparison of different color and texture segmentation/feature combination in skin classification

	Different metrics		
Low level Featr. Comb.	FP(%)	FN(%)	Classn. Err. (%)
Chroma_YCrCb +Gabor	6.98	5.83	12.81
Chroma_YCrCb +wavelet	6.71	4.36	11.07
Chroma YCrCb +MRSAR	8.9	8.57	17.56
Chroma_CIE-luv + Cabor	7.14	5.89	13.03
Chronia_CIE-luv + wavelet	7.69	6.41	14.10
Chroma CIE-luv + MRSAR	9.12	7.97	17.09
Chroma_CIE-lab +Cabor	8.98	6.58	15.56
Chroma_C IE-lab +wave let	7.77	8.39	16.16
Chroma CIE-lab +MRSAR	11.06	10.45	21.51
Chroma_rgb + Cabor	9.56	6.12	15.68
Chroma_rgb + wavelet	10.78	8.23	19.01
Chroma_rgb + MRSAR'	10.36	9.87	20.23

3.3 Color Feature

Twenty skin patches were collected by cropping off from face images. A new low level color feature is derived utilizing the pure chromatic information (*Cr* and *Cb*) of the original color model as;

$$Chroma\ C\ (r, b) = C\ (r, b) - C\ (r, b)\, mean \tag{1}$$

Where $C_{(r,b)}mean = \frac{\sum_{n=1}^{N} C_{(r,b)}}{N}$ and N = Nos. of pixels in a single skin patch.

(1) Texture Feature Texture characterizes local variations of image color or intensity. There is no formal or unique definition of texture, even though texture-based methods are commonly used in computer vision. Each texture analysis method defines texture from its own context. However, the neighbourhood property is a common similarity among all the available texture definitions/descriptions. So, we can defin texture as an image feature which is characterized by the gray value or color pattern in a neighbourhood surrounding the pixel.

Smoothness of the skin's surface, hydration of the skin, collagen, elastin and glycosaminoglycan content have direct relation with the skin texture. Several key factors like hormones, diet, sunlight, and various environmental factors have gradual impact on these essential elements of skin. While a children's skin apparently looks similar to normal adult skin, there is significant differences in terms of both the structural and functional characteristics. Some primary impacts occur in the soft-tissue, in the formation of wrinkles, lines, ptosis, and soft tissue while children enter their adulthood. Here, we need to have a powerful texture descriptor that has the ability to encode this subtle texture information effectively.

Transform Domain Feature that refers to a mathematical representation of an image characteristic. There are several texture classifications using transform domain features in the past, such as discrete Fourier transform (DFT), and discrete wavelet transform (DWT). In this particular work, we employ DWT considering its effectiveness in capturing subtle texture information compared with other texture descriptors. Wavelet analysis breaks up a signal into shifted and scaled versions of have collected the original wavelet (mother wavelet) which refers to decomposition of a signal with a family of basis functions obtained through translation and dilation of a special [35]. Moments of wavelet coefficients in various frequency bands have been shown to be effective for representing [36].

Decomposition of textures is one of the keys to access characteristics of textures in different scales and a wavelet transform can do such decomposition. The wavelet transform is based on using an orthonormal family of basis functions. A wavelet expansion is a Fourier series expansion but is define by a two-parameter family of functions;

$$f(x) = \sum_{m,n} C_{m,n} \psi_{m,n}(x) \tag{2}$$

Where and are integers, the functions $\psi_{m,n}(x)$ are the wavelet expansion functions and the two-parameter expansion coefficient, are called the discrete wavelet transform (DWT) coefficients of $f(x)$.

The coefficients are given by

$$C_{m,n} = \int_{-\infty}^{+\infty} f(x) \psi_{m,n}(x) \tag{3}$$

define another low-level texture feature employing the diagonal, horizontal, and vertical coefficients as yielded by the DWT as;

$$dwt_{(d,h,v)} = Coef_{(d,h,v)} - Coef_{(d,h,v)} mean \tag{4}$$

where $Coe\ f_{(d,h,v)} mean = \frac{\sum_{n=1}^{N} Coe\ f_{(d,h,v)}}{N}$ Here,

N = Nos. of pixels in a single skin patch and $Coef_{(d,h,v)}$ denotes diagonal, horizontal, and vertical coefficients.

(2) Local Binary Feature Local binary patterns (LBP) has been found to be a powerful feature allowing a classifier or detector to exploit fundamental properties of local image texture efficiently and effectively in a classification or matching. Applying LBP on an image yields occurrence histogram, which is a powerful texture descriptor that contains information about the distribution of the local micro-patterns like edges, spots, flat areas, over the image region. Label for every pixel of the image is assigned by the original LBP operator applying a thresholding technique on the centre pixel with his 3 × 3 neighbourhood. Local Binary Patterns were originally introduced as a texture descriptor by Ojala [37], and have subsequently been employed as face and expression identification features [38, 39].

By assigning a binomial factor 2^p for each sign $S(g_p - g_c)$ in the following equation, a unique number that characterizes the spatial structure of the local image texture;

$$LBP_{P,R} = \sum_{p=0}^{P-1} s\left(g_p - g_c\right) 2^p \; s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{5}$$

Where, $P =$ nos. of neighbouring pixels, $R =$ radius of the neighbourhood, $g_p =$ gray values of circularly symmetric neighbourhood, and $g_c =$ gray value of centre pixel [40].

LBP encode a thresholded local neighbourhood at the gray value of the centre pixel in the form of a binary pattern and remain constant as long as the order of gray values in the image are not changed. In fact, this is highly discriminative and robust local texture operator that encodes the occurrences of the various patterns in the neighbourhood of each pixel in a P-dimensional histogram. In this work, we employed LBP on two chromatic channels $Chroma_r$and $Chroma_b$ to encode chromatic channel-wise texture description to have better texture discriminative ability.

(3) Facial Appearance Feature In order to incorporate new knowledge on face in terms of location and inter-relationships among the visual cues like eyes, eyebrows, mouth, nose, chin, side of face, we use cranio-facial development theory [14], which states that the appropriate mathematical model to describe the growth of a person's head from infancy to adulthood is the revised cardioid strain transformation, cardioid strain transformation, written in a polar form as $\theta^{'} = \theta$, $R^{'} = R(1 = k(1 - \cos\theta))$,where θ is the angle from the Y-axis, R is the radius of circle, k is a parameter that increases over time, and $(R^{'}, \theta^{'})$,s are the successive growths of the circle over time. According to the revised cardioid strain transformation, top of the head can be visualized as a series of ever growing circles all attached to a common tangent "base" point resulting growth of the lower parts of the face is more remarkable than that of the upper parts. Thus, for example, within the top and bottom margins of the head, eyes occupy an higher position in an adult than in a child, which is not due to eye migration, but instead to an outgrowing and dropping of the chin and jaw.

Localization of these facial cues is performed using iterative framework as described in [41]. One these facial cues are localized, in the next step, geometric ratios are computed using their localized position that distinguish children from adult. A six of ratios are computed by the authors utilizing the relationships between these primary visual features or cues out of which our proposed approach employs the first 5 ratios only as shown in Fig. 3 and computed in [41]. These evaluated ratios only need the automatic localization of the primary features such as eyes, nose, mouth, chin, and virtual top of head. Ratio 1 is the T -ratio formed by two segments – the segment $T1$ joining the two eyes and the segment $T2$ between the midpoint of $T1$ and the nose. Ratio 2 is the T -ratio formed by two segments - the

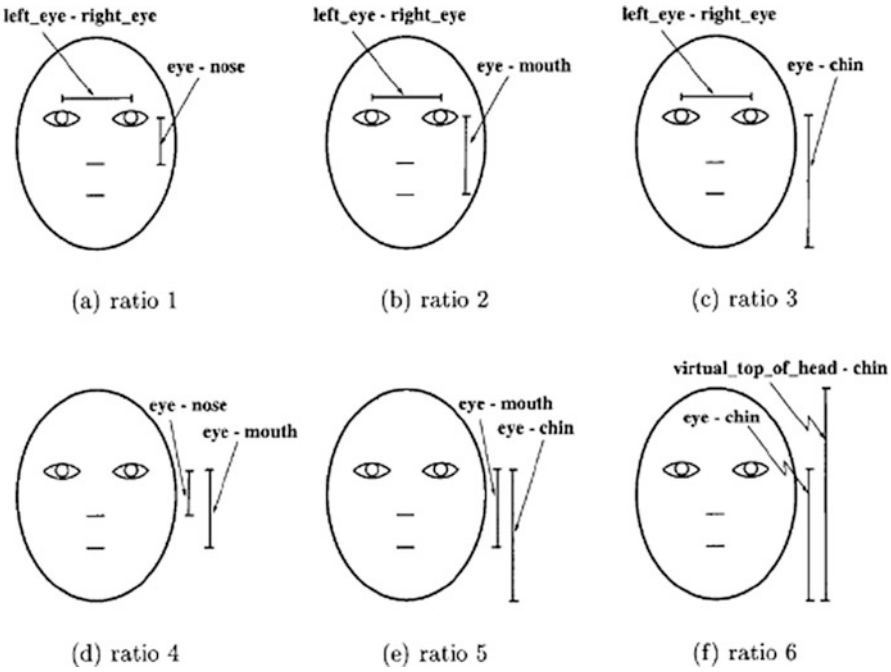


Fig. 3 Six cranio-spatial ratios. (Courtesy: Age classification from facial images [41])

segment *T1* as above, and the segment *T3* between the midpoint of *T1* and the mouth. Ratio 3 is the *T* -ratio formed by two segments - the segment *T1* as above, and the segment *T4* between the midpoint of *T1* and the chin. Ratio 4 is the ratio of the segment representing the difference in height between nose and eye-midpoint, and the segment representing the difference in height between mouth and eye midpoint. Ratio 5 is the ratio of the segment representing the difference in height between mouth and eye-midpoint, and the segment representing the difference in height between chin and eye-midpoint. Ratio 6 is the height of the eyes within the top and bottom head-margins [41].

In addition to these ratios, we introduce a novel ratio that involves eyebrows and nose. Localization of nose is done according to the technique as depicted in [41]. Here, we need to model eyebrows as a deformable template, which is described by a number of parameters. This deformable template is specified by a set of parameters which enables a priori knowledge about the expected shape of the features to guide the detection process. The size and other parameters of this template are flexible enough to match itself to actual eyebrows in an image. The parameters as obtained from the final iteration are used to describe the eyebrows. This deformable template interacts with the image dynamically. An energy function is defined which contains terms attracting the template to salient features such as edges and valleys in the image intensity and the intensity itself. Gradient Descent,

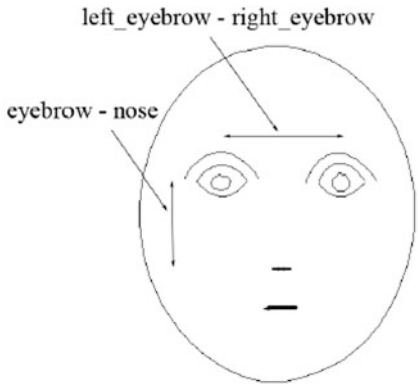


Fig. 4 Our newly proposed ratio

Fig. 5 Eyebrow deformable template. Eyebrow is arbitrarilly centered on a point $\overline{x}$. Area of the eyebrow is bounded by the upper and lower parabola located at a distance d_1 and d_2 from the centre $\overline{x}$, and the right hand side parabola located at a distance a and b from the same centre

a state-of-the-art energy minimization technique is employed to get the best fit with the image. Initial parameters are determined by pre-processing. In each iteration, these parameters are updated that corresponds to altering the position, orientation, size, and other relevant properties of the template.

We propose the Ratio 7 which is the T -ratio formed by two segments - the segment *T5* joining the midpoints of two eyebrows and the segment *T6* between the outer ends of the eyebrows and the nose as shown in Fig. 4.

The Eyebrow Template The deformable templates act on three representations of the image, as well as on the image itself. These representations are chosen to extract properties of the image, such as peaks and valleys in the image intensity and places where the image intensity changes quickly. An additional representation could be added to describe textural properties. An advantage of using these representations is that the templates need only be specified in simple terms. Ability to long range interactions to occur is another advantage of using these representations. These representations do not have to be very precise, and they can be calculated simply. For details, please see [42].

Our present methods involve using morphological filter [43–45] to extract these features. The advantage of these methods for extracting the edge, valley, and peak fields is they yield measures of the strengths of feature in question.

We define an eyebrow template as shown in Fig. 5 in terms of a coordinate system (x, y) positioned at a point (assumed as centre of mass on an eyebrow shaped flat

rigid body). The $\overline{x}$template we defined as follows: (1) The edges at the bottom of eyebrow is represented by P_b a parabola

$$y_e(x) = (a)\left\{1 - \frac{4}{(d1+d2)^2}\left[\overline{x} - \left(\frac{d2-d1}{2}\right)\right]^2\right\} \tag{6}$$

(2) The edge at the top of eyebrow is represented by a parabola P_t

$$y_v(x) = (b)\left\{1 - \frac{4}{(d1+d_2)^2}\left[\overline{x} - \left(\frac{d2-d1}{2}\right)\right]^2\right\} \tag{7}$$

(3) The edges at the inner end (close to nose) of eyebrow is represented by another parabola P_i

$$y_i(x) = (c)\left\{1 - \frac{4}{(a+b)^2}\left[\overline{x} - \left(\frac{b-a}{2}\right)\right]^2\right\} \tag{8}$$

The template depends on 6 parameters $\overline{g} = (\overline{x}, a, b, c, d1, \&d2)$ and its potential.

Energy function E_M is given by

$$E_M = E_e + E_v + E_l + E_m \tag{9}$$

where the edge potential, E_e is computed along the upper, lower and side parabola, is

$$\begin{aligned} E_e = -\tfrac{\beta_1}{|P_b|}\textstyle\int_{P_b} \Phi_e(\overline{x})\, ds - \tfrac{\beta_2}{|P_t|}\textstyle\int_{P_t} \Phi_e(\overline{x})\, ds \\ - \tfrac{\beta_3}{|P_i|}\textstyle\int_{P_i} \Phi_e(\overline{x})\, ds \end{aligned} \tag{10}$$

where Φ_e is edge potential computed earlier using morphological operator.

The valley potential, E_v is specified by the integral over the interior of the eyebrow divided by the area of the eyebrow

$$E_v = -\frac{e_1}{|R_e|}\int_{R_e} \Phi_v(\overline{x})\, dA \tag{11}$$

Φ_v is valley potential computed in a similar manner like edge potential above.

The internal potentials are

$$\begin{aligned} E_l &= \tfrac{k_1}{2}(d1 - d2) \\ E_l &= \tfrac{k_2}{2}\left[\tfrac{c-(a+b)}{(a-b)-(d1-d2)}\right]^2 \end{aligned} \tag{12}$$

The internal parameter, E_l attempts to place the centre of the eyebrow in an arbitrarily middle position while another internal parameter E_m helps prevent the upper boundary getting pulled up to the lower boundary and the nose.

Typical values of the constants are $\beta_1 \approx 400$, $\beta_2 \approx 150$, $\beta_3 \approx 300$, $k_1 \approx 0.2$, $k_2 \approx 0.1$, and $e_1 \approx 1000$.

Updating of these parameters are done by gradient descent.

Tracking Deformable template can be applied for tracking in a straight forward manner where tracking eyebrows are done automatically given an initial position and a set of potential fields. We initialize the template manually for the first time but in subsequent frames we use initial position as the best fit of the preceding frame. This process succeeds only as long as the best fit at time $(t - 1)$ lies in the basin of attraction of the system at time t. A dynamic sequence of eyebrow template on both eyebrows is graphically illustrated in Fig. 6.

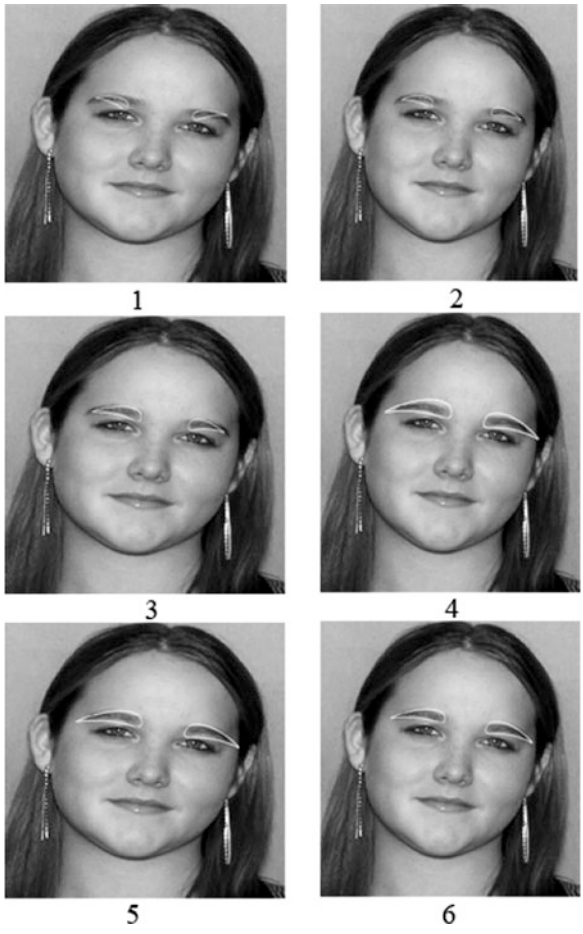


Fig. 6 A dynamic sequence of eyebrow template on both eyebrows. Final state of the template acting on the eyebrows is shown on the right column of the bottom row

3.4 Classification

For model training, an image set X (matrix), where n is the image number and m is the feature dimension, the face estimation expression can be represented as,

$X = [x_1, x_2, x_3, \ \ldots \ , x_n]$, $x_i \ \in \mathcal{R}_m$. The categorical age label for the image x_i is denoted as $l_i \in \{1, -1\}$. The task is to predict the categorical age label of an input image X using binary Classification.

In this work, we use the Support Vector Machine for binary classification.

In this scenario, a hyperplane can be described by

$$w \cdot x + b = 0 \tag{13}$$

where, w is normal to the hyperplane, and $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin.

Now, we create plane H, where

$$H_{ij} = y_i y_j x_i \cdot x_j \tag{14}$$

Find a Lagrange multiplier $\propto$using a QP solver so that

$$\sum\nolimits_{i=1}^{K} \alpha_i - \frac{1}{2} \alpha^T H \alpha \tag{15}$$

Is maximized, satisfying the constraints $\alpha_{1 \geq 0} \forall_i$ and $\sum_{i=0}^{K} \alpha_i y_i = 0$

Now, we calculate

$$w = \sum\nolimits_{i=1}^{K} \alpha_i y_i x_i \tag{16}$$

where x_i and y_i are class label and training vectors respectively. Once feature extraction is done, each face image is represented as an assembly of cropped skin patches, and geometrical constraints expressed as ratios. We use low level features as described above of these skin patches along with geometrical ratios on primary visual cues as features for categorical age estimation. Here, we employ first five ratios as computed by Kwon and Lobo [41] along with our newly proposed ratio 7 as formulated earlier. Doing so, we get a high dimensional feature vector representative of each individual face, which we call *Luminance Invariant & Geometrical Relation based Descriptor (LIGRD)*;

$$\begin{aligned} LIGRD = x_i = \Bigg[\sum_{n=1}^{N} \Big(& C\ hroma_{(r,b)} + dwt_{(d,h,v)} \\ & + lbp_{(r,b)} \Big) + Ratio_{(1,2,3,4,5,7)} \Bigg] \end{aligned} \tag{17}$$

where, $N =$ nos. of skin samples.

Determine the set of Support Vectors S by finding the indices such that $\alpha_i > 0$. Now, b is computed as;

$$b = \frac{1}{N}\left(y_s - \sum_{m \in S} \alpha_m y_m x_m . x_s\right) \tag{18}$$

Each new each feature vector x_i is classified by evaluating

$$y' = \text{sgn}\left(w \cdot x' + b\right) \tag{19}$$

4 Experimental Results

In this particular research, we have outlined a computational framework for categorical age estimation (adult and children) from facial images irrespective of gender. First, the low-level features are extracted from the skin patches of facial images. Next, ratios are computed from the primary cues (eyes, eyebrows, nose, mouth, & chin) as localized by the respective deformable templates. Combining these low levels and high level features forms a powerful descriptor that is able to distinguish between adult and children with maximum accuracy.

Comparison of our experimental results with that of another recently published work reveals our proposed approach yields the highest precision and recall, and overall accuracy in recognition. For quantitative performance comparison, so far, we have not found any existing technique that estimate age in explicitly two categories – adult and children. So, we do not any choice other than comparing our results with a recent relevant work [19], though it is not aimed to categorical age estimation as reported by the authors. Further, the authors have recommended further improvement in their categorical age estimation (baby, adult, & senior) part to make it a reliable categorical age estimation approach.

For quantitative performance comparison, a popular measure, called Precision and Recall (Eqs. 20 and 21), based on True Positive (TP), True Negative, False Positive (FP), and False Negative (FN) is applied on both – our experimental results and the results obtained from Islam et al. [46] to evaluate the classification performance of our novel approach. Apart from the Precision and Recall, we also compared accuracy (Eq. 22) of our approach with the same.

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

Table 4 Facial images taken from different databases for training

Method	Children
FG-NET	100
NCMEC	100
LAG	200

Table 5 Facial images taken from different databases for testing

Method	Adult	Children	Total
FG-NET	50	100	150
NCMEC	0	100	100
LAG	50	100	150

Table 6 Comparison of performance of our proposed method with another existing method

Methods	TP	FP	TN	FN
ICSIPA	270	14	103	13
Proposed	276	9	104	11

Table 7 Comparison of accuracy of our proposed method with other existing method

Methods	Precision (%)	Recall (%)	Accuracy (%)
ICSIPA	95	95	93
Proposed	91	96	95

$$Accuracy = \frac{TP + TN}{P + N} \tag{22}$$

Where, P = Nos. of positive sample and N = Nos. of negative sample.

4.1 Training

We have sourced 400 children face images from different database as shown in Table 4 below [33, 34, 46]. After, different pre-processing steps (denoising, gamma correction, enhancement, and normalization, feature extraction done on these facial images. No negative sample (adult face) has been taken into consideration.

4.2 Testing

The recognition results were evaluated on a dataset composed of 400 images (not included in the training set) from the same datasets shared equally as shown in Table 5. Out of these 400 images 100 adult face images were taken equally from each of the FG-NET & LAG database.

We compared the performance comparison and accuracy of our proposed approach with another existing methods and found the proposed approach outperformed the existing approach significantly as shown in the Tables 6 and 7.

5 Conclusion

We have proposed a novel child face model, where low level features detect child and adult face with maximum accuracy. It can be applied as a dedicated tool for detecting and/or monitoring child pornography while integrated with any existing pornography detection approach. As feature extraction is the key in classification, in this work, we proposed few novel low and high-level feature extraction techniques to achieve maximum accuracy in classification while applied with few existing techniques. To the best of our knowledge, this is the first of its kind which is able to recognize a child and adult face image effectively with highest accuracy and thus paves the way for research in this area to not only help categorical age detection, but also to identify real contents using contextual constraints in the detected skin regions, which may contribute significantly in security & surveillance, human body parts identification, anatomical & dermatological applications and even in marketing – tailoring advertising in shop windows based on the age of shoppers walking past. These alternate uses for the outcomes of this particular work provide extended usefulness and avenues.

References

1. Ropelato J (2011) Internet pornography statistics. http://internet-filter-review. toptenreviews.com/internetpornograph statistics-pg5.html. Retrieved on 15 June, 2011
2. Carr J, Child abuse, child pornography and the internet, internet consultant, NCH, UK. http://www.nch.org.uk
3. Quayle E, Taylor M (2002) Child pornography and the internet: perpetuating a cycle of abuse, deviant behaviour. Interdiscipl J 23:331–361
4. BBC News (2001) International child porn ring smashed. Retrieved on 25 March, 2009, from http://news.bbc.co.uk/1/hi/world/americas/1244457.stm
5. BBCNews (2001) US breaks child cyber-porn ring. Retrieved on 28 March, 2009, from http://news.bbc.co.uk/1/hi/world/americas/1481253.stm
6. BBC News (2001) Child porn raids in 14 countries. Retrieved on 30 March, 2009, from http://news.bbc.co.uk/1/hi/world/americas/1656779.stm
7. Suffern D (2010) Hot post: Facebook's child porn subculture baffles executives, challenges parents. http://www.newsrealblog.com/2010/10/31/facebooks-child-porn-subculture-baffles-executives-challenges-parents-1/
8. Winter J (2010) Exclusive: Facebooks filters fall short in blocking pedophiles. http://www.foxnews.com/scitech/2010/10/21/exclusive-facebook-filters-fall-short-in-blocking-pedophiles/
9. Gannett JS, ABC investigation claims Facebook a tar- get of child predators. http://abcnews.go.com/Technology/story?id=3645840page=1
10. Eendebak P et al (2008) Visual tools for assisting child pornography investigators, TNO Science and Industry, Delft, University of Amsterdam, The Netherlands
11. Shupo A et al Toward efficient detection of child pornography in the network infrastructure. IADIS Int J Comp Sci Inform Syst 1(2):15–31, ISSN: 1646-3692
12. Ibrahim AA (2009) Detecting and preventing the electronic transmission of illicit images, Master thesis, University of Ontario Institute of Technology, Canada

13. Kwon YH, da Vitoria Lobo N (1993) Locating facial features for age classification. Proc SPIE Int Soc Opt Eng 2055:62–72
14. Kwon YH, da Vitoria LN (1999) Age classification from facial images. Comput Vis Image Underst 74(1):1–21
15. Hayashi J et al (2001) A method for estimating and modeling age and gender using facial image processing. In: Seventh international conference on virtual systems and multimedia, pp 439–448
16. Lanitis A et al (2004) Comparing different classifiers for automatic age estimation. IEEE Trans Syst Man Cybern B 34(1):621–628
17. Geng X et al (2006) Learning from facial aging patterns for automatic age estimation. In: Proceedings of ACMMultimedia'06, pp 307–316
18. Yan S et al (2007) Auto-structured regressor from uncertain labels. In: International conference on computer vision
19. Khoa L et al (2009) Age estimation using active appearance models and support vector machine regression. In: Biometrics: theory, applications, and systems, 2009. BTAS '09. IEEE 3rd international conference. Washington, USA, pp 1–5
20. Cootes TF et al (2001) Active appearance models. In: IEEE transactions on pattern analysis and machine intelligence
21. P&G Beauty & Grooming, skin care throughout life
22. Guo Z et al (2010) Rotation invariant texture classification using LBP variance (LBPV) with global matching, Biometrics Research Centre, Department of Computing, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China. Pattern Recogn 43:706–719. Geng X et al (2007) Automatic age estimation based on facial aging patterns. IEEE Trans Pattern Anal Mach Intell 29(12):2234–2240
23. Fu Y, Huang TS (2008) Human age estimation with regression on discriminative aging manifold. IEEE Trans Multimedia 10(4):578–584
24. Guo G et al (2008) Image-based human age estimation by manifold learning and locally adjusted robust regression. IEEE Trans Image Process 17(7):1178–1188
25. Yan S et al (2009) Synchronized submanifold embedding for person independent pose estimation and beyond. IEEE Trans Image Process 18(1):202–210
26. Adult Skin vs. Baby Skin with Diaper Rash. http://www.desitin.com/adult-skin-vs-baby-skin. Retrieved on 10 March, 2009
27. P&G Beauty & Grooming, Skin Care Throughout Life. http://www.pgbeautygroomingscience.com/skin-care-throughoutlife.html 2010
28. Mathworks Documentation, Contrast-limited adaptive histogram equalization (CLAHE), R2011a. Matlab documentation– image processing toolbox. http://mathworks.com/help/toolbox/images/ref/adapthisteq.html
29. Kato Z, Pong TC (2001) markov random field image segmentation model using combined color and texture features. In: Proceedings of international conference on computer analysis of images and patterns, Vol. 2124 of lecture notes in computer science, Springer, pp 547–554
30. Kato Z et al (2003) Unsupervised segmentation of color textured images using a multi-layer mrf model. Proc Int Conf Image Process 1:961–964
31. Kato Z, Pong TC (2006) A markov random field image segmentation model for color textured images. Image Vis Comput 24:1103–1114
32. Islam M (2008) Unsupervised color image segmentation using markov random fields model, Master's thesis, Graduate School of Information and Mathematical Science, UOB, Australia
33. NCMEC, National Center for Missing & Exploited Children (NCMEC). http://www.missingkids.com
34. FG-NET, Face and Gesture Recognition Research Network. http://www.fgnet. rsunit.com/
35. Manjunath BS, Ma WY (2002) Texture features for image retrieval. Wiley
36. Unser M (1995) Texture classification and segmentation using wavelet frames. IEEE Trans Image Process 4(11):1549–1560
37. Ojala T et al (1996) Comparative study of texture measures with classification based on feature distributions. Pattern Recogn 29:51–59

38. Ahonen T, Hadid A, Pietikainen M (2004) Face recognition with local binary patterns. In: 8th European conference on computer vision, pp 469–481
39. Shan C et al (2005) Conditional mutual information based boosting for facial expression recognition. In: Proceedings of the British machine vision conference
40. Adult Skin vs. Baby Skin with Diaper Rash. http://www.desitin.com/adult-skin-vs-baby-skin. Retrieved on 10 March, 2009
41. Kwon YH, Lobo NV (1999) Age classification from facial images. Comput Vis Image Underst 74(1):1–29
42. Yuille AL et al (1992) Feature extraction from faces using deformable templates. Int J Comput Vis 8(2):99–111
43. Maragos P (1986) Tutorial on advances in morphological image processing. Opt Eng 26: 623–632
44. Serra J (2001) Image analysis and mathematical morphology. Academic, New York
45. Hallinan PW (1991) Recognizing human eyes. In: Proceedings of Conference 1570, SPIE, San Diego
46. Islam M et al (2011) Child face detection using age specific luminance invariant geometric descriptor. In: IEEE International Conference on Signal and Image Processing Applications (ICSIPA2011)