FACULTY OF ENGINEERING AND ARCHITECTURE

Generalization of network intrusion detection systems with state-of-the-art deep learning

Alexander Van Nevel
Student number: 01505316

Supervisors: Prof. dr. Bruno Volckaert, Dr. ir. Tim Wauters
Counsellor: Laurens D'hooge

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2020-2021

GHENT UNIVERSITY

Word of thanks

This thesis has helped me to discover more possibilities of machine learning than I previously was aware of. The strict deadlines has helped this thesis to be pushed forward to find solutions based on several studies that are discussed within papers. I want to thank a few people who helped to support me during this period.

I want to thank my counsellor Laurens D'hooge for his hard work, support and feedback every other week; the interesting and pleasant meetings. The insights I got from him and helping me discover a deeper world of machine learning. I also want to thank dr. ir. Tim Wauters and prof. dr. Bruno Volckaert for making this thesis available for me and their flexibility during these difficult times.

Friday 8th January 2020 Alexander Van Nevel

Generalisatie van netwerk intrusie detectie systemen met behulp van state-of-the-art deep learning

Alexander Van Nevel

Supervisor(s): Laurens D'hooge, Tim Wauters, Bruno Volckaert

Abstract— **Met de groeiende vraag naar meer internetdiensten en IoT is beveiliging een steeds groter wordend probleem; steeds meer bedrijven rapporteren aanvallen van hackers. networkintrusiedetectiesystemen worden gebruikt voor het detecteren van een aanval. Aan de hand van machine learning kan een accuraat model worden gemaakt dat binnen een netwerkomgeving een aanval kan detecteren. Alhoewel dit goed werkt binnen dezelfde omgeving is er nog te weinig onderzocht of dit model mogelijk goed zou presteren in een andere netwerkomgeving. In de thesis wordt er onderzocht of het mogelijk is om per aanvalsklasse een model te trainen dat ook overdraagbaar is naar een andere onbekende netwerkomgeving. Als basis hiervoor worden enkele State-Of-The-Art-modellen gereproduceerd vanuit vaak geciteerde papers. Hierna wordt elk model fijn afgestemd om de performantie van het model te maximaliseren. Daarnaast zijn twee bijkomende interessante SoTa-modellen geanalyseerd.**

Keywords—**(Netwerk) intrusiedetectiesysteem ((N)IDS), anomaliedetectie, convolutional neuraal netwerk (CNN), LuNet, recurrent neuraal netwerk (RNN), long short-term memory (LSTM), auto-encodering (AE), generatieve adversarial neuraal netwerk (GAN), deep belief network, particle swarm-optimalisatie, genetische algoritmen, overfitten.**

I. Introductie

HET succes van het steeds groeiende internet heeft gezorgd voor een grotere behoeft naar beveiliging. [1] Hackers hebben steeds betere hulpmiddelen om succesvolle inbraakpogingen uit te voeren en dit is een probleem. [2]

Eén van de methodes om deze aanvallen te detecteren zijn netwerkintrusiedetectiesystemen of NIDS. Een NIDS monitort steeds het in en -uitgaande internetverkeer. Als een NIDS ongewenst gedrag detecteert dan kan de verantwoordelijke systeembeheerder hiervan op de hoogte gebracht worden. Echter zijn de capaciteiten van een NIDS beperkt. Een expertsysteem zal bijvoorbeeld enkel in staat zijn om een aanval te detecteren met enkele vastgelegde regels. Als een aanvaller zijn aanval ontwerpt zodat het expertsysteem omzeilt wordt, dan is zijn aanval succesvol. Een ander nadeel van een statisch ontwerp is dat er geen *zero-day*-aanvallen gedetecteerd kunnen worden, omdat ze simpelweg nog niet opgenomen zijn in het expertsysteem. Als laatste heeft een statisch model erg veel last van vals positieven, met andere woorden het model geeft vaak foute, onnodige meldingen van inbraakpogingen. [3]

Er bestaat echter een alternatief, namelijk een NIDS geproduceerd met Artificiële Intelligentie of AI, meer specifiek Machinaal Leren of ML. Een NIDS-model dat geconstrueerd wordt met deze techniek is in staat om een meer generieke representatie te krijgen van hoe een aanvalspatroon er in de praktijk uitziet. Dit moet ervoor zorgen dat ook zero-day aanvallen gedetecteerd kunnen worden en heeft als doel foutpositieven zoveel mogelijk in te perken. [4]

De huidige staat van dit onderzoek maakt het mogelijk om goede tot zeer goede ML-modellen te produceren die een uitstekende, (gebalanceerde) accuraatheid behalen wat de aanvallen betreft en ook de hoeveelheid foutpositieven kunnen beperken. [4]

Het blijkt dat de performantie van deze modellen overschat wordt wanneer een model naar een andere omgeving gebracht wordt of, in dit geval, wanneer een model een vreemde dataset moet classificeren. Dan blijkt dat de (gebalanceerde) prestaties erg slecht zijn en het model onbruikbaar is, dit voor zowel het detecteren van alle typen aanvallen als aanvallen van één specifieke aanvalsklasse (bijvoorbeeld DDoS-aanvallen).

Dit onderzoek gaat na of dit nog steeds het geval is voor enkele State-of-The-Art of SoTa-technieken binnen deep learning. Dit door de deeplearningmodellen van vaak geciteerde papers te reproduceren en hun prestaties te evalueren. De prestaties van de SoTa-modellen worden erna geëvalueerd op netwerkdata uit een andere omgeving. Het model wordt erna verder geoptimaliseerd om een meer generiek model te verkrijgen dat in de beste omstandigheden goed gebalanceerde prestaties behaalt. [5]

II. IDS met Machinaal Leren research

A. Beslissingsboom

Belissingbomen gebruiken if-then regels om intrusies te detecteren. Het algoritme berekent in elke iteratie het attribuut dat de trainingsdata in even grote groepen kan scheiden, de informatieinhoud. Het attribuut met de meeste informatieinhoud wordt de volgende knoop in de beslissingsboom. Vaak gebruikte algoritmen voor beslissingsbomen zijn C4.5 en CART. [4]

B. Naive Bayes

Naive Bayes classificeert de instanties aan de hand van probabiliteitsregels. Er zijn drie verschillende Naive Bayes algoritmen beschikbaar: (1) Gaussian Naive Bayes voor continue attributen, verspreid volgens een Gausscurve. (2) Bernoulli Naive Bayes, een binomiaal model. Deze gebruikt binaire *feature*-vectoren voor bijvoorbeeld het *Bag-of-Words*-model. En als laatste is er (3) Multinomiaal Naive Bayes, dat gebruikt wordt voor discrete waarden met feature-vectoren die de frequenties van eventen representeert. [6] [7]

C. Support Vector Machine

Een Support Vector Machine of SVM probeert een hypervlak te leggen tussen intrusie-instanties en normale instanties. Soms is het nodig om de data-instanties te mappen naar een hogere dimensie wanneer het niet mogelijk is om een scheidend hypervlak te bepalen. [7]

D. K-means Clustering

K-means is een clusteringalgoritme. Het algoritme baseert zich op het idee dat instrusienetwerkdata verder gelegen moeten zijn van normale netwerk data dan dat normale netwerkinstanties van elkaar zouden liggen. In de eerste stap worden er K clusters bepaald aan de hand van de data-instanties. Erna worden de overige instanties aan clusters toegewezen, afhankelijk van hun afstand tot die clusters. In de volgende iteraties worden de clusters herberekend als het gemiddelde van de datapunten die tot de cluster behoren. eventueel komen zo nieuwe punten in deze cluster. Deze iteraties gaan door tot een stopcriterium bereikt is.

Dit algoritme kan gebruikt worden om enerzijds de uitschieters te scheiden van de trainingsdata en anderzijds normaal netwerkverkeer te scheiden van intrusies. Het algoritme heeft echter een keerzijde: indien de intrusie instanties zelf alle clusters vormen dan is dit algoritme niet bruikbaar. [7]

E. Genetische Algoritmen

Genetische Algoritmen of GA's maken gebruik van adaptieve zoekmethodes, gebaseerd op natuurlijke selectie en genetica (survival of the fittest). Het algoritme gaat opzoek naar de parameters die een vooraf gedefineerde fitnessfunctie maximaliseren. Een nadeel van deze techniek is dat er geen zekerheid is om een globaal optimum te vinden. Er zal dus enkel een lokaal optimum gevonden worden. Daarnaast is het complex om de ruimte waarin een parameter zich kan bevinden, te definiëren. [7]

F. K-Nearest Neighbor

K-Nearest Neighbor of K-NN is een techniek die een vooraf bepaald aantal zwaartepunten plaatst tussen de instanties. Het algoritme probeert steeds zoveel mogelijk datapunten samen te nemen om het gemiddelde van deze datapunten te bepalen. Het resultaat hiervan zijn k zwaartepunten. Het algoritme gaat er dus vanuit dat instanties vergelijkbaar zijn als de parameters ervan dichter bij elkaar gelegen zijn. [7]

G. Fuzzy Logic

Fuzzy Logic of FL is een methode die het menselijk redeneren tracht na te bootsen. Het idee van het algoritme is om een binaire scheiding te maken tussen instanties. FL vereist meer werk om een goed model te maken en af te stemmen dan andere methoden. [8]

H. Hidden Markov Model

Een Hidden Markov Model of HMM produceert een model van transities en acties dat het gedrag nagaat. Tussen de transities binnen een HMM-model is een probabiliteit berekend. Het model identificeert intrusies wanneer deze overgaat naar een staat die een intrusie aangeeft. HMM's zijn vatbaar voor overfitting. Verder gebruiken HMM's het Viterbi-algoritme om de probabiliteit tussen de staten te bepalen en dus om het model te produceren. [7]

I. Swarm Intelligence

Swarm Intelligence of SI maakt gebruik van een groep samenwerkende agenten. Deze agenten werken samen om een taak te vervullen en om een doel te bereiken. SI-algoritmen zijn lerende algoritmen die gebaseerd zijn op de evolutietheorie. De agenten bevatten een set classificatieregels om intrusies te detecteren. Verder zijn SI-algoritmen eenvoudig aanpasbaar, ze passen zich namelijk gemakkelijk aan aan nieuwe stimuli. [7]

J. Ensemble-Leren

Ensemble-leren combineert verschillende soorten classifiers om de uiteindelijke resultaten van de verschillende classifiers te combineren. Een voordeel van ensemble-leren is dat algoritmes minder vatbaar zijn voor overfitting. [7]

K. Artificieel Neuraal Netwerk

Een Artificieel Neuraal Netwerk of ANN is een reeks lagen die elkaar sequentieel opvolgen met steeds een welbepaald aantal neuronen in elke laag. In een laag is elk neuron verbonden met alle neuronen van de vorige laag en met alle neuronen van de opvolgende laag. Dit is de meest eenvoudige representatie van een neuraal netwerk. [9]

L. Stacked Auto-encoder

Een stacked auto-encoder of SAE is een neuraal netwerk dat wordt gebruikt om input-features te comprimeren. Dit is een vorm van dimensiereductie. Een SAE wordt voorgesteld als een reeks gestapelde auto-encoders. Het comprimeren ervan is steeds met verlies. Met een auto-encoder is het verder mogelijk om de input-features eventueel te reconstrueren in de laatste laag. [10]

M. Convolutional Neuraal Netwerk

Een Convolutional Neuraal Netwerk of CNN is een variant van een ANN. Een CNN bevat convolutielagen die als doel hebben om de ruimtelijke eigenschappen van inputs te extraheren. Een dergelijke convolutielaag maakt gebruik van een filter die de data zal onderbemonsteren om een compactere representatie te verkrijgen van de input-features. De laatste laag in een CNN is een volledig geconnecteerde laag waarbij alle neuronen binnen de laag verbonden zijn met elkaar. Het doel van een volledig geconnecteerde laag is om de geëxtraheerde features te combineren naar betekenisvolle features. Deze extra eigenschappen van een CNN komen ten koste van trainingstijd. [11]

N. Recurrent Neuraal Netwerk

Een Recurrent Neuraal Netwerk of RNN is gebaseerd op menselijke kennis uit het verleden. Deze unieke eigenschap van een RNN wordt bekomen door een cyclische structuur van de neuronen. Deze cyclische structuur wordt gebruikt om sequentiegebonden features te interpreteren in de data. [12]

O. Long Short Term Memory Recurrent Neural Network

Een Long-Short-Term-Memory-laag of LSTM-laag bouwt verder op de eigenschappen van een RNN. De belangrijkste, vernieuwende eigenschap is de geheugencel. Een geheugencel

laat toe om gehele sequenties van data in beschouwing te nemen. Verder lost het ook een probleem op binnen het trainen van een RNN. Bij het feedbackproces, waarbij de fout wordt doorgegeven aan de verborgen lagen, verdwijnt het nut steeds meer des te dieper de fout terugkeert in het RNN. [12]

P. Deep Belief Networks met Restricted Boltzmann Machine

Een Deep-Belief-Netwerk met Restricted Boltzmann Machine of een DBN-RBM is een gestapeld, ondiep, neuraal netwerk. Een ondiep, neuraal netwerk bestaat uit twee lagen waarbij elk neuron uit de eerste laag is verbonden met de neuronen uit de andere laag. Het wordt een Restricted Boltzmann Machine genoemd omdat er geen enkel neuron uit de ene laag is die een connectie deelt. De RBM wordt getraind om de inputdata te reconstrueren, net zoals bij de auto-encoder. Het verschil met een RBM is het gebruik van een stochastische benadering, bijvoorbeeld een Gauss distributie, tegenover een auto-encoder die deterministisch werkt. Het Deep Belief Netwerk of DBN-gedeelte ervan wijst erop dat de Boltzmann Machines op elkaar gestapeld worden, als een sequentie van opeenvolgende Boltzmann Machines. Een DBN-RBM kan gebruikt worden bij zowel gelabelde data als niet-gelabelde data. Bij gelabelde data kan deze techniek gebruikt worden voor classificatie en bij niet-gelabelde data kan een RBN-DBN gebruikt worden voor feature-extractie. [13]

Q. Generative Adversarial Networks

Generatieve Adversarial Netwerken of GAN's zijn neurale netwerken die hoogdimensionale data die nog niet gepreprocest is, modelleert. De methode maakt gebruik van een generatief, neuraal netwerk, een discriminator, neuraal netwerk en een black-box IDS. De bedoeling van het generatief, neuraal netwerk is het genereren van nieuwe data die een black-box IDS niet correct zal kunnen classificeren. Deze moet helpen bij het creëren van een robuuster black-box-IDS-model dat ook de afgeleide data correct zal kunnen classificeren. [14] [15]

III. Beschikbare Datasets

Alhoewel er wel enkele datasets beschikbaar zijn, zijn deze datasets niet altijd betrouwbaar. Een voorbeeld hiervan is de KDD 1999 Dataset. Deze dataset is een vaakgebruikte dataset binnen het classificeren van intrusies. De dataset is echter meer dan 20 jaar oud waardoor deze dataset 20 jaar aan evoluties binnen het internet mist.

Er werd gekozen om gebruik te maken van de CICIDS2017 dataset. Het is een van de meest recentste datasets die beschikbaar is, al heeft deze ook zijn nadelen. Het opvallendste nadeel is dat de dataset een IDS zou bevoordelen tijdens het trainen. [16] Daarnaast heeft de dataset in vergelijking met andere datasets meer types bedreigingen in de dataset zitten. [16]

IV. Pre-processing

Datasets hebben nood aan preprocessing. Een dataset die niet opgekuist is, kan attributen missen, uitspringers bevatten of ongebalanceerd zijn. Het doel van preprocessing in deze context, is om de specifieke eigenschappen die een netwerkomgeving in een dataset heeft gebracht, weg te werken. Enerzijds wordt dit gedaan door attributen, zoals IP-adres van bron en bestemming te verwijderen. Maar ook door de waarde-attributen te normaliseren. Dit heeft als doel de attributen binnen eenzelfde interval te brengen.

V. Analyse van geselecteerde SoTa-modellen

In deze sectie worden de SoTa modellen geselecteerd uit enkele, vaak geciteerde papers. Elk model wordt zo goed als mogelijk gereproduceerd. Indien er toch verschillen zijn met de originele methoden, dan worden ze vermeld.

A. Reproductiemodel 1: SVM

Dit model is gebaseerd op de paper "Shallow and Deep Network Intrusion Detection System: A Taxonomy and Survey", meer specifiek de sectie over SVM's. De besproken SVM in de paper werd als een betere classifier gezien dan een modern ANN, omdat de reden dat deze classifier een kortere trainingstijd had in combinatie met iets betere resultaten. [4]

B. Reproductiemodel 2: Beperkt RNN

Deze paper maakt gebruik van RNN's. RNN's zijn zinvol voor intrusiedetectie aangezien ze in staat zijn om patronen te zien in opeenvolgende sequenties van data.

Verder bespreekt de paper in detail hoe de architectuur van hun ontwerp eruit ziet. Echter moeten er hieraan enkele aanpassingen gebeuren. De paper maakt gebruik van een ander labelsysteem. Er wordt namelijk onderscheid gemaakt tussen meer dan twee klassen. [17]

C. Reproductiemodel 3: SAE met RNN

Dit neuraal netwerk begint met een gestapelde auto-encoder waarbij in elke laag de input verder gecomprimeerd wordt. Hierna volgt een LSTM-laag die zal zorgen voor het extraheren van de sequentiële features. [12]

D. Reproductiemodel 4: Gecombineerd CNN en RNN

Dit model maakt gebruik van een meer specifieke architectuur, genaamd LuNet. LuNet maakt gebruik van zowel CNN als RNN, met als doel zowel de ruimtelijke features als features uit sequenties van data te halen. LuNet biedt in tegenstelling tot de andere deeplearning-technieken een hoger detectiepercentage van de aanvallen en minder foutpositieven. [18]

VI. Generalisatie van geselecteerde SoTa modellen

A. Reproductiemodel 1: SVM

Aan dit model werd verder geen tijd meer besteed. Het doel van dit model was om een werkende omgeving op te zetten.

B. Reproductiemodel 2: Beperkt RNN

Dit SoTa-model werd niet verder onderzocht. Uit onderzoek is gebleken dat de resultaten bevoordeeld waren door identificerende attributen, namelijk de IP-adressen. De IP-adressen werden gebruikt om de aanvallen te classificeren. Dit gedrag is aan te tonen door de IP-adressen uit de dataset te halen

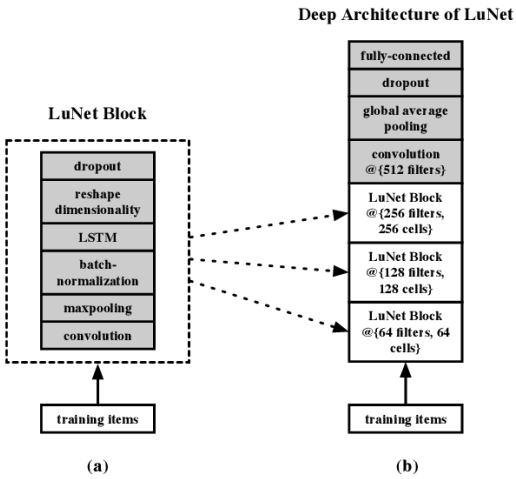

Fig. 1. LuNet architecture

en het model opnieuw te trainen. Hieruit bleek dat het model slecht presteerde zonder deze IP-adressen.

C. Reproductiemodel 3: SAE met RNN

Reproductie van SoTa-model 2 was succesvol, hierop kan verder worden gegeneraliseerd.

- Een dropout van 0,5 toevoegen na elke verborgen laag. [5]
- Training laten starten vanaf elke dataset.
- Training starten vanaf de meest accurate dataset.
- Training starten vanaf de meest accurate dataset, waarbij één dataset wordt achtergehouden ter evaluatie.
- Train het model op één aanvalsklasse.

D. Reproductiemodel 4: Gecombineerd CNN en RNN

Reproductie van SoTa-model 3 was ook succesvol. In dit SoTa-model is meer gefocust op regularisatietechnieken. De reden hiervoor is dat het vorige reproductiemodel goed presteerde met eender welke trainingsmethode, en dus minder impact had op de resultaten dan voorheen gedacht werd.

VII. Resultaten

De resultaten worden gemeten door de gebalanceerde accuraatheid en de F1-score te bepalen. In gebalanceerde accuraatheid wordt aan beide klassen hetzelfde gewicht gegeven, 50% van de score uit het aantal correct geclassificeerde aanvallen en 50% uit het normale netwerkverkeer.

De F1-score geeft inzicht in de verhouding van foutpositieven tegenover het totaal aantal correct geclassificeerde instanties samen met het aantal correct geclassificeerde aanvallen tegenover het totaal aantal geclassificeerde aanvallen in één enkele metriek.

De resultaten zijn bekomen door gebruik te maken van de CICIDS2017 datasets.

A. Resultaten model 1: SVM

Tabel I bevat de resultaten van dit model. De resultaten van dit model zijn minder van belang aangezien dit model gebruikt werd om de omgeving op te zetten en te testen om reproductie van de andere modellen zo vlot mogelijk te laten verlopen.

	Origineel	Reproductie
Acc. Gebalanc.	82,45%	62,79%
F1-score	-	63,64%

TABLE I
Resultaten SVM

B. Resultaten model 2: Beperkt RNN

Reproductie van dit model was initieel succesvol, zie Tabel II. Na verder onderzoek was de hoge accuraatheid te wijten aan het belang dat werd gehecht aan de IP-adressen. Na het wegnemen van de IP-adressen zakte de accuraatheid. Classificatie was praktisch willekeurig. Om die reden kon dit model niet verder gebruikt worden.

	Origineel	Reproductie	Gegeneraliseerd
Gebalanc. acc.	84,83%	97,53%	-

TABLE II
Resultaten SoTa-1

C. Resultaten model 3: SAE met RNN

Dit model heeft uitstekende prestaties, zowel bij reproductie als bij generalisatie. Echter bleek dat dit model niet goed generaliseerde wanneer het een dataset moest classificeren waarop nog niet getraind werd. Het model toont hetzelfde gedrag voor training op zowel elke aanvalsklasse als training op één aanvalsklasse, namelijk een gebalanceerde accuraatheid van 50%. Tabel III toont de resultaten hiervan, de generalisatieprestaties zijn de prestaties na training op alle datasets en het de toegevoegde dropout lagen.

	Origineel	Reproductie	Gegeneraliseerd
Gebalanc. acc.	92%	97,98%	98,46%
F1-score	-	92,64%	94,52%

TABLE III
Resultaten SoTa-2

Nadien werd een laatste poging gedaan door het model op één aanvalsklasse te trainen. Dit werd gedaan door op de 2017 DDoS dataset, 2018 DDoS Dataset en op een fractie (10%) van de 2019 DDoS dataset te trainen. De overige 90% van de 2019 DDoS dataset werd dan gebruikt om de prestaties te evalueren. Dit was de laatste poging en bleek geen succes te zijn. Tabel IV toont de resultaten.

Hieruit volgt de conclusie: een relatief klein model kan enkel goede resultaten behalen wanneer deze op voldoende data heeft kunnen trainen uit de omgeving waarin het model geëvalueerd zal worden.

D. Resultaten model 4: Combined CNN en RNN

Om verder te bouwen op de resultaten die werden bekomen uit model 3 werd dit model gekozen door eenzelfde technologie

	Generalization Results 2
Balanced accuracy	50,25%
F1-score	83,25%

TABLE IV
SoTa-2 trainen op een fractie van een dataset

te selecteren, maar met een andere aanpak.

Dit model maakt gebruik van sequentiële LuNet-blokken, zoals beschreven in de analyse van model 4. Deze LuNet-blokken zijn echter erg groot. Elk blok wordt opgevolgd door een groter blok. Alhoewel de initiële resultaten goed waren, zorgde de omvang van het neuraal netwerk voor overfitting. Het model kon bij de ene training goede resultaten tonen, wanneer erna op dezelfde manier getraind werd, kon deze dan slecht presteren. Dit gedrag werd uitvergroot wanneer dropout-lagen werden toegevoegd en er getraind werd op één aanvalsklasse met drie dataset (2017, 2018 en 2019) DDoS.

	Origineel	Reproductie	Gegeneraliseerd
Gebalanc. acc.	82,78%	97,53%	50,0%
F1-score	-	90,8%	32,83%

TABLE V
Results SoTa-3

VIII. Conclusie

Dit onderzoek toont aan dat de keuze van het model weinig invloed heeft op de gegeneraliseerde prestaties. Daarnaast werd aangetoond dat zelfs een robuust model met goede prestaties de prestaties overschat over andere verschillende datasets. De netwerkomgeving heeft een grotere invloed dan eerder gedacht werd.

IX. Gerelateerd werk

Deze sectie vermeld twee andere technieken die ook onderzocht werden maar niet gereproduceerd werden.

A. DBN met RBM

De analyse is gebaseerd op de paper "An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network" [13]. Deze paper stelt een deep neuraal netwerk voor dat bestaat uit meerdere Restricted Boltzmann Machines of RBM's en een Back Propagation Neuraal Netwerk-of BPNN. Dit model wordt in meer detail uitgelegd in Deel II, Restricted Boltzmann machines. Dit model wordt verder geoptimaliseerd met genetische algoritmen, meer bepaald door een particle-swarm-algoritme uitgelegd in Deel II, genetische algoritmen. Het particle-swarm-algoritme gaat de parameters van het neuraal netwerk optimaliseren.

B. GAN

GAN's worden uitgelegd in Deel II, Generatieve Adversarial Neurale Netwerken. Dit model gaat een reeds bestaand black-box IDS verbeteren door netwerkdata te genereren waarvan het weet dat de black-box IDS het niet zal opmerken. Het doel van deze techniek is om de dataset verder aan te vullen zodat een IDS zich beter kan wapenen. Onderzoek naar deze techniek is echter beperkt. De prestatieclaims van de paper moeten dus met een korrel zout genomen worden. [15]

Bibliography

[1] A. H. L. Iman Sharafaldin and A. A. Ghorbani, "Towards Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," pp. 108–116, 2018.

[2] "Microsoft digital defense report cyber threats." [Online]. Available: https://blogs.microsoft.com/on-the-issues/2020/09/29/microsoft-digital-defense-report-cyber-threats/

[3] "Intrusion Detection Systems definition." [Online]. Available: https://www.barracuda.com/glossary/intrusion-detection-system

[4] A. H. C. T. Elike Hodo, Xavier Bellekens and R. Atikinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey," 2017.

[5] J. H. Fahimeh Farahnakian, "A Deep Auto-Encoder based Approach for Intrusion Detection System," p. 178, 2018.

[6] "Bag of Words example." [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[7] U. T. E. S. P. Preeti Mishra, Vijay Varadharajan, "A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection," pp. 686–728, 2018.

[8] "Fuzzy Logic." [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence

[9] "Simple introduction to Convolutional Neural Networks." [Online]. Available: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

[10] "Stacked Autoencoder." [Online]. Available: https://medium.com/@venkatakrishna.jonnalagadda/sparse-stacked-and-variational-autoencoder-efe5bfe73b64

[11] "CNN." [Online]. Available: https://www.geeksforgeeks.org/difference-between-ann-cnn-and-rnn/

[12] J. W. Y. L. L. C. Yu Yan, Lin Qi, "A Network Intrusion Detection Method Based on Stacked Autoencoder and LSTM," 2020.

[13] Z. Z. T. H. Z. L. Peng Wei, Yufeng Li and D. Liu, "An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network," 2019.

[14] B. L. G. M. V. R. C. Houssam Zenati, Chuan-Sheng Foo, "Efficient GAN-Based Anomaly Detection," pp. 1–7, 2018.

[15] S. L. J. Q. A.-A.-F. Muhammed Usama, Muhammed Asim, "Generative Adversarial Networks For Launching and Thwarting Adversarial Attacks on Network Intrusion Detection Systems," pp. 78–344, 2019.

[16] S. B. Ranjit Panigrahi1, "A detailed analysis of

CICIDS2017 dataset for designing Intrusion Detection Systems," pp. 479–482, 2018.
[17] K. R. Sara A. Althubiti, Eric Marcell Jones Jr., "LSTM for Anomaly-Based Network Intrusion Detection," 2018.
[18] P. Wu and H. Guo, "LuNet: A Deep Neural Network for Network Intrusion Detection," 2019.

Contents

List of Tables

List of Figures

1

Introduction

The popularity and the rapid advancement of the Internet has made network security a growing problem. [2] More than ever are companies being compromised by malicious hackers and forced to pay a ransom. Hackers are also using more sophistication techniques over the past few year, techniques that make intrusions harder to detect.

To resist these attacks, cybersecurity experts have to develop more sophisticated techniques to mitigate these attacks. However, no method is perfect and hackers will always be able to compromise the security system in some way. For that reason, security is a never ending process of developing a security system, finding a way to compromise the security system and then again, patching the vulnerabilities in the security system.

The introduction discusses security techniques of a specific category of security systems, namely Network Intrusion Detection Systems or NIDS; since new AI techniques for NIDS has shown its potential and because these techniques are very new, research in this field has not been completed.

1.1 Security systems

Security systems help to defend the infrastructure for potential attacks. There exist three main different security solutions to protect infrastructure: Firewalls, Antivirus software and Intrusion Detection Systems or IDSs. Each technique has its specific role and place in the infrastructure.

Firewalls

Firewalls guard traffic at a computer its entry point or ports, which is where information is exchanged with external devices. For example, ”Source address 172.18.1.1 is allowed to reach destination 172.18.2.1 over port 22.”

Think of IP addresses as houses, and port numbers as rooms within the house. Only trusted people (source addresses) are allowed to enter the house (destination address) at all—then it is further filtered so that people within the house are only allowed to access certain rooms (destination ports), depending on if they are the owner, a child, or a guest. The owner is allowed to any room (any port), while children and guests are allowed into a certain set of rooms (specific ports). [3]

Antivirus Software

Antivirus programs are designed to evaluate data such as web pages, files, software, and applications to help find and eliminate malware as soon as possible.

Also, most of the existing antivirus programs provide real-time protection, which protects devices from incoming threats; scan the entire system regularly for known threats and provide automatic updates. They identify, block and delete malicious code and software.

Antivirus software works by checking programs and files against a database of known types of malware. Since new viruses are constantly created and distributed by hackers, it will also scan computers for the possibility of new or unknown type of malware threats. [4]

Intrusion Detection Systems

Intrusion detection systems are devices or software applications that monitor a network for malicious activity or policy violations. Any malicious activity or violation is typically reported or collected centrally using a security information and event management system. Some IDS

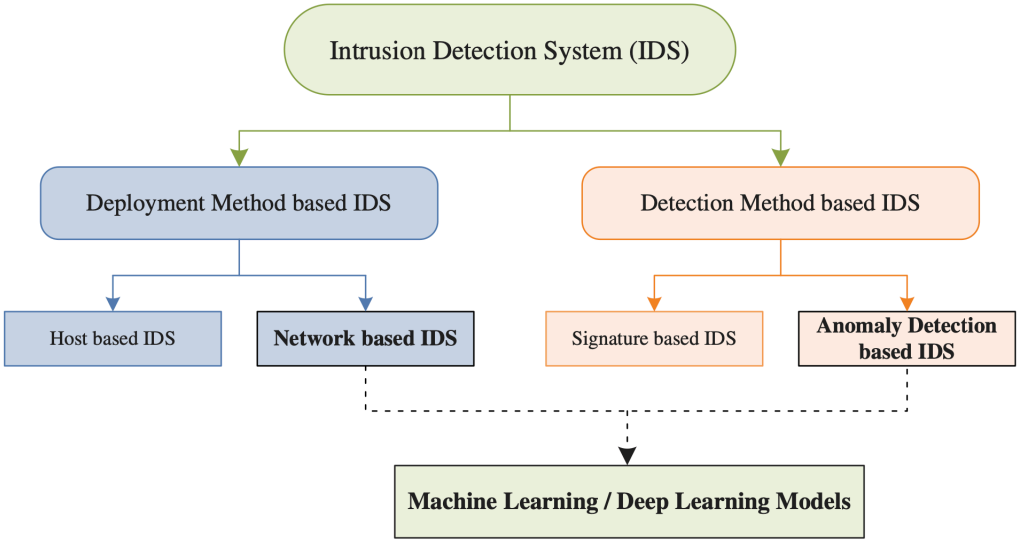

Figure 1.1: Intrusion detection system classification taxonomy

are capable of responding to detected intrusion upon discovery. These are classified as Intrusion Prevention Systems or IPS. [5]

1.2 Intrusion Detection Systems in detail

Intrusion detection systems exists in two types: host intrusion detection systems and network intrusion detection systems. A Host Based Intrusion Detection System or HIDS monitors an individual host or device and sends notifications to the user if suspicious activity has been detected (system files being modified or deleted, a suspicious system call, ...). A Network Intrusion Detection System or NIDS is generally located at a strategic location, such as at a gateway or router; it will monitor in real-time for intrusion attempts. IDS can be implemented in three different ways: misuse detection, anomaly detection and hybrid detection. [6]

1.2.1 Misuse detection

In misuse detection, the IDS uses a set of rules to identify attack types. Misuse detection can be classified into a knowledge based system and a machine learning system. In a knowledge-based system, network data is compared to predefined rules or attack patterns, where machine learning based IDS tries to discover the attack classes by itself.

A machine learning model tries to distinguish between the attack classes. The purpose of a

machine learning based IDS is to provide a general representation and distinction of the attack classes. This is where misuse detection fails. Misuse detection is not able to detect unknown attacks and they also need regular updates to detect the newer attacks.

1.2.2 Anomaly detection

Anomaly detection based IDS tries to distinguish between normal and abnormal internet traffic. It is based on the hypothesis that the network traffic of the average user must be different from that of a attacker. Anomaly detection models the normal use of the system and updates this model over time. For example, each network connection is identified by a set of parameters, the behavior of those parameters are monitored over time, any abnormal deviation is marked as anomalous. Anomaly detection can be implemented with statistical data, machine learning and Finite State Machines or FSMs.

Finite state machine

A FSM produces a model of behavior, consisting of transistions, states and actions. Network traffic that deviates from the expected state is marked as ”anomalous”.

Machine learning based IDS

A unique property of machine learning in IDS is that it is also able to detect zero-day attacks. [1] In practice, it appears that these types of anomaly detection suffer from a large number of false positives. The reason for these false positives is that it is not always able to distinguish between a attack and normal network traffic data. [7]

1.2.3 Hybrid detection

Hybrid detection based IDS combines anomaly detection and misuse detection to detect attacks.

Further, machine learning for IDS can catch some of the Achilles heels of a signature based IDS: A signature based IDS can be easily bypassed by making a tiny adjustment to the attack pattern. [7]

[1] A zero-day attack is a exploit of a vulnerability that was not yet known by the developers.

Other machine learning advantages include:

- CPU load is low to moderate as not every signature needs to be analyzed.
- Machine learning can detect new or unknown attack types.
- Machine learning can detect the complex characteristics of attack behavior, this provides faster and higher detection accuracy.
- Signature-based IDS needs regular updates to detect the newer attacks as well. Machine learning techniques do not need such updates.

1.3 Problem definition and goal of this master's thesis

Many successful machine learning based IDSs have already been developed since datasets for network traffic data have been created. [6]

However, this work will show that these IDS models are not performing well in other environments but the environment it was trained and evaluated in. Another problem that was discovered in the literature study is that many researchers are using an older dataset, namely the KDD Cup 1999 or a derived version of this dataset. When researchers are using this dataset, they are missing out on 20 years of innovation or changes. One of the reasons for this, is that there are not many network traffic datasets available. This can be a problem since the Internet changed significantly over the past 20 years. Attacks are more sophisticated, a new category of mobile devices have emerged and newer internet protocols have been developed. It is already very hard for the newest datasets to keep up with the fast changing and growing internet.

The main goal of this master's thesis is to focus on the generalizability of machine learning methods, specifically that of deep neural networks. Five recent Machine Learning IDS or ML-IDS proposals. [8] [9] [10] [11] that rely on neural architectures have been selected. Three of which have been reproduced and verified, before evaluating them under more realistic test circumstances and two other ML-IDS proposals are theoretically analysed but not reproduced.

Well-adopted academic papers

The main goal in these choices is to consider the most popular, new techniques. Currently there are a few new State-of-The-art or SoTa techniques that have been drawing the attention of many researchers. It is important to consider the most popular ones in this thesis, most of the choices are made based upon the recommendations by different papers [12] [7]. As listing of these popular methods:

- Long Short Term Memory Recurrent Neural Network
- Stacked Auto-encoders
- Convolutional Neural Network
- Deep Belief Networks with restricted Boltzmann Machine
- Generative Adversarial Network

Proposed solution

Before building a model, each dataset is preprocessed within a pipeline. This pipeline has several stages with the purpose to streamline the datasets. This is done to overcome the deviations that a specific computer network environment could bring into the dataset. Secondly, each model is evaluated for consistency and tested in more challenging, realistic circumstances. The model is evaluated by its ability to differentiate normal traffic data from attack data. Thirdly, the models are trained by the attack class. Meaning that a dataset will only contain one certain attack class. When the reproduced model is able to match the performance of the paper its model, the parameters are further tweaked in the generalization process to enhance the performance of the model. Lastly, the model its performance is then evaluated with a different dataset. This different dataset must contain the same attack class as the dataset that was used to train the model. Another strategy to enhance the capabilities of the model would be to merge different datasets from the same attack class, but leave one out to evaluate the generalizability performance for one specific attack class.

2

Literature review

This section forms the base structure of this thesis. It concludes what is learned during the literature study from papers that contain knowledge on the current state of intrusion detection with machine learning. This section paves the way of the research that is done within the thesis.

2.1 Machine Learning in IDS research

Various but popular machine learning techniques for IDS are discussed. These techniques have different characteristics and result in different intrusion detection mechanisms. [12]

Disclaimer: the purpose of this section is not to compare classifiers to each other. All of these classifiers are able to perform adequate for anomaly detection (when implemented correctly). The main purpose of this section is to analyse the classifiers their advantages and disadvantages. A detailed performance analysis can be found in [6].

2.1.1 Decision Tree

Decision tree learning algorithms use a binary tree to illustrate every possible outcome of a decision. The learned trees are then represented in the form of if-then rules. There are many different algorithms to generate a tree, common algorithms are the C4.5 and CART algorithm. [6] These algorithms choose in each iteration the attribute with the highest entropy. A high entropy means that its attribute is able to separate the training data very well. The ability to separate training data can be calculated, which is called the information gain of the attribute. The root node should have the attribute with the highest information gain. A decision tree can be used for problems where (a) Instances can be represented in attribute-value pairs (where each attribute can have a disjoint set of possible values). (b) The target function should have a discrete output value (for example yes or no). (c) Decision trees are robust to errors in the training data. (d) Training data may contain missing value attributes. [12]

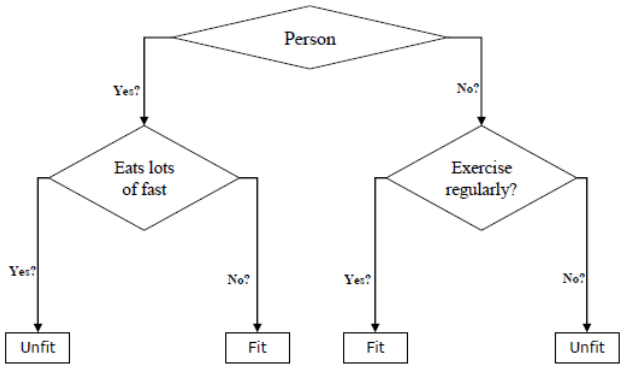

Figure 2.1: Decision Tree example

2.1.2 Naive Bayes

Naive Bayes is a probabilistic classifier. There are three Naive Bayes algorithms: Gaussian Naive Bayes, used for continuous data values which are distributed according to a Gaussian distribution, Bernoulli Naive Bayes, a binomial model used for binary feature vectors such as the Bag of words model [1] and Multinomial Naive Bayes, which is used for discrete values in which feature vectors represent the frequency in which certain events occur. Naive Bayes classifiers achieve high detection speeds and are less complex than other classifiers. However, it makes an assumption that features are independent of each other, which is not true in detecting various types of attack. Notice, Hidden Naive Bayes is a extension of Naive Bayes and relaxes the assumption that all features are independent. [12]

[1] A bag-of-words model is a technique for extracting features from text for use in machine learning algorithms. Binary feature vectors in this example are vectors with the occurrences of words in a sentence. [13]

2.1.3 Support Vector Machine

Support Vector Machine or SVM tries to form a separation plane that separates the different classes. The generalization error can be reduced by maximizing the margin of this plane and preferably create the largest possible distance between the separating hyperplane and instances on either side of the plane. The data points that lie on the margin of optimum separating hyperplane are known as support vector points and the solution is represented as a linear combination of these points. If the data contains misclassified instances, the SVM may not be able to find separating hyperplanes. A solution for this problem is to map the data to a high dimensional data space or feature space; the kernel function is used to map the data into a new feature space for classification. SVM algorithms can be categorized into two types, based on the kernel function. In a linear SVM algorithm, the linear kernel function separates the training data if the training data is linearly separatable. For non-linear data, a non-linear kernel function tries to map the input data to a higher dimensional feature space that has a linear plane. Based on the detection type, an SVM can be categorized into two types: One-class SVM and Multi-class SVM. The performance of a SVM is dependent on the chosen kernel function. The training time of a SVM classifier is very high. Unlike in IDS, where it is not desirable to have a high training time since retraining IDS is very common because the user his behavior changes frequently. [12]

2.1.4 Genetic algorithms

Genetic Algorithms or GAs are an adaptive searching method, using principles of natural selection and genetics. [6] It uses four operators: initialization, selection, crossover and mutation. GA tries to search to the highest quality population of individuals starting from the arbitrary selected initial population. The quality of the genes are measured with a fitness function. In each iteration or generation, the three operators selection, crossover and mutation are sequentially applied to each generation. Only the fittest genes can survive and reproduce. GA performs well when applied with other classifiers to optimize the parameters of the classifier. However, GAs have no assurance that a global optimum is found. Moreover, representing a problem space in the GA is complex. GAs need a large number of fitness function evolution. [12]

2.1.5 K-means Clustering

A K-means algorithm is a clustering based anomaly detection algorithm. It is based on the assumption that normal data instances lie closer to their closest centroid while anomalies lie far away from their closest cluster centroid. At first, the data is clustered into K clusters assuming K data points as the centroid of different clusters. The other data points are assigned to a cluster, based on their distance from the clusters. In the next step, the location of the centroids

are recalculated as the average of all the closest data points of the cluster, this is done for each cluster. This process is repeated until some stopping criteria is reached. Some researchers are using this algorithm to separate anomaly data from normal data, while other researchers are using this to separate outliers from the training data to provide the refined training dataset for the classifier. [12] Both cases have improved the detection rate. It is important to note that this technique will fail if the anomalies in the data form the clusters by themselves and thus no intrusions will be detected. [12]

2.1.6 K-Nearest Neighbor

K-Nearest Neighbor or K-NN is a technique for sample classification. It is non-parametric, which means that the technique does not make any assumptions on the distribution of the data points. [6] The technique is classified under the lazy learning algorithms since lazy learning algorithms delay the generalization until the classification is performed. This makes the training phase of K-NN much faster than other methods but classification takes more computational time. K-NN makes the assumption that similar data points would lie more close to each other than data points that are not similar. Another assumption that K-NN makes, is that normal data points occur in dense neighborhoods while anomalies occur far from their closest neighbor. K-NN algorithms exist in two separate categories: the distance between data points which is used for an anomaly score and the relative density of the data points to calculate the anomaly score. Performance is affected by the K variable and the computational complexity is high. K-NN is not robust against noise and it can misclassify instances when noise is present. Researchers use distance based K-NN in IDS to do the initial refinement of anomalies in the training dataset. Performance depends on the distance measurement that is defined between a pair of data instances. [12]

2.1.7 Fuzzy Logic

Fuzzy Logic or FL is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values true and false. [14] Intrusion detection involves the classification of a normal and an abnormal class, which makes fuzzy logic a tool for IDS since it also separates two classes. Fuzzy logic requires more fine-tuning and simulation before operational and it is hard to develop a model from a fuzzy system in comparison to other machine learning solutions due to the complexity involved in building the fuzzy model. [12]

2.1.8 Hidden Markov Model

The function of a Markov model is to produce a behavioral model. This behavioral model is composed of states, transitions and actions. An Hidden Markov Model or HMM can be defined as a tool for presenting the probability distribution of a sequence. An HMM is able to catch dependencies between consecutive sequences. Therefore, an HMM performs well when used for recognizing the structure of sequences. An disadvantage of HMM is that they are prone for overfitting when trained on datasets with a large parameter space. [2] Training of a HMM with the Viterbi algorithm [3] is computationally and memory intensive. [12]

2.1.9 Swarm Intelligence

Swarm intelligence or SI can be represented as a swarm of cooperating agents. The agents work together to achieve some purpose and task. Swarm operations are learning algorithms that are based on evolutionary computations. The agents are used to provide a set of classification rules for intrusion detection. SI based systems are adaptable so they can be adjusted to new stimuli. SI is scalable and flexible, agents can be removed or added without affecting the architecture but the complexity associated with swarms provides unpredictable results. There is no control over the swarms which makes the system uncontrollable. [12]

2.1.10 Ensemble Learning

Ensemble learning uses multiple classifiers to classify network data and then combines the prediction results of these classifiers to a final prediction measure. This technique helps in generating a set of hypotheses for a problem. It also provides a stronger generalization capability compared to individual classifiers. Each classifier is generated by using other machine learning algorithms such as a Decision Tree, Naive Bayes, Neural Network, Support Vector Machine etc. Some of the ensemble methods make use of the homogeneous base learners in which multiple instances of the same machine learning algorithm are used to generate a set of hypotheses over different sub-samples of the same training dataset. For example, a Random Forest is a ensemble classifier which combines the predictions made by the multiple decision trees. Other ensemble methods make use of the heterogeneous base learners in which different machine learning algorithms are used as base learners to generate a set of hypotheses. For example, a Neural Network can be trained over a training dataset and their predictions can be combined to generate common predictions. Also for combining these results, there are multiple options, such as majority voting for

[2] A parameter with a high parameter space are parameters that have a large space of possible parameter values.
[3] The Viterbi algorithm computes the most probabilistic sequence of states.

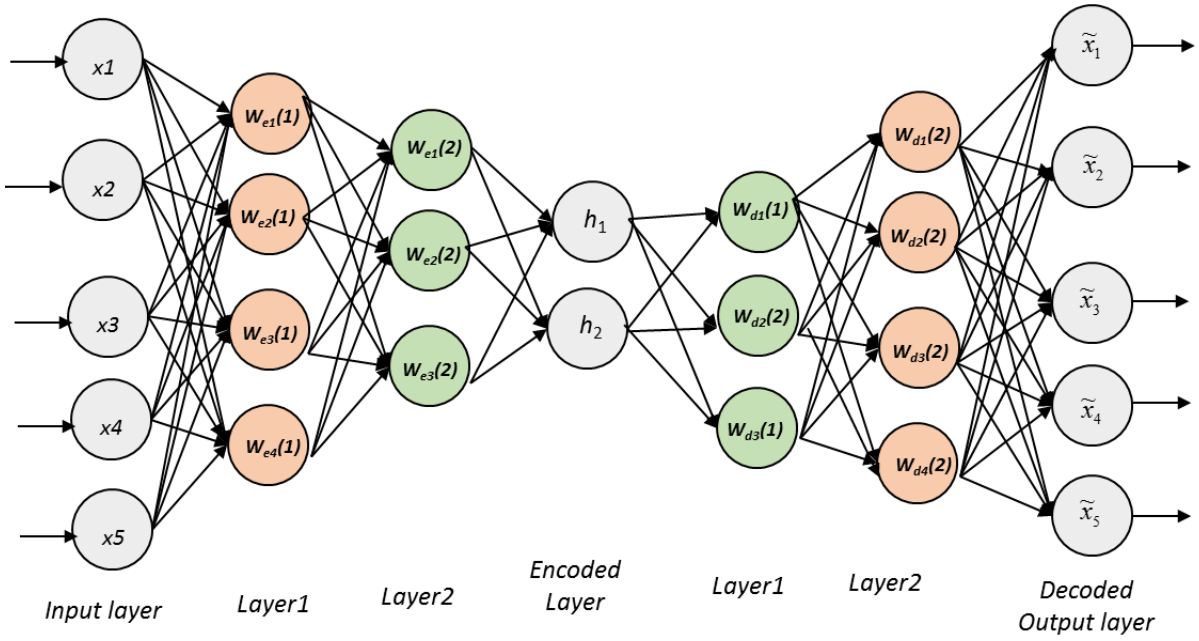

Figure 2.2: Stacked Auto-encoder

example. [4] Using ensemble methods makes the model more generalizable than a single classifier. Also, ensemble methods reduce overfitting and the prediction error. This comes at the cost of added complexity since there are multiple classifiers. As a result, this approach requires more training time and is more memory intensive. [12]

2.1.11 Stacked Auto-encoder

An auto-encoder or AE is used as a feature compression algorithm, this is achieved by a neural network. The compression and decompression are lossy. Further, an AE is a unsupervised learning algorithm which uses back propagation to generate output value that is very similar to the input value. An AE takes an input with a very high dimensionality, runs it through the neural network and compresses the input data into a small representation with two principal components. The first one is the encoder which has full connected layers with the purpose to compress input data into a smaller representation with less dimensionality than the input data, this is known as a bottleneck structure. From this bottleneck structure, it tries to reconstruct the input using full connected layers. [15]

[4] The class with the highest accuracy is the majority class. In majority voting, this class gets 100%.

The encoding process

A concise representation of the encoding process is as follow:

$$H = f_\theta(X) = \sigma(W_{i_j} X + b_{i_j}) \tag{2.1}$$

In the formula is X the input data and σ is the activation function. W_{ij} is the weight while b is the bias of the neuron unit. As for the activation function, the hyperbolic tangens is used:

$$Tanh(t) = \frac{1 - e^{-2t}}{1 + e^{-2t}} \tag{2.2}$$

The decoding process

The decoding process is to reconstruct the data so that it can be fed back to the neural network with the loss function for network training. In general, decoding can be regarded as the inverse of the encoding process. A simplified expression of decoding is presented as following:

$$Y = g_\theta(X) = \sigma(W_{i_k} X + b_{j_k}) \tag{2.3}$$

The loss function

The loss function is an essential element of the gradient descent (minimizing the function to its local minimum). Due to the reconstruction operation in the decoding phase, the Mean Squared Error or MSE is used in the AE model. The loss will be propagated back to the hidden layer to update the weights and bias of the neuron units:

$$J(W, b) = \frac{1}{2N} \sum_{n=1}^{N} ||Y_n - X_n||^2 \tag{2.4}$$

With N indicates the total number of samples.

Enhancing auto-encoding

To further enhance the capabilities of the auto-encoder, many auto-encoders are stacked together to form a Stacked Auto-encoder or SAE. The benefit from using the stacked self-encoder is that the feature extraction process gradually deepens as the number of layers increases. To train such a model, pre-training and fine-tuning are required.

2.1.12 Recurrent Neural Network

The RNN is created based on the idea that human cognition is based on past memory. The unique feature of an RNN are the cyclic structure of neurons, which is useful for extracting time-related characteristics. Assuming a sequence $X = x_0, X_2, .., X_{T-1}$, the hidden state h_t at time t can be represented as:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.5}$$

In the formula is σ a non-linear function, W_{xh} and W_{hh} are weight matrices, and b is a bias vector.

A stacked auto-encoder is an auto-encoder that consists of several layers of sparse auto-encoders, where the output of each hidden layer is connected to the input of the successive hidden layer. [15] Stacked auto-encoders are a popular technique for anomaly-based IDS to identify intrusions using deep learning. [7] They have the ability to reduce the dimensionality of the inputs and it addresses the imbalance issue between the minority and majority classes. [5] [7]

2.1.13 Convolutional Neural Network

Convolutional Neural Networks or CNNs are variants of the Artificial Neural Networks or ANNs.

Artificial Neural Networks

An ANN is a group of multiple neurons at each layer. A ANN is also known as a Feed-Forward Neural network because inputs are processed only in the forward direction. This type of neural networks are one of the simplest variants of neural networks. They pass information in one direction, through various input nodes, until it makes it to the output node. The network may or may not have hidden node layers, making their functioning more interpretable. [17]

Convolutional Neural Networks

A CNN consists of several hidden layers, called convolutional layers. The operation, when feeding the output forward is different, it is a convolutional operation. A more interpretable

[5] Imbalanced classifications are a challenge as most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class. This results in models that have poor predictive performance, specifically for the minority class. This becomes a problem because typically, the minority class is more important and therefore the problem is more sensitive to classification errors for the minority class than for the majority class. [16]

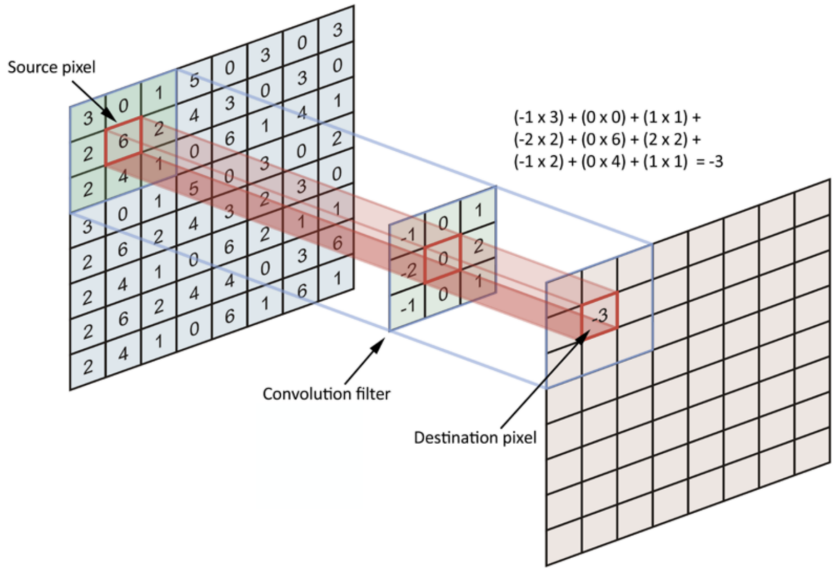

Figure 2.3: Global average pooling

way to explain the convolutional layer or pooling layer, the layer down-samples the data to a more compact representation using a filter, which is the extraction of spatial features shown in Figure 2.3. The last layer of a CNN-layer is fully connected. By connecting all the neurons of one layer together, a CNN is able to capture spatial features. This approach also enables the capability to automatically detect the important features without human supervision. These enhancements come at the cost of increased training time. [18]

2.1.14 Long Short Term Memory Recurrent Neural Network

This type of a ANN has neurons with memory, the content of this memory is held for an arbitrary amount of time. With this memory, a Recurrent Neural Network or RNN is able to take entire sequences of data into account such as network traffic data.

The continuous multiplication of the weight matrices W will cause the gradient descent to disappear or explode. A Long Short Term Memory layer or LSTM layer is an enhanced variant of RNN and addresses the issue of the vanishing gradient. The gate mechanism enables LSTM to perform better in long sequence problems, contains an input gate i_t, a forget gate f_t, and an output gate o_t. The neural unit updates as follows: the input gate combines the history information it has and the current input of the neuron:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_h) \tag{2.6}$$

After that, the forget gate must determine whether or not to pass the previous memory:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.7}$$

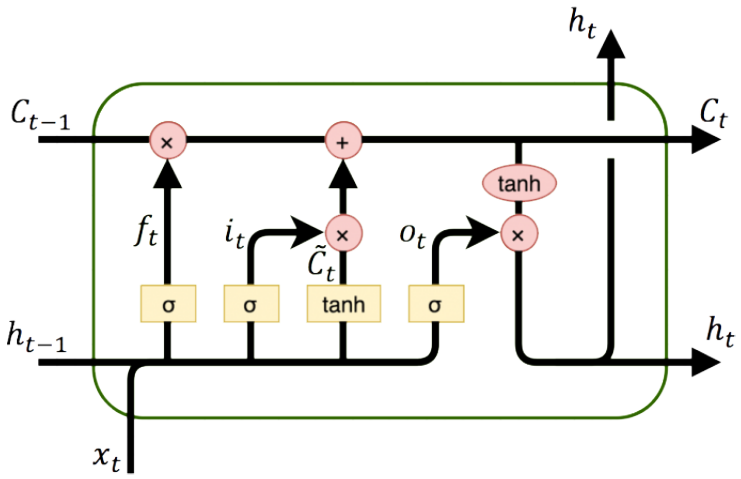

Figure 2.4: LSTM cell

c_t is an internal memory cell that combines the forget gate and the input gate. The output gate o_t shows the process to obtain the output of a neuron of a LSTM layer from c_t.

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.8}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \tag{2.9}$$

The activation of the hidden unit is updated as following:

$$h_t = o_t \circ \tanh(c_t) \tag{2.10}$$

Where $\circ$ denotes the Hadamard product. [10]

2.1.15 Deep Belief Networks with Restricted Boltzmann Machine

A Restricted Boltzmann Machine or RBM is another popular technique in IDS. It is a shallow, two layer neural network that consists of a visible and a hidden layer, where the nodes in the visible layer are connected to each node in the hidden layer. The reason that it is called a Restricted Boltzmann Machine is because there is no node in one layer that shares a connection. The RBM is trained to reconstruct the input data. It uses a similar approach as an auto-encoder but with the main difference that an RBM uses a stochastic approach with a defined distribution [6], while an auto-encoder is deterministic. A Deep Belief Network or DBN is a sequence of RBMs with at the end a Back Propagation Neural Network or BPNN. When data is

[6]For example, a Gaussian distribution.

not labeled, it is used to extract features for dimensionality reduction. When the data is labeled, a DBN is used for classification. [11]

2.1.16 Generative Adversarial Networks

Generative Adversarial Networks or GANs are neural networks that are able to model complex, high dimensional distributions of raw, real-world data, which indicates that it is an effective method for anomaly detection. Since network traffic data is raw and difficult to normalize, there are little to no well-defined structural rules for normalizing network traffic data.

Only a few researchers have explored the capabilities of GAN-Based anomaly detection which makes this a less popular intrusion detection method. [19] This paper states that it is the first paper in their knowledge that acknowledges GAN-Based methods for IDS, the paper dates from 2018 which indicates that it is one of the most State-of-The-Art models. The expected performance results should align with other SoTa methods. Training and comparison with other SoTa models is done with the KDD CUP 1999 dataset.

GANs make use of a discriminator neural netwerk and a generative neural network. The purpose of a GAN is to make an IDS more robust against intrusions that a black-box IDS is not be able to detect. A GAN generates intrusion instances that the black-box won't be able to classify correctly. These generated instances can be used to re-train the black-box IDS with the generated data. [1]

2.2 Regularization for Deep Learning

Many strategies for generalization have been developed in the field of deep learning. When a model is regularized, the model is modified in a way that the generalization error is reduced, and not the training error. Subjectively, regularization comes in two forms: explicit and implicit. Both control the effective capacity of the network with the purpose of reducing the overfitting. [20]

- Explicit regularization: Regularizing methods which are not structural parts of the network architecture, the algorithms or the data can be added or removed easily. These algorithms are: Weight decay, Dropout, Data augmentation and Stochastic depth. [20].
- Implicit regularization: These regularization methods use characteristics of the network architecture, the learning algorithm or the data in order to control the effective capacity of a neural network. These algorithms are: Stochastic Gradient Descent Algorithm, Convolutional layers and Batch normalization [20].

Sparse regularization or L1 regularization

Classic multi-layer perceptron are composed by fully connected layers, often referred as dense layers, given that every unit has independent weighted connection with all the units in the next layer. In contrast to dense layers, sparsity refers to representations with most coefficients being zero. Deep learning often is designed with the objective of obtaining such sparse representation in its hidden layers. The idea of sparsity regularization is based on the assumption that the output of a model can be learned by reduced number of variables. Sparsity can be enforced using implicit regularization (for example using convolution layers) and explicit regularization (e.g. including a loss function that penalizes non-zero weights like in sparse auto-encoders). Dense connections often waste valuable resources, often adding capacity that is used inefficiently. Enforcing sparsity is attractive given that it reduces computational and memory requirements. [20]

Weight decay or L2 regularization

Weight decay has the ability to constrain a network, thus capable of decreasing the complexity of a neural network. It limits the growth of the weight, preventing the weights growing into too large values, unless it is necessary. [20]

Data Augmentation

One of the best ways to generalize a model is by training it on more data. When data is limited, new data can be generated and can be added to the training dataset. This technique is called Data Augmentation. It is a regularization method that has shown the capability of achieving higher or same performance without any other explicit regularization techniques. [20]

Stochastic Gradient Descent

Stochastic Gradient Descent or SGD is the most common optimization algorithm used to minimize the objective function of a neural network. SGD can be considered a implicit regularization method for Deep Learning. SGD methods garantees of convergence to minimizers of strong-convex functions, saddle point avoidance and robustness to input data. However, SGD has some limitations. They require small batch sizes, limited parallelization capabilities since an SGD is a sequential model. [20]

Adversarial Training

Machine learning models are often prone to adversarial [7]. In adversarial training, a neural network is trained on adversarial examples in order to make the model more robust against attacks or to reduce the test error on clean input examples. The idea has become more popular in recent years since it reduces the error rate in the original dataset via adversarial training. [20]

Pooling

Pooling is a operation that is almost always used in all the convolution neural networks. Pooling operations make the output representations approximately invariant to small translations of the input image, i.e. translate the input by a small amount, the values of most of the pooled outputs do not change. It is performed by modifying the output of a neural network layer by replacing the output of a particular location in with a summary of the nearby outputs. [20]

Dropout

Dropout addresses two major issues of neural networks: it prevents overfitting and it enables the combining of multiple different neural networks. Dropout randomly drops units, i.e. temporarily removing them from the network with their incoming and outgoing connections during the training process. This method has proven to reduce overfitting and it results in major improvements over other methods against overfitting such as weight decay and data augmentation. [20]

DropConnect

DropConnect is a newely proposed regularization method for large fully-connected layers. It regularizes the dropout to prevent co-adaptation of feature detectors. DropConnect randomly selects subset of weights within the network and sets them to zero, so it introduces dynamic sparsity which helps with the generalization of the model on small datasets. It has been found that the performance can sometimes be better compared to the dropout layer. [20]

2.3 Overview of available datasets

Datasets are necessary for modeling a network intrusion detection model. The data is needed to train a model (training data) and evaluate the prediction performance of the model (test

[7] Malicious input that is designed to thwart the model.

data) with a smaller fraction of the dataset. Unlike many other research fields, such as image recognition, the available datasets for network intrusion detection are limited. Also, some of the datasets have specific properties that makes the dataset not useful for training models. For example, a dataset that contains few normal network traffic data is not realistic. Most of the traffic should be normal behavior. The most common datasets for intrusion detection are discussed and selected that are further used to train and evaluate models.

DARPA98

DARPA98 was one of the first datasets developed for network security analysis, developed by Lincoln Laboratory in 1998-1999. It exposes the issues with injection of artificial attacks and benign [8] traffic. The dataset contains e-mail, browsing, FTP, Telnet, IRC and SNMP network data. It contains attacks such as DoS, Guess password, Buffer overflow, remote FTP, Syn flood, Nmap and Rootkit. This dataset does not represent real-world network traffic data: the dataset does not contain false positives. [2] Since the dataset is more than 20 years old, it is considered outdated. When using it to built a Intrusion Detection System, it would have less effective results in modern situations.

KDD'99

KDD'99 is a updated version of DARPA98. It contains different attacks such as Neptune-DoS, pod-DoS, Smurf-DoS, and buffer overflows. Bening and attack traffic are merged together in a simulated environment. The dataset contains errors such as duplicate records, data corruptions that led to skewed testing results. Later on, the NSL-KDD dataset was created to address some of the issues with KDD'99. [2]

DEFCON

DEFCON-8 was created in 2000. The dataset contains port scanning and buffer overflow attacks. Later, DEFCON-10 was created in 2002 which contains port scan and sweeps, bad packets, administrative privilige and FTP by Telnet protocol attacks. This dataset was created during a "Capture The Flag (CTF)" contest, so it is not real world network traffic data since it mainly consist of intrusive network traffic. The dataset is mainly used to evaluate alert correlation techniques. [2]

[8] Benign or normal user network behavior.

CAIDA

Developed by Center of Applied Internet Data Analysis in 2002-2016, CAIDA is an organization that released three different datasets over a period of time. Firstly, the CAIDA OC48 was the first dataset to be released. This dataset contains different types of data, observed on a OC48 link in San Jose. Secondly, the CAIDA DDoS dataset, contains one-hour of DDoS attack traffic which split of 5-minutes pcap files. And lastly, the CAIDA Interet traces 2016, which is passive traffic traces from CAIDA's Equinix-Chicago monitor on the Internet backbone. These CAIDA datasets contain very specific events and network traffic data and also, the network traffic data is anonymized together with their payload, protocol information and destination. [2]

LBNL

Lawrence Berkeley National Laboratory and ICSI 2004-2005 (LBNL) contains network traffic data that was collected in a medium-sized site. It does not have a payload and is completely anonymized, which means that all the information that could identify a individual IP is removed. [2]

CDX

Created by the United States Military Academy in 2009, this dataset represents the network warfare competitions, that could be utilized to generate modern day labelled dataset. It includes network traffic such as Web, e-mail, DNS lookups and other required services. The attackers used tools such as Nikto, Nessus and WebScarab to carry out reconnaissance [9] and attack automatically. [2]

Kyoto

Developed by Kyoto University, 2009. This dataset has been created through honeypots, so there is no process for manual labelling and anonymization. However, the dataset has a limited view of the network traffic because only attacks directed at the honeypots can be observed. It has ten extra features such as 'IDS_Detection', 'Malware_Detection' and 'Ashula_Detection'. These features are useful in NIDS analysis an evaluation. The normal traffic data was repeatedly simulated during the attacks and produced only DNS and mail traffic data. This dataset does not contain false positives, despite that false positives are important for limiting the amount of false alerts. [2]

[9] Gathering information about the target

Twente

This dataset was created by the University of Twente in 2009. The dataset includes three services such as OpenSSH, Apache web server and ProFTP using auth or ident on port 113 and captured data from a honeypot network by Netflow. This dataset contains some unknown and uncorrelated alerts traffic. The network traffic data is labelled and is more realistic compared to previous discussed datasets, but the lack of volume and diversity of attacks is obvious. [2]

UMASS

University of Massachusetts developed this dataset in 2011. The dataset includes tracefiles. [10] It is generated using a single TCP-based download scenario. Since it lacks variety in network data, it would not be usable for testing IDS. [2]

ISCX2012

Created by the University of New Brunswick in 2012. The dataset contains various multi-stage attack scenarios and realistic benign traffic data with background noise. It includes network traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols with the full packet payload. However, it is no realistic traffic data since 70% of todays network traffic data are HTTPS while this dataset does not contain any HTTPS traffic data. Also, the simulated attacks are not based on real world statistics. [2]

ADFA13

This dataset is generated by the University if New South Wales in 2013. The dataset includes normal training and validating data. The attack scenarios include FTP and SSH password brute force, Java based meterpreter [11], Add new Superuser, Linux Meterpreter payload and C100 Webshell attacks. However, these attacks lack variety and some attacks are not well seperated from normal network traffic behavior. [2]

[10] A trace file is a file containing a trace of certain events that happen or will happen.

[11] Meterpreter is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code. [21]

UNSW-NB15

"The raw network packets of the UNSW-NB 15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours.

Tcpdump tool is utilised to capture 100 GB of the raw traffic (e.g., Pcap files). This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The Argus, Bro-IDS tools are used and twelve algorithms are developed to generate totally 49 features with the class label." [22]

CIDDS-001

The CIDDS-001, Coburg Network Intrusion Detection Dataset, is a labelled flow based dataset, developed for the evaluation of Anomaly based NIDS. It contains unidirectional NetFlow data. It consists of traffic data from two servers: OpenStack and External server. The dataset is generated by emulating a small business environment that consists of OpenStack environment having internal servers (web, file, backup and mail) and a External server for file replication and to host a web server for a total of 172,839 instances. The dataset consists of three log files: attack logs, client configuration and client logs; the dataset has 14 attributes. Appendix B contains a table to describe the attributes of the dataset. [23]

CICIDS2017

In 2017 the Canadian Institute for Cybersecurity developed the CICIDS2017 dataset. First, to evaluate the effectiveness of the IDS, the Canadian Institute of Cybersecurity presented a state of art dataset named CICIDS2017, consisting of several files. The dataset draws attention of many researchers as it represents threats which were not addressed by older datasets. While undertaking an experimental research on CICIDS2017, the dataset has few major shortcomings. These issues are sufficient enough to bias the detection engine of any typical IDS. The dataset contains 3,119,345 instances and 83 features, containing 15 class labels (1 normal + 14 attack labels). Further, examining the instances of the combined files, the dataset contains 288,602 instances having missing class label and 203 instances having missing information. By removing such missing instances, the combined dataset of CICIDS2017 has 2,830,540 instances and no duplicate instances can be found. [24] More details of CICIDS2017 can be found in Appendix A.

2.4 Critisism on Machine Learning in IDS

Machine Learning is a new, powerful and successful technique for detecting patterns in data, but to create a good model you need lots of data. Within image recognition, you can easily validate good or bad data. Within network traffic data, it is much harder to evaluate the quality of the dataset, this leads to many uncertainties. For example, what is the difference between a good network traffic dataset and a biased network traffic dataset? Another critical point in network traffic data is the limited amount of labeled network traffic data that is available. It is not difficult to log network traffic data and generate a dataset, but it is much harder to label every instance correctly. Since there is no way to know beforehand if network traffic data is benign or not, it is necessary to simulate and label these attacks one by one. A third critique is the change in user behavior or evolutions within intrusion techniques. The internet is growing fast and constantly changing. Millions of users with different behavior that keeps on changing, different types of attack being designed or new exploits being found, with of course different behavior. It is hard to keep up with all the different kinds of behavior while being able to separate the benign network traffic from malicious network traffic. But even knowing this, it is important to keep on exploring better techniques for intrusion detection systems since malicious users are determined to bypass the implemented security techniques.

3

Preprocessing

A dataset may contain outliers, missing values, duplicate instances or discrepancies. In data cleaning, datasets are corrected by removing corrupt values, renaming values or correcting inaccurate values. Data cleaning can also be used to solve class imbalance, this is caused by one class that is represented by an overly big fraction of the dataset or a class that is not represented by much data. In some cases, it is preferred to remove a fraction of the instances of the majority class or, in other cases where there is not enough data available, additional data is added of the minority class to the dataset. There are several ways to do this, for example, adding missing instances that completes a Gaussian distribution.

Preprocessing can also involve transformations within the dataset. For example, data that is generated by a large datacenter will contain a much bigger amount of data packets, than data that is generated by a school environment. In this situation, a possible solution could be to scale the data. This could put the data into perspective, but it is important to note that this is trial-and-error.

The last part that can be considered preprocessing is the selection of the important features that have a high impact on the training of the model, in classic machine learning, this is done in a more manual way, while deep learning does this automatically.

3.1 Selected datasets

The previous Chapter 2: Literature review discussed several datasets with their pros and cons. One point that was made in 2.3 "Critisism on Machine Learning in IDS", was the lack of modern datasets. It was clear from each dataset that each and every one of the datasets have their own shortcomings, however some shortcoming can be overcome through preprocessing.

CICIDS2017

CICIDS2017 is one of the most relevant of the discussed datasets. It contains several types of attacks and is also one of the most current datasets available. However, the problems of the CICIDS2017 dataset must be addressed to form a consistent dataset. The dataset is scattered among 8 different files. All the files combined, results in a volume of 2,830,540 instances in total and although the dataset is imbalanced, the dataset can become balanced by removing instances from the majority classes in each file. Lastly, the scattering of the dataset can be used in our advantage by training a model for every attack class, which is desirable since each type of attack is different.

3.2 Pipeline

A pipeline is a process that involves several stages that are used to clean and process a dataset. Each dataset that is used for training must run through this pipeline to be able to evaluate the models, generated by these datasets in the same way.

The first step removes attributes: destination port, timestamp and protocol. The reason is that it could bias the classifier because of the lack of variation between the values it contains. In the second step, since every file contains their own attack class, there is no need to use strings in the label attribute, so the attack label is replaced by a "1" representing "True" and benign traffic data is replaced by a "0" which represents "False". Then all string values that contain "Infinity" or a deviation thereof are replaced with NaN, representing a empty value.

Most training of deep learning techniques is done through GPUs since they are optimal for matrix operations, which can be fully processed in parallel. But GPUs are mostly designed for 32-bit operations, unlike the majority of CPUs that mostly support 64-bit instructions. To optimize the datatype for GPUs, numeric data is casted to a float32 value, the optimal dataformat for GPUs. Most of the data in the dataset is not in the most optimal format, therefore every numeric datatype is tried to be replaced with its smallest possible data type that fits the data.

3.3 Feature selection

As mentioned before in the introduction of Chapter 3, feature selection is mostly required for the classic machine learning techniques that do not involve deep learning. In feature selection, a subset of the attributes of the datasets are selected that have the highest contribution on the variance of the label classification. Several approaches have been studied, the best ones for this case:

- Correlation of the attributes with the attack label, done through a correlation function of the pandas library. [25]
- Univariate feature selection with F-test for feature scoring, this is the analysis of variance or ANOVA, which measures the variation between a sample its means or variations within a sample.
- Backwards elimination, removing attributes and testing if the results are still the same.
- Forwards elimination, adding attributes and testing if the results are still the same.

3.4 Dimensionality Reduction

The goal of dimensionality reduction is to transform a high dimensional attribute space to a lower dimensional attribute space so that the lower dimensionality is able to represent the higher dimensional attribute space more compact, which is beneficial for machine learning methods which can require high computational or memory requirements. Feature selection is different from dimensionality reduction. Unlike dimensionality reduction, feature selection does not change the data, while dimensionality reduction can transform the data, but it is not a neccesity.

3.5 Normalization

Different datasets are generated in different setting with their own environmental properties. To overcome this difference in scale, it is possible to normalize the data. This will make sure that attributes fall within the same range when their attributes are used for training a model.

- Min-Max normalization, this scales all the numeric values in the range [0,1], the minimum value gets transformed into 0 and the maximum value into 1.
- Z-Score normalization (zero-mean normalization), values are normalized based on mean and standard deviation of the data A.

4

Reproduction of SoTa-Models

In this section several proposed State-Of-The-Art models from frequently cited papers are analysed. The initial paper is chosen for its ease to setup the environment fast. Each of the other papers discusses innovative methods to detect intrusions. The described model design is always adopted as good as it is described. If the model experimentally shows that the detection rate is high, further research is conducted into the generalizability capabilities of the model. First, the proposed ideas are discussed for each paper with the reasoning of the choice that explains why the paper is interesting for this research. In addition, each paper has at least one model that can be reproduced, but it will often be necessary to make structural adjustments to be able to reproduce the model. The reasons for this lie mostly in the fact that the structure of the dataset used in the paper has a different structure than the datasets that was chosen in the thesis. Another reason would be the vaguely described model. In this case, assumptions must be made to reproduce the model. Those assumptions are defended with the general methods that are used in other papers or standard values, proposed by the TensorFlow [26] or Sklearn [27] library.

4.1 Reproduction model 1: Support Vector Machine

The paper "Shallow and Deep Network Intrusion Detection System: A Taxonomy and Survey" analyses multiple methods for network intrusion detection, one of them is an SVM.

4.1.1 Discussion of the paper

This paper discusses multiple Machine Learning models that detect intrusions. The Support Vector Machine that was discussed in section 2.1.3 "Support Vector Machine" is one of the more popular techniques within classical machine learning. [6] The paper gives a reference to another paper that further explains the reason [28] [29].

The referenced paper "Study on Implementation of Machine Learning Methods Combination for Improving Attacks Detection Accuracy on Intrusion Detection System (IDS)" states the following: *"This IDS using SVM methods because related to Mukkamala research stated that SVM is better than Artificial Neural Network." "In 1999, Mukkamala research about Intrusion Detection System using SVM and Neural Network methods with KDDCUP 1999 data as the training data. The result of this research shows that the SVM method is more accurate than Neural Network method."* [6] [28] [29] This statement should be taken with a grain of salt, the complexity of neural networks was limited in 2000 since computing capabilities have evolved significantly in the last 20 years (>1,000 times increase in computing power [30]) to be adequate for neural network training. It is also only in the recent years that the massive parallelization of a GPU has been taken advantage of.

The paper gives the intention that a dataset can be classified very well with many different machine learning methods. The anomaly detection accuracy is higher than 99% in both cases, this classification was achieved with 14,000 random chosen instances. 7,000 instances are used for training. The discussed SVM was rated as a better classifier in the paper, mainly due to the fact that it had a shorter training time, this in combination with a slightly higher anomaly detection accuracy. Also, with the Artificial Neural Network or ANN, no statements were made of the amount of false positives or false negatives, as was the case with the SVM classifier.

4.1.2 Reproduction

The main purpose of the model is to create a starting point to test the preprocessing and test the evaluation techniques. The sklearn library offers a excellent range of different SVM implementations, for this model a LinearSVC was chosen, this SVM must be able to distinguish attacks from non-attacks. "This class supports both dense and sparse input and the multiclass

Training	ANN	SVM
# features	13	13
# of data points	7,000	7,000
Architecture	[13,40,40,1]	-
Balanced accuracy	99,41%	99,52%
# of false positives	-	19
# of false negatives	-	15

Table 4.1: Comparison of ANN and SVM

support is handled according to a one-vs-the-rest scheme."[31] The paper describes the model in detail, exact specifications are listed below.

- The 10 best features with the highest impact on the results from the original 41 features are used.
- The dataset contains 100,000 instances.
- From the 100,000 instances, 95% are used to train the model and 5% of the dataset is used to test the model.

Remarks

The first specification states that the 10 best features must be chosen to achieve the best results. To select the top 10 features that came with the dataset, the correlation with the attack class is calculated with a built in pandas library function. From all these features, a score is calculated that measures the correlation with the attack class and from the resulting list, the 10 best features are selected.

Then, 100,000 records are chosen randomly. A better idea would be to select 50% of the data as anomaly network traffic and the other 50% to be normal user behavior. But since this is reproduction, it is important to follow the guidelines of the paper.

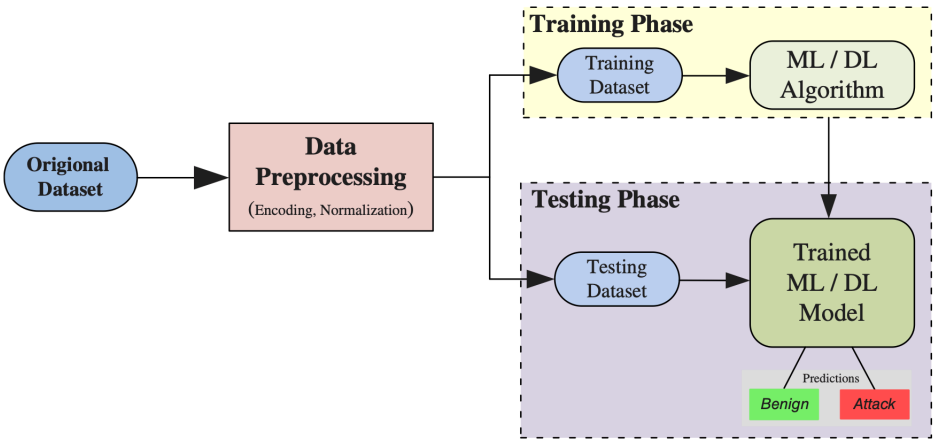

Figure 4.1: Flow for building a NIDS; DL may split the train dataset into a train and a evaluation dataset

4.2 Reproduction model 2: A Limited RNN

4.2.1 Discussion of the paper

This paper investigates whether connections can be found in sequences of data. This requires a modified version of an artificial neural network or ANN, namely a RNN. A summary of how an RNN works is explained in 2.1.12 "Recurrent Neural Network". In this version of an ANN, there will be cells that keep values for an arbitrary time. It is interesting to know whether, in addition to pattern recognition, it is also able to take entire sequences of data into account, such as network traffic data. An ANN gets the properties of an RNN by adding special cells to it, cells always come in layers. This described layer is called a Long-Short-Term-Memory or LSTM-layer. The paper is describing each setting and parameter of the model in detail and also makes use of a dataset that, relatively speaking, is very recent (2017). Further, the paper also states that limited processor time is required for the model, so training and thus research could go faster. [8]

There are other benefits of using a model that require limited processing time.

- On-device training becomes feasible.
- Headroom for generalization.
- Insight into the model its behavior.

4.2.2 Reproduction

Specifications:

- For training, the CIDDS-001 dataset is used. Containing 671,241 records, 67% or 449,731 instances are used for training and the 221,510 instances are used for testing.
- One hidden LSTM layer with six neurons with a output layer that has 5 neurons.
- For the logarithmic loss function, ”categorical_crossentropy” is used.
- From the 100,000 instances, 95% are used for training and 5% is used for testing the model.
- Features: Source IP-address, Source Port, Destination IP-address, Destination Port, Protocol, Date first seen, Duration, Bytes, Packets, Flags, Class, AttackType and Attack ID
- After hyperparameter tuning, learning rate = 0.01, number of epochs = 200, batch size = 500, rho = 0.05, epsilon = 1e-8, decay = 0.0.
- Class balanced performance is 84.83%.

The model showed a relative low accuracy when built with all the named features except for the source and destination IP-addresses, namely 60%.

However, when the IP-addresses were included, the accuracy rose to almost 97%. This could mean that the model classifies network traffic data based on the identification of the sender and receiver. This is highly unwanted behavior, the model is expected to classify network traffic by its technical properties, not by the IP-addresses.

This raises the following question: Why did the researchers of the paper not recognize this problem?

Whatever the answer might be, the model is not useful for intrusion detection. This SoTa model is not further evaluated since its training is biased by strongly correlation between the IP-addresses and the attack class.

4.2.3 Remarks

The results are not fully unexpected, when performing a correlation operation on the dataset, the features (without the source and destination IP-addresses) that this model used were not correlated to a attack class.

Differences with reproduction model: The authors of the paper give a detailed description of the architecture of the model but there are a few things that were not clear.

"TCP flags are concatenated in one single column": How should the column be encoded? Solution: It is not clear how they achieved this, the general solution for this problem is to create a column for every feature. The performance impact in the reproduced model is expected to be low to non-existent. First, the flags did not show any correlation with the attack class and second, every neuron is connected with every input. This should make the weights of the input connections more precise than the original model.

4.3 Reproduction model 3: Stacked Auto-encoder with a Recurrent Neural Network

4.3.1 Discussion of the paper

A Network Intrusion Detection Method Based on Stacked Auto-Encoder or SAE and LSTM. The following reproduction model addresses the challenges that come with network traffic data that prevents it from creating a well-functioning model. One of those problems is the amount of features. The datasets come with 76 features, it is interesting to filter out the least interesting features and let an ANN only take into account the most important features. They cite a few methods for this and decide that an SAE is the pioneer in terms of feature reduction. Each choice is carefully considered and they also defended their decisions with strong arguments. It is therefore interesting to write these out as well. [10]

Why not train a model to only recognize normal traffic behavior?

By discriminating the difference between the input data and normal data, the anomaly-based network intrusion detection methods are able to detect the unknown attacks. However, this method can easily make the intrusion detection system face a high false alarm rate. This is because some intrusions act similarly to the normal activities. This generally results in a high false positive alarm rate and thus not recommended.

Why build a model with deep learning?

Deep learning is a star member of the machine learning community and also has been applied in many fields, such as medical diagnosis, automatic driving, etc. The SAE serves as a feature

extractor tool. The output of each layer becomes the input of the next layer. The LSTM-layer aims to find the time relationship, and is the second important component of the model. Finally, several fully connected layers are joined after the LSTM-layer.

Which feature reduction method to use?

Network traffic has high dimensional data, which makes machine learning methods easy to suffer from too much dimensions. Many studies have already focused on this problem of feature reduction, aiming to provide a simple and effective set of features. [12] Feature selection and extraction achieve the purpose of feature reduction. In the feature selection process, certain original features will be selected according to rules. In contrast, the results of feature reduction is low-dimension data through feature mapping or other transformations.

Auto-encoder

An auto-encoder is used as a feature compression algorithm, this is achieved by a neural network. The compression and decompression are lossy. A detailed description can be found in section 2.1.11 ”Stacked Auto-encoder”.

Architecture of the model

The model consists of 3 main components. The first component is the feature extractor or, in this case, a stacked auto-encoder or SAE. The second component is the classifier or the RNN. The last component is the evaluation block.

4.3.2 Reproduction and Remarks

The model is described with enough detail to make reproduction feasible. Most of the parameters are mentioned in detail. However, there is a difference between the described model and the reproduced mode. The paper has 77 initial parameters but with one-hot-encoding of the categorical features, it was expanded to 196 features, while the datasets that are used for reproduction only contain 42 features. This problem is overcome by keeping the ratio of the number of neurons to the number of features equal. This results in a smaller neural network that, in theory, should be even more resistant for overfitting.

In the next step, the preprocessed dataset is split in a training dataset (70% of the dataset) and a testing dataset (30% of the dataset). The model is trained with the CICIDOS2017 dataset

and achieved a balanced accuracy of 97% or higher.

This behavior is validated by the other datasets. These were also able to achieve good results. This makes the model suitable for IDS. Also a relatively small neural network creates headroom to make adjustments in the next phase and to test the neural network more in-depth.

4.4 Reproduction model 4: Combined CNN and RNN

4.4.1 Discussion of the paper

Using ML based methods for intrusion detection has greatly improved detection accuracy of anomalies but it comes with the problem of high rate of false alarms or false positives. This significantly discounts the effectiveness of ML based IDS. This model considers both spatial and temporary features in network traffic datasets. The proposed idea is a hierarchical CNN+RNN neural network, LuNet. In LuNet, the CNN and the RNN learn input traffic data in sync with a gradually increasing granularity such that both spatial and temporary features of the network traffic data can be effectively extracted. LuNet offers a high level of detection capability and has a much lower false positive alarm rate compared to RNN-LSTM-only techniques. [9]

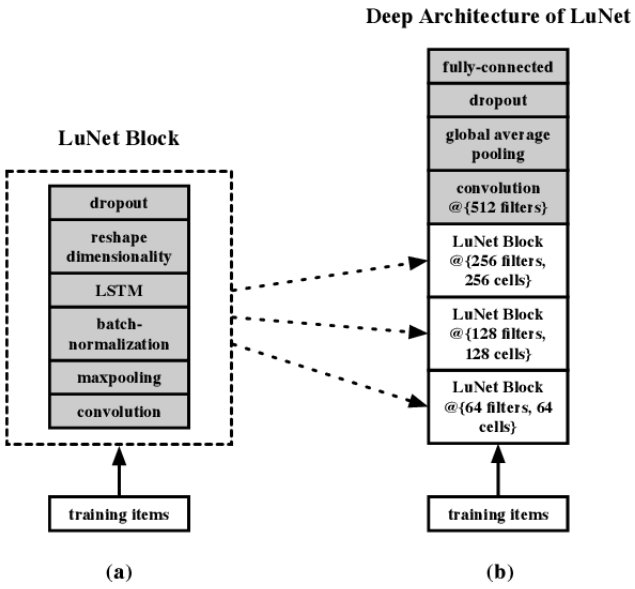

Figure 4.2: LuNet architecture

4.4.2 Reproduction and Remarks

Architecture of the LuNet block

LuNet is a hierarchical deep neural network, it is made of multiple levels of combined convolutional and recurrent neural networks. In each LuNet block, the input data is learned by both CNN and RNN. As the learning progresses from the first LuNet block to the last LuNet block, the learning granularity becomes increasingly detailed. This arrangement effectively creates a synergy between the CNN and RNN layers which causes temporary and spatial feature extraction. [9]

Temporary feature extraction

The temporary features are extracted by the RNN-layer. The RNN-layer is explained in more detail in section 2.1.14 "Long Short Term Memory Recurrent Neural Network". In short, an RNN-layer is able to learn on individual instances by recognizing the relationship between instances. This is achieved through feeding back to the current learning what has been learned from the previous learning. This is how the temporary features from the input data are extracted. [9]

The LSTM-layer further enhances these capabilities. It is an improved version of an RNN-layer, it keeps track of the learning error in the long dependency. If this error becomes large enough, the LSTM-layer will invalidate errors by dropping these features out so that only the persistent temporary features are retained. [9]

Remark: Using an RNN-layer requires input data of 3 dimensions, since the input data only has 2 dimensions, the input data needs to be reshaped to a higher, 3^{th} *dimension. As a solution, a zero is added to each instance to match the* 3^{th} *dimensional data format requirement.*

Spatial feature extraction

The next layers are used to extract spatial features, which is handled by the CNN-layers. The specific details of this CNN implementation are explained further but a detailed description of the CNN-layers can be found in section 2.1.13 "Convolutional Neural Network". The hidden layers in a CNN are called convolutional layers, these layers are used to extract spatial features. At the end of a CNN, a fully connected layer makes the network able to take the spatial features into account to make a better decision at the output.

Overfitting prevention

At the end of a LuNet block, overfitting is considered by adding a dropout layer at the end of the model. A dropout layer randomly removes connections from a deep neural network which helps to prevent overfitting. Dropout is explained in more detail in section ”2.2 Regularization for Deep Learning”.

The datasets that the paper uses are the NSL-KDD and the UNSW-NB15 datasets, both described in subsections of 2.2 ”Overview of available datasets”. This architecture combined with these datasets resulted in a accuracy of 99.24% for NSL-KDD and 99.36% for UNSW-NB15. The false positive alarm rate for NSL-KDD is 0.53% and 3.96% for UNSW-NB15. [9] It is however important to note that these datasets were not balanced before training.

5

Analysis of the generalizability of selected SoTA models

The previous section discussed the reproduction process with the goal being reproducing the models of the papers as good as possible. In this section, the models that were successfully reproduced are further investigated by generalization. This generalization process involves several steps that are further explained, also there are many possible methods to do this. Generalization techniques are performed until the limits of the model are reached which means that, if, after many different attempts, the model its performance does not evolve, the generalization process is interrupted.

5.1 Structural Generalization

Generalization in this context refers to adding algorithms to a neural network that makes the model prone for overfitting; the paper [20] presents several techniques that can be applied in SoTa Deep Learning techniques, shown in Table 5.1. The details of these techniques are further discussed in section 2.2 "Regularization of Deep Learning".

Deep Learning technique	Regularization Technique	Implemented Technique
LSTM-RNN	Dropout, Mixed-Norm Regularization, Weight Sharing, DropConnect, L2-Regularization, Zoneout	Dropout
CNN	Drop Out, Weight Decay, Pooling, Drop Connect, Data Augmentation	Drop Out, Pooling, Weight Decay
RBM	L2-Regularization, Weight Decay, Sparsity Regularization	-
DBN	Dropout, L1-Regularization, Sparsity Regularization	-
AE	Weight Decay, Dropout, L1-Regularization, Max-Pooling, Data Augmentation, Gradient Decent	Dropout, L1-Regularization Gradient Decent

Table 5.1: Regularization for different Deep Learning Techniques

5.2 Conceptual Generalization

Conceptual generalization refers to generalization techniques that do not involve new algorithms to the neural network. Instead it alters the way the model will learn, so that the model its performance is maximized.

5.2.1 Limit the size of the neural network

By limiting the size of the neural network, the neural network is less likely to overfit since there are not as many neurons available to learn from the input data, so learning abilities are more scarce and thus the features must be selected more carefully so only the general important features can be selected.

5.2.2 Stop training early when the model is overfitting

When the neural network its training data is growing but the validation dataset its accuracy is shrinking more than the model is overfitting, when this happens the training procedure is

interrupted. This is set as default behavior in all the models, even for the reproduction models. There is no harm in doing that because training is stopped when the model has achieved the highest/best possible results.

Training can also be stopped early when performance has converged, when the validation loss [1] has not changed since the last 20 training steps, for less than 0.0001, the training stops.

5.2.3 Training with every the dataset

This approach focuses on the anomaly detection by differentiating normal user behavior from any malicious network behavior, so that the model to separates malicious network data from normal network data traffic.

Start training from any dataset

To reduce bias from a particular dataset, it is important to test the performance when training starts from any dataset. This process requires a lot of training time since 36 different datasets are available. Each model is re-trained 36 different times. Some default criteria are included, to exclude datasets that are not helpful and to reduce overfitting. The first criterium is to not include datasets to the resulting dataset that hurt the performance, which means that a baseline is set for training. This baseline is the balanced accuracy of the testing dataset. This testing dataset contains 30% of the data and is randomly chosen. When the newly appended dataset manages to keep the performance levels, this new dataset is appended to the resulting dataset. When the dataset did not manage to keep the performance metrics, the dataset is not included in the resulting dataset. Reducing overfitting is a major problem in IDS training because it was known beforehand that a model did not learn to detect anomalies and therefor the model seemingly became aware of the environment. Thus not getting an good estimation of what a anomaly is. To address this issue, training stops when the performance of the testing dataset starts to decline when the training data its accuracy starts to grow.

The final result of this approach should be a model that has the highest balanced accuracy, over as many data as possible. So any network traffic data from any dataset will be classified correctly, in theory.

[1] Validation loss is the error after running the validation set of data through the trained network. [32]

Start training from the most accurate dataset

This approach further builds upon the previous approach. In the previous approach, a frequency table was added. When a model was able to combine well with another dataset, its counter was increased by one. The final result is a tabel with the amount of times a dataset was able to combine itself well with other datasets. This dataset is then used to start training and a similar approach is used as in the pervious section, every dataset is tested to evaluate its performance when added with this dataset.

The resulting model should be the most flexible model.

Start training from the best dataset but exclude one dataset for evaluation

Previous models are evaluated in this generalization step. Both models are now re-trained with the datasets that were able to combine well, except for one dataset. The dataset that had the highest accuracy after the combining procedure, is kept apart.

5.2.4 Train a model to detect one attack class

Another approach to train a model was to only expect from the model to detect one attack class. For example, DoS attacks. There are 3 different DoS datasets available, a DoS dataset from 2017, 2018 and 2019. Since the available datasets are limited, every combination can be tested while one dataset is kept apart for evaluation purposes.

5.3 Reproduction model 1: Support Vector Machine

There is no need to look further into the generalizability of this model since it did not perform well in the first place and second because it was never expected from this model to set a benchmark for IDS. The main purpose for this model is to create a working environment to develop, to evaluate and to test the SoTa models, in which this model has fulfilled its goal.

5.4 Reproduction model 2: A Limited Recurrent Neural Network

No further research is needed for this model because it is proven that the model is biased. The attack class is solely based on the source and/or destination address.

5.5 Reproduction model 3: SAE with a RNN

Reproduction was successful and therefor generalization is performed as the next step for this model. A more generic model should be able to differentiate anomalies from normal network traffic behavior.

- A dropout of 0.5 in after each hidden layer. [7]
- Start training from any dataset.
- Start training from the most accurate dataset.
- Start training from the best dataset but exclude one dataset for evaluation.
- Train a model to detect one attack class.

5.6 Reproduction model 4: Combined CNN with a RNN

Reproduction was also successful for this model, generalization is performed in a more thoughtful manner by including more regularization and less testing cases. The focus is shifted in this model since the previous model has proven that it is able to achieve consistent high detection accuracy within the different training techniques (by starting with training from each dataset and appending the datasets that were easily to combine with).

Reproduction was a success, no problems were encountered.

6

Results

After reproduction and generalizing, each model is evaluated in this section. IDS based on Machine Learning models have specific challenges such as a high false positive rate. These performance metrics must therefor be evaluated with different methods that are able to take the high false positive ratio into account. In addition, there is also the evaluation of the obtained results. Generally, four metrics are often used for evaluation, and these are discussed in the following sections. [2]

6.1 Evaluation metrics

The generalizability of the models must be measured as the balanced accuracy it can achieve with this unseen dataset. A balanced accuracy of 50% means that 50% is classified correctly and 50% is classified wrong, which would mean that the model, or this approach, failed. Another important metric is to take the false positives into account. The reason for that is that a high false positive rate is a known problem for ML based IDS.

- The balanced accuracy is the average of the percentage correctly classified instances of each class.

$$BalancedAccuracy = \frac{1}{2}(\frac{TP}{P} + \frac{TN}{N}) \tag{6.1}$$

- The average accuracy or Precision or Pr is the amount of correctly classified attacks over the total amount of instances classified as attack.

$$Pr = \frac{TP}{P} \tag{6.2}$$

- Sensitivity is also called Recall or Rc. It is the amount of correctly classified attacks or True Positives over the total amount of attacks.

$$Rc = \frac{TP}{TP + FN} \tag{6.3}$$

- F-Measure or F1 is a harmonic combination of the precision and the sensitivity as one metric.

$$F1 = 2\frac{Pr * Rc}{Pr + Rc} \tag{6.4}$$

6.2 Results model 1: SVM

Reproduction results

The expectations for this model are low, the main goal is to get a working environment. The model was evaluated based upon its balanced classification accuracy of the test dataset and its F1-score. For this reason, reproduction performance and generalizability performance is not discussed. The dataset that was used was the CICIDOS2017 dataset.

	Original Results	Reproduction Results	Generalized Results
Balanced accuracy	82,45%	62,79%	-
F1-score	-	63,64%	-

Table 6.1: Results SVM

6.3 Reproduction model 2: Limited RNN

Reproduction results

Training the model with the suggested features results in a accuracy of 97%. However, removing the source and destination IP-addresses results in a accuracy of 36%. This indicates that classification is only based on the source and destination IP-addresses, making the model not relevant for network intrusion detection. The model assigns a label based on whom the traffic came from. The purpose of the model would be to classify attacks on technical properties of the connection, which this model does not do correctly.

The first SoTa model is therefor not useful, no further research in generalizability or other performance metrics are necessary.

	Original Results	Reproduction Results	Generalized Results
Balanced accuracy	84,83%	97,53%	-

Table 6.2: Results SoTa-1

6.4 Reproduction model 3: SAE with a RNN

Reproduction results

The model to be reproduced was trained with the CICIDOS2017 dataset, which resulted in a balanced accuracy of 97,53%. The paper describes an accuracy of 92% [15]. At first, the reproduced model seems to outperform the original model but the paper classifies the different attack types. Instead of a binary classification, by separating anomalies from normal network traffic data. And this explains the higher performance numbers.

	Original Results	Reproduction Results	Generalized Results
Balanced accuracy	92%	97,98%	98,46%
F1-score	-	92,64%	94,52%

Table 6.3: Results SoTa-2

Generizability results

Adding dropout layers at each stage resulted in similar performance as the original model, see Table 6.3.

Since presenting all the 1296 different performance metrics is not possible, aggregated results are presented.

The dataset with the best balanced class weight accuracy when being combined with other datasets: Wednesday-28-02-2018_TrafficForML_CICFlowMeter (99.08%)
The dataset with the best balanced class weight accuracy when using as a starting point: Wednesday-21-02-2018_TrafficForML_CICFlowMeter (99.99%)
Average balanced class weight accuracy of the merged datasets (99.62%)
Average balanced class weight accuracy of the datasets, used to start training (99.62%)

Combining dataset requires a lot more computing power than default deep learning requires since there are 35 datasets to train on (one of the 36 datasets is kept apart for evaluation purposes). 35x35 (1,225) different neural networks are trained. This takes over 2,000 minutes to complete training on the DGX-2 server of UGent. Some datasets were able to easily achieve performance metrics of 99% or higher. When trying to predict the attacks with the dataset that was kept apart (this dataset is the dataset that was able to merge best with the other datasets), the accuracy is 74%. However, the dataset is not balanced [1]. After balancing the dataset the balanced class weight accuracy was only 50%, and a 50% on a binary prediction translates to a complete random prediction. These are signs that the model is overfitting on the training datasets and thus not training correctly.

In the next test, all the datasets were used for training and started with the "Wednesday-21-02-2018_TrafficForML_CICFlowMeter" dataset, since this dataset was the best dataset to begin with. Then the best dataset for combining: "Wednesday-28-02-2018_TrafficForML_CICFlowMeter" included only a fraction of its data (10%) and also note, this training data was not balanced. Evaluation of the performance was done with the other 90% of the dataset. The resulting balanced accuracy of the testing dataset is not good.

	Generalization Results 2
Balanced accuracy	50,25%
F1-score	83,25%

Table 6.4: SoTa-2 slice of dataset Performance

[1] Balanced dataset: 50% of the network data is attack data and the other 50% of the data is normal traffic flow

The next step is to find the limitations that make the model behave randomly, therefor in the next step the model is trained on only one attack class and one dataset is left out for evaluation purposes. These tests include: training the model by attack class and training by attack class with the limitation that there must be at least 50 instances of the same attack class in the dataset. All of these attempts failed, resulting in a balanced class weight accuracy of 50%.

6.5 Reproduction model 4: Combined CNN with a RNN

Reproduction results

The last reproduction model is a very big model, each following LuNet block is bigger than the previous LuNet block. This results in a very big neural net that expresses random behavior during training. The accuracy during training is either very high or low. Measures were not consistent and varies very much on the data it was fed. Using the whole CICIDOS2017 dataset, results are very good (Table 6.6).

	Original Results	Reproduction Results	Generalized Results
Balanced accuracy	82,78%	97,53%	50,0%
F1-score	-	90,8%	32,83%

Table 6.5: Results SoTa-3

Generizability results

In the generalization step, the model is trained on 2 different DDoS datasets and evaluated on a third DDoS dataset. However, during training, the validation accuracy varies wildly between 43% and 82%, leaning more towards the 43% mark. The testing dataset achieved an accuracy of 50%. This makes the model less generalizable and could be probably due to the fact that the neural network is too large and thus prone for overfitting.

7

Conclusion

The purpose of this thesis was to develop an intrusion detection system that is able to detect attacks from one attack class and to generalize this intrusion detection system to other network environments with similar performance. Previous research has shown that many different machine learning techniques can be used to develop a accurate intrusion detection system. However, there is almost no research done on generalization of intrusion detection systems. One paper can be found that analyses all the different generalization techniques [20] without building a model, which gives an indication of the feasibility of generalization of intrusion detection systems. During the research of this thesis, four different deep learning architecture models were developed to detect intrusions. Every model was able to successfully detect attacks. However, this research shows that even a robust model with good performance over different network intrusion datasets in SoTa overestimates the performance of the model. Network environment has a bigger role than previously thought.

This thesis proofs the choice of model does not affect the generalizability performance over unknown or other similar datasets, since SoTa model exhibits similar results when evaluated in the same circumstances. Although not every possible SoTa model is reproduced and tested, section 8 further mentions other SoTa models that can contribute to the generalizability of a ML based IDS.

8

Future work

This section completes the thesis by mentioning other SoTa models that can contribute to the generalizability of ML IDS models. These models are analysed on the basis of a paper on how they can contribute to create a better and thus more generalizable ML IDS model; in a different way than previously analysed SoTa models. Researche on generalizability of SoTa IDS models is therefor not completed.

8.1 DBN with RBM

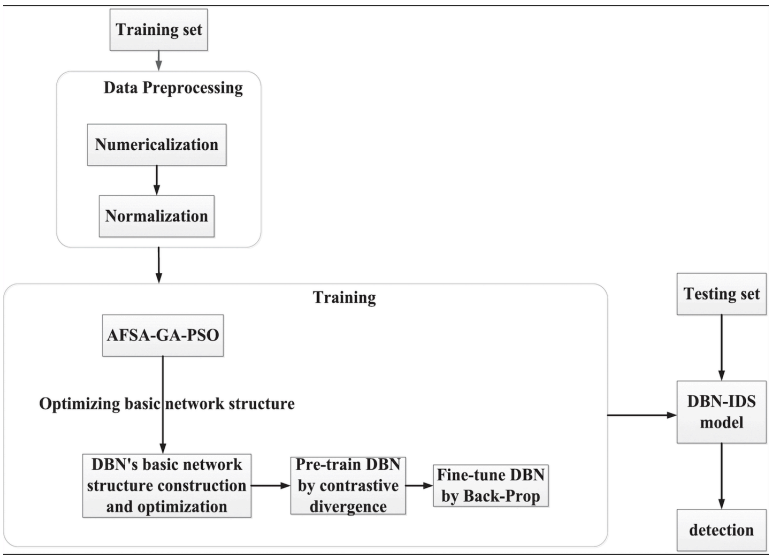

Figure 8.1: Deep Belief Network architecture

A Deep Belief Network or DBN is a deep neural network classifier, composed of multiple Restricted Boltzmann Machines or RBMs and a Back Propagation Neural Network or BPNN, which are explained in more detail in section 2.1.15 "Deep Belief Networks with Restricted Boltzmann Machines". It is an common method for network intrusion detection, the multi-hidden layer neural network in the DBN model is optimized by a Genetic Algorithm or GA, using a artificial fish swarm algorithm - particle swarm optimization or AFSA-PSO. It optimizes the basic structure of the DBN-IDS by generating the parameters of the model automatically. The particle swarm algorithm generates multiple sets of initial network structures, shown in Figure 4.3. The AFSA-PSO method is used to find the best fitness particles, which can be used to construct the network structure. After that, the pre-training procedure consists of a learning stack of RBMs, while the DBN is further fine-tuned by using back propagation of error derivatives. [11]

The described AFSA-PSO algorithm requires a fitness function [1], which is used to evaluate the parameters. It is defined as follows: [11]

$$\begin{aligned} Fitness = {} & (1 - a - b - c - d - e) * ERRate \\ & + a * \frac{n}{maxdim} + b * \frac{sum(x_i)}{n * smax} \\ & + c * FPRate + d * FNRate + d * FNRate + e * DRate \end{aligned} \tag{8.1}$$

Note, a, b, c, d, e ϵ $[0, 1]$ indicate the weight parameter. $ERRate$ is the error rate, defined as $ERRate = 1 - Accuracy$. n is the total number of DBN hidden layers and $maxdim$ indicates

[1] A lower fitness means a better optimized network structure. [11]

the maximum amount of hidden layers. $smax$ is the maximum of allowed neurons in a hidden layer. $sum(x_i)$ is the total amount of hidden layer which are present in the DBN. $FPRate$ is the false negative rate and $DRate$ is the detection rate. $Accuracy, FPRate, FNRate$ and $DRate$ are used as evaluation indicators for the optimization algorithm performance in the intrusion detection classification model.

$$FNRate = \frac{FN}{TN + FN} \tag{8.2}$$

$$FPRate = \frac{FP}{TP + FP} \tag{8.3}$$

$$DRate = \frac{FN}{TP + FN} \tag{8.4}$$

$$FNRate = \frac{TP + TN}{TP + TN + FP + FN} \tag{8.5}$$

8.1.1 AFSA Optimization PSO Algorithm

The artificial fish swarm algorithm - particle optimization algorithm or AFSA-PSO is a strategy that is based upon animal autonomy. The basic idea is that the food with the most fish accumulation is also very rich. Artificial fish are constructed according to the above characteristics, which simulate fish swarm clusters through the virtual waters of the fish swarm to achieve optimization. The AFSA-PSO algorithm is in detail described in Appendix C.

8.2 Generative Adversarial Neural Networks

Researchers have shown that ML models are vulnerable to adversarial perturbations through which an adversary can cause IDSs to malfunction by introducing small impracticable perturbation in the network traffic data. Generative Adversarial Networks or GANs are able to perform adversarial ML attacks that are able to successfully evade a ML based IDS. Researchers also have shown that GANs can be used to inoculate the IDS, making it more robust to adversarial perturbations while also improving the generalizability of the model. [1] A GAN based adversarial defense includes a generative model in the IDS pipeline shown in Figure 8.2. The IDS is trained on the input data and also on generated adversarial examples generated by the generative model. The proposed solution in paper [1] claims that a GAN can detect known

and unknown adversarial perturbations. The results in Figure 8.3 show a clear improvement in the detection rate for multiple ML IDS models. This indicates that a GAN is a must-have for defense against adversarial perturbations. Although the claims of this paper must be taken with a grain of salt since the models are trained on only one attack class and the input data came from the KDD99 dataset.

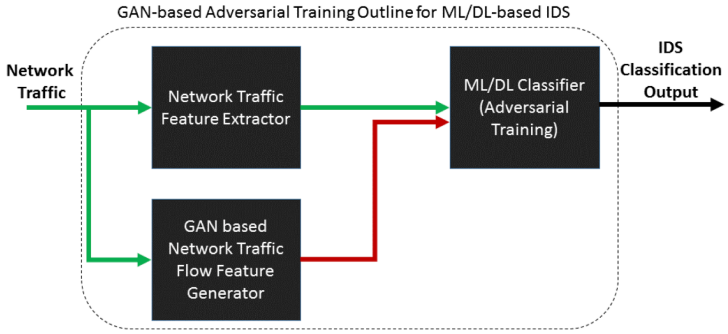

Figure 8.2: Outline of GAN-based adversarial training for ML/DL-based IDSs [1].

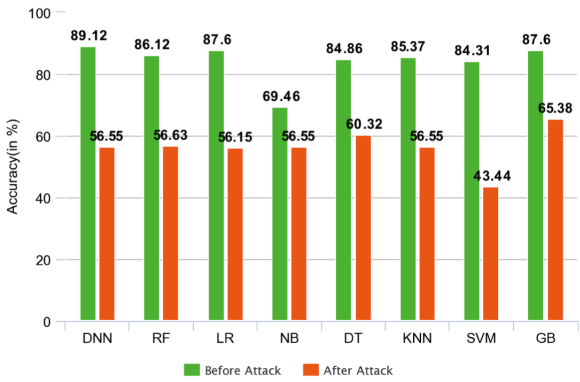

Figure 8.3: Performance evaluation of our GAN-based adversarial attack framework (Previous attack approaches are not shown as a baseline of comparison as they are not applicable in our settings as they, unlike our approach, do not ensure the preservation of networking functional behavior) [1].

Bibliography

[1] S. L. J. Q. A.-A.-F. Muhammed Usama, Muhammed Asim, "Generative Adversarial Networks For Launching and Thwarting Adversarial Attacks on Network Intrusion Detection Systems," pp. 78–344, 2019.

[2] A. H. L. Iman Sharafaldin and A. A. Ghorbani, "Towards Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," pp. 108–116, 2018.

[3] "Firewall." [Online]. Available: https://www.forcepoint.com/cyber-edu/firewall

[4] "Antivirus." [Online]. Available: https://www.verizon.com/info/definitions/antivirus/

[5] "Intrusion Detection Systems definition." [Online]. Available: https://www.barracuda.com/glossary/intrusion-detection-system

[6] A. H. C. T. Elike Hodo, Xavier Bellekens and R. Atikinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey," 2017.

[7] J. H. Fahimeh Farahnakian, "A Deep Auto-Encoder based Approach for Intrusion Detection System," p. 178, 2018.

[8] K. R. Sara A. Althubiti, Eric Marcell Jones Jr., "LSTM for Anomaly-Based Network Intrusion Detection," 2018.

[9] P. Wu and H. Guo, "LuNet: A Deep Neural Network for Network Intrusion Detection," 2019.

[10] J. W. Y. L. L. C. Yu Yan, Lin Qi, "A Network Intrusion Detection Method Based on Stacked Autoencoder and LSTM," 2020.

[11] Z. Z. T. H. Z. L. Peng Wei, Yufeng Li and D. Liu, "An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network," 2019.

[12] U. T. E. S. P. Preeti Mishra, Vijay Varadharajan, "A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection," pp. 686–728, 2018.

[13] "Bag of Words example." [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[14] "Fuzzy Logic." [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm

[15] "Stacked Autoencoder." [Online]. Available: https://medium.com/@venkatakrishna.jonnalagadda/sparse-stacked-and-variational-autoencoder-efe5bfe73b64

[16] "Class imbalance." [Online]. Available: https://machinelearningmastery.com/what-is-imbalanced-classification/

[17] "Simple introduction to Convolutional Neural Networks." [Online]. Available: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

[18] "CNN." [Online]. Available: https://www.geeksforgeeks.org/difference-between-ann-cnn-and-rnn/

[19] B. L. G. M. V. R. C. Houssam Zenati, Chuan-Sheng Foo, "Efficient GAN-Based Anomaly Detection," pp. 1–7, 2018.

[20] K. A. M. M. Chathurika S. Wickramasinghe, Daniel L. Marino, "Generalization of Deep Learning for Cyber-Physical System Security: A Survey," 2018.

[21] "What is meterpreter?" [Online]. Available: https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/

[22] D. N. Moustafa, "UNSW-B15 Dataset," 2018. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[23] V. R. Abhishek Verma, "Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using Distance-based Machine Learning," 2018.

[24] S. B. Ranjit Panigrahi1, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," pp. 479–482, 2018.

[25] "Feature selection: correlation between attributes." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

[26] "TensorFlow library." [Online]. Available: https://www.tensorflow.org/api_docs/python/tf

[27] "Sklearn Library." [Online]. Available: https://scikit-learn.org/stable/modules/classes.html

[28] F. A. S. D. S. Bisyron Wahyudi Masduki, Kalamullah Ramli, "Study on Implementation of Machine Learning Methods Combination for Improving Attacks Detection Accuracy on Intrusion Detection System (IDS)," 2015.

[29] W. Lee and S. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," pp. 227–261, 2000.

[30] "The Law of Moore." [Online]. Available: https://www.intel.com/content/www/us/en/silicon-innovations/moores-law-technology.html

[31] scikit-learn developers, "LinearSVC," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[32] "Validation loss explained." [Online]. Available: https://stackoverflow.com/questions/36963054/what-is-train-loss-valid-loss-and-train-val-mean-in-nns

Appendix

Appendix A - Summary of CICIDS2017

Details of CICIDS2017 files [24]

Filename	Labels
Monday-WorkingHours.pcap_ISCX.csv	Benign
Tuesday-WorkingHours.pcap_ISCX.csv	Benign, FTP-Patator, SSH-Patator
Wednesday-workingHours.pcap_ISCX.csv	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign, Web Attack–Brute Force, Web Attack – Sql Injection, Web Attack–XSS
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Benign, Infiltration
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign, Bot
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign, PortScan
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Benign, DDoS

Appendix B - Summary of CIDDS-001

Attributes of CIDDS-001 [23]

Attribute Name	Description
Src IP	Source IP Address
Src port	Source port
Dest IP	Destination IP Address
Dest port	Destination port
Proto	Transport protocol
Date first seen	Start time flow first seen
Duration	Duration of the flow
Bytes	Number of transmitted bytes
Packets	Number of transmitted packets
Flags	OR concatination of all TCP Flags
Class	Class label (Normal, Attacker, Victim, Suspicious and Unknown)
AttackType	Type of attack (PortScan, DoS, Bruteforce, PingScan)
AttackID	Attacks from the same type get the same attackID
AttackDescription	Details of the attack (e.g. password attempts)

Appendix C - AFSA-PSO algorithm

Algorithm 1: AFSA Optimization PSO Algorithm

input : Field of view: Visual

Moving step: Step

Number of trials of foraging behavior: $trytimes$

Maximum fitness: $maxfit$

Number of interations: $iter = 0$;

The maximum number of iterations of AFSA-PSO: $fmaxiter$

output: Initial optimization solution set $temppbest$

Step 1: Parameter initialization

1. Randomly initialize the particle swarm $swarm$ according to the n value.
2. Solve the $swarm$ fitness $fswarm$
3. Initialize $pbest$ to the optimal fitness particle in the $swarm$.
4. Initialize $gbest$ to the optimal fitness particle in the $swarm$.
5. Solve the $gbest$ fitness $fgbest$.
6. Initialize the extremum record matrix $tempbest_i$ of the i^{th} particle $pbest$ and its fitness matrix $ftempbest_i$.

Step 2: End condition judgment

if $iter <= fmaxiter || fgbest <= maxfit$ **then**
    Jump(Step 3);
else
    Jump(Step 7);
end

Step 3: Particles perform cluster behavior

7. Detect the neighborhood particle center x_j;

if $Fitness_i > Fitness_j$ **then**
    $x_i^{t+1} = x_i^t + (x_j + x_i^t) * rand$;
    Jump(Step 2);
else
    Jump(Step 4);
end

Step 4: Particles perform foraging behavior

while *While trials* $< trytimes$ **do**
    $x_k = x_i^t + 2 * (rand - 0.5).Visual$;
    if $Fitness_i > Fitness_k$ **then**
        $x_i^{t+1} = x_i^t + 2 * (rand - 0.5).Step.(x_k - x_i^t)$;
        Jump(Step 2);
    else
        if $forwardCondition(x_k = Rand)$ **then**
            $break$;
        end
    end
end

Algorithm 2: (Continued) AFSA Optimization PSO Algorithm

Step 5: Particles perform random behavior

8. Randomly selects new particle in the field of view
$x_i^{t+1} = x_i^t + 2 * (rand - 0.5).Step$;

Step 6: Update and record particle swarm related properties

9. Update particle intertia *weight*, the learning factors c_1 and c_2, particle swarm *velocity* and *position*, *pbest*, *gbest*, $fpbest$ and $fgbest$.;

10. Record each particle *pbest* to *temppbest*, let $iter = iter + 1$ && Jump(Step 2);

Step 7: Get global optimization solution

11. Use *temppbest* as a initial optimization solution set

With $Visual$ denoting the fish his field of behavior:

$$x_i^{t+1} = x_i^t + 2 * (rand - 0.5).Step.(x_k - x_i^t) \tag{6}$$

Where *Step* denoting the step size of the fish:

$$x_i^{t+1} = x_i^t + 2 * (rand - 0.5).Step \tag{7}$$