Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης & Συστηματων Πληροφορικης

Ανίχνευση Ανωμαλιών Zero-day με Χρήση AutoEncoders

Διπλωματικη Εργασια

Νικολαος Μπαζωτης

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2023

National Technical University of Athens

School of Electrical and Computer Engineering
Division of Communications, Electronics & IT Systems

Zero-day Anomaly Detection using AutoEncoders

Diploma Thesis

Nikolaos Bazotis

Supervisor: Symeon Papavassiliou
Professor, National Technical University of Athens

Athens, July 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης & Συστηματων Πληροφορικης

Ανίχνευση Ανωμαλιών Zero-day με Χρήση AutoEncoders

Διπλωματικη Εργασια

Νικολαος Μπαζωτης

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Ιουλίου 2023

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Συμεών Παπαβασιλείου	Γεώργιος Στάμου	Ευστάθιος Συκάς
Καθηγήτης Ε.Μ.Π.	Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

..
Αθήνα, Ιούλιος 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης & Συστηματων Πληροφορικης

(Υπογραφή)

.....
Νικόλαος Μπαζώτης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι επιθέσεις Zero-day έχουν γίνει ολοένα και πιο εξελιγμένες τα τελευταία χρόνια, αποτελώντας σημαντική απειλή τόσο για επιχειρήσεις όσο και για ιδιώτες. Η επικινδυνότητα αυτών των επιθέσεων βρίσκεται στην εκμετάλλευση αγνώστων ευπαθειών στο λογισμικό ή το υλικό, κάτι που καθιστά τα παραδοσιακά μέτρα ασφάλειας συχνά αναποτελεσματικά. Ως αποτέλεσμα, οι οργανισμοί είναι ευάλωτοι σε παραβιάσεις δεδομένων, οικονομικές απώλειες και ζημία στην υπόληψη τους.

Ανταποκρινόμενοι σε αυτή την αυξανόμενη απειλή, οι ερευνητές και οι επαγγελματίες της κυβερνοασφάλειας στρέφονται προς τις τεχνικές του deep learning όπως τα AutoEncoders. Ειδικότερα, το επίκεντρο βρίσκεται στη μη επιβλεπόμενη μάθηση, η οποία βασίζεται σε μη επισημασμένα δεδομένα. Αυτή η προσέγγιση παρακάμπτει τις προκλήσεις και τις πιθανές ανακρίβειες που συνδέονται με την επισήμανση των δεδομένων, προσφέροντας έτσι μια πιο αποτελεσματική μηχανισμό άμυνας ενάντια στις επιθέσεις Zero-day.

Αυτή η διπλωματική εργασία έχει ως στόχο την υλοποίηση ενός Συστήματος Ανίχνευσης Απειλών Δικτύου χρησιμοποιώντας AutoEncoders, με στόχο την πρόωρη ανίχνευση των επιθέσεων Zero-day προτού οι χάκερς προβούν σε κάποια κακόβουλη ενέργεια. Το προτεινόμενο σύστημα θα χρησιμοποιήσει το δαταση CICIDS 2017, ένα ευρέως αναγνωρισμένο βενσημαρκ για την αξιολόγηση των συστημάτων ανίχνευσης εισβολών, που περιέχει δεδομένα κίνησης δικτύου που προσομοιώνουν διάφορους τύπους επιθέσεων.

Μια ουσιαστική συνιστώσα αυτής της εργασίας περιλαμβάνει πειραματισμό με διάφορους τύπους AutoEncoders, όπως οι Denoising AutoEncoders, οι Variational AutoEncoders(VAEs), οι Deep AutoEncoders και τα Beta VAEs. Πραγματοποιείται μια συγκριτική μελέτη μεταξύ αυτών των μοντέλων και η απόδοσή τους μετράται έναντι των επιφανειακών μοντέλων από την υπάρχουσα βιβλιογραφία, συγκεκριμένα του SVM σε αυτήν την περίπτωση. Μέσω αυτών των αυστηρών πειραματισμών και συγκρίσεων, αυτή η διατριβή στοχεύει στην προώθηση της τρέχουσας κατανόησης και εφαρμογής των τεχνικών του deep learning στην πρόληψη των επιθέσεων Zero-day.

Λέξεις Κλειδιά:

Νευρωνικά Δίκτυα, Μηχανική Μάθηση, Βαθιά Μάθηση, Σύστημα Ανίχνευσης Διείσδυσης, Σύστημα Πρόληψης Διείσδυσης, Επιθέσεις Zero-day, Δίκτυο, Κυβερνοασφάλεια

Abstract

Zero-day attacks have become increasingly sophisticated in recent years, presenting a significant threat to both businesses and individuals. The danger lies in these attacks exploiting unknown vulnerabilities in software or hardware, making traditional security measures often ineffective. Consequently, organizations are left vulnerable to data breaches, financial losses, and reputational damage.

In the wake of this escalating threat, cyber security researchers and practitioners are turning to deep learning techniques such as AutoEncoders. Particularly, the focus is on unsupervised learning, which relies on unlabeled data. This approach circumvents the challenges and potential inaccuracies associated with data labeling, thereby offering a more effective defense mechanism against zero-day attacks.

This diploma thesis sets out to implement a Network Threat Detection System using AutoEncoders, targeting early detection of zero-day attacks before they can be exploited. The proposed system will leverage the CICIDS 2017 dataset, a well-recognized benchmark for evaluating intrusion detection systems, encompassing network traffic data that simulates diverse types of attacks, including zero-day attacks. Training the deep learning model on this dataset enables the system to identify anomalies in network traffic data and flag potential zero-day attacks.

An integral part of this work involves experimenting with different kinds of AutoEncoders, such as Denoising AutoEncoders, Variational AutoEncoders (VAEs), Deep AutoEncoders, and Beta VAEs. A comparative study is carried out between these models, and their performance is measured against shallow models from existing literature, specifically SVM in this case. Through such rigorous experimentation and comparison, this thesis aims to advance the current understanding and application of deep learning techniques in preventing zero-day attacks.

Keywords: neural networks, machine learning, Deep Learning, Intrusion Detection System, Intrusion Prevention System, feature engineering, Unsupervised Learning, Autoencoders, zero-day attacks, cybersecurity

Ευχαριστίες

Η συγγραφή της παρούσας Διπλωματικής Εργασίας σηματοδοτεί την ολοκλήρωση των προπτυχιακών μου σπουδών. Πριν κλείσει το μεγάλο αυτό κεφάλαιο της ζωής μου, θα ήθελα να ευχαριστήσω τα άτομα που στάθηκαν δίπλα μου και συνέβαλαν στην μέχρι τώρα πορεία μου.

Θα ήθελα να ευχαριστήσω τον ομότιμο καθηγητή κ. Μάγκλαρη για την επίβλεψη της παρούσης εργασίας, καθώς και τον υποψήφιο διδάκτορα Νίκο Κωστόπουλο για τις χρήσιμες συμβουλές του.

Επίσης, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους φίλους μου, με τους οποίους έχω μοιραστεί πολλές από τις πιο ξεχωριστές στιγμές της ζωής μου. Ήταν αυτοί που μου χάρισαν πολύ όμορφες στιγμές και έκαναν αυτά τα χρόνια αξέχαστα.

Τέλος θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου, για την στήριξη που μου παρείχε κατά την εκπόνηση αυτής της εργασίας, αλλά και για την βοήθεια που μου προσέφερε καθ΄ όλη την διάρκεια των σπουδών μου. Αναγνωρίζω τους κόπους που έχουν καταβάλει και ελπίζω μια μέρα να τους το ανταποδώσω. Ιδιαίτερα θέλω να αφιερώσω την παρούσα διπλωματική στην μητέρα μου που δεν βρίσκεται πια στην ζωή αλλά θα ήθελα όσο τίποτα να της δείξω ότι η αγωνία της και οι κόποι της δεν πήγαν χαμένοι,ξέρω ότι θα ήταν ο πιο περήφανος άνθρωπος στον κόσμο αυτή τη στιγμή.

Contents

Εκτεταμένη Περίληψη

Εισαγωγή

Επιθέσεις Zero Day

Οι επιθέσεις Zero-day ονομάζονται εκείνες οι οποίες εκμεταλλεύεται τρωτά σημεία σε λογισμικό ή hardware που είναι προηγουμένως άγνωστα στους προγραμματιστές. Αυτές οι επιθέσεις είναι ιδιαίτερα επικίνδυνες επειδή τα παραδοσιακά μέτρα ασφαλείας είναι συχνά αναποτελεσματικά εναντίον τους, αφήνοντας τους οργανισμούς ευάλωτους σε παραβιάσεις δεδομένων, οικονομικές απώλειες και ζημιά στο κύρος τους.

Σύμφωνα με πρόσφατη έκθεση της Symantec, ο αριθμός των τρωτών σημείων zero-day που εντοπίστηκαν το 2020 αυξήθηκε κατά 55% σε σύγκριση με το προηγούμενο έτος. Αυτή η αύξηση των τρωτών σημείων Zero-day υποδηλώνει ότι οι εισβολείς γίνονται πιο εξελιγμένοι και βρίσκουν νέους τρόπους για να εκμεταλλευτούν άγνωστες ευπάθειες σε λογισμικό και υλικό.

Ο αντίκτυπος των επιθέσεων Zero-day μπορεί να είναι σοβαρός, ιδιαίτερα για τις επιχειρήσεις και τους οργανισμούς που βασίζονται στην τεχνολογία για τη πραγματοποίηση των εργασιών τους. Οι παραβιάσεις δεδομένων που προκύπτουν από επιθέσεις Zero-day μπορεί να οδηγήσουν σε κλοπή ευαίσθητων πληροφοριών, συμπεριλαμβανομένων οικονομικών και προσωπικών δεδομένων, και μπορεί να έχουν ως αποτέλεσμα σημαντικές οικονομικές απώλειες για τις επιχειρήσεις. Επιπλέον, η ζημιά στη φήμη που προκαλείται από παραβίαση δεδομένων μπορεί να είναι μακροχρόνια και μπορεί να οδηγήσει σε απώλεια πελατών και εσόδων.

Προκειμένου να μετριαστεί ο αντίκτυπος των επιθέσεων Zero-day, είναι σημαντικό για τους οργανισμούς να εφαρμόσουν αποτελεσματικά μέτρα αντιμετώπισης. Αυτό περιλαμβάνει την τακτική ενημέρωση λογισμικού και υλικού για την αντιμετώπιση γνωστών τρωτών σημείων και την εφαρμογή προηγμένων μέτρων ασφαλείας, όπως συστήματα ανίχνευσης και πρόληψης εισβολών. Επιπλέον, η χρήση τεχνικών βαθιάς μηχανικής μάθησης, όπως οι AutoEncoders, μπορεί να προσφέρει αποτελεσματική άμυνα έναντι των επιθέσεων Zero-day, εντοπίζοντας πιθανές ευπάθειες πριν από την αξιοποίησή τους.

Συνολικά, ο αυξανόμενος αριθμός τρωτών σημείων Zero-day που ανακαλύπτονται κάθε χρόνο υπογραμμίζει την ανάγκη οι οργανισμοί να παραμείνουν σε επαγρύπνηση

και να εφαρμόσουν ισχυρά μέτρα ασφαλείας για την προστασία από επιθέσεις στον κυβερνοχώρο.

Στόχοι

Ο πρωταρχικός στόχος αυτής της διπλωματικής εργασίας είναι να εφαρμόσει ένα Σύστημα Ανίχνευσης Απειλών Δικτύου που βασίζεται σε AutoEncoders για τον εντοπισμό και την πρόληψη επιθέσεων Zero-day. Το Σύστημα Ανίχνευσης Απειλών Δικτύου που χρησιμοποιεί AutoEncoders στοχεύει να παρέχει έναν αποτελεσματικό αμυντικό μηχανισμό έναντι επιθέσεων Zero-day εντοπίζοντας πιθανές ευπάθειες προτού μπορέσουν να χρησιμοποιηθούν.

Οι ειδικοί στόχοι αυτής της διπλωματικής εργασίας περιλαμβάνουν την εφαρμογή ενός συστήματος ανίχνευσης εισβολής (IDS) με χρήση AutoEncoders για τον εντοπισμό ανωμαλιών στα δεδομένα κίνησης δικτύου που θα μπορούσαν να υποδηλώνουν επίθεση Zero-day. Η αξιολόγηση της αποτελεσματικότητας του Συστήματος Ανίχνευσης Απειλής Δικτύου δοκιμάζεται στο σύνολο δεδομένων CICIDS 2017, το οποίο είναι ένα ευρέως χρησιμοποιούμενο σύνολο δεδομένων αναφοράς για την αξιολόγηση συστημάτων ανίχνευσης εισβολής.

Η σύγκριση της απόδοσης του προτεινόμενου συστήματος ανίχνευσης απειλών δικτύου με χρήση AutoEncoders εξετάζεται και σε σχέση με άλλα συστήματα ανίχνευσης εισβολής και παραδοσιακά μέτρα ασφαλείας για την αξιολόγηση της αποτελεσματικότητάς του στην πρόληψη επιθέσεων Zero-day, διερευνώντας τον αντίκτυπο διαφόρων παραμέτρων στην απόδοση του προτεινόμενου Συστήματος Ανίχνευσης Απειλών Δικτύου, όπως ο αριθμός των κρυφών επιπέδων στον αυτόματο κωδικοποιητή και το μέγεθος του συνόλου δεδομένων εκπαίδευσης και η παροχή πληροφοριών σχετικά με την εφαρμογή τεχνικών βαθιάς μάθησης, όπως π.χ. AutoEncoders στην ασφάλεια στον κυβερνοχώρο και δυνατότητες για μελλοντική έρευνα σε αυτόν τον τομέα.

Στην προσπάθεια μας να παρέχουμε την λύση αυτή βασιζόμαστε σε unsupervised learning και στην χρήση όχι μόνο ενός μοντέλου AutoEncoder αλλά και άλλων όπως οι Denoising Autoencoders και οι Variational Autoencoders. Εξερευνόντας αυτό το πεδίο θα αντλήσουμε γνώση για το ποιες τεχνικές και ποια μοντέλα είναι καταλληλότερα για τον τομέα της ανίχνευσης ανωμαλιών στο δίκτυο.

Συνολικά, η εφαρμογή του Συστήματος Ανίχνευσης Απειλών Δικτύου χρησιμοποιώντας AutoEncoders έχει τη δυνατότητα να βελτιώσει σημαντικά την ασφάλεια των οργανισμών παρέχοντας αποτελεσματική άμυνα έναντι επιθέσεων Zero-day. Με την επίτευξη αυτών των συγκεκριμένων στόχων, η παρούσα διπλωματική εργασία στοχεύει να συμβάλει στον τομέα της ασφάλειας στον κυβερνοχώρο και να παρέχει πληροφορίες σχετικά με την εφαρμογή τεχνικών βαθιάς μάθησης για την πρόληψη επιθέσεων Zero-day.

Θεωρητικό υπόβαθρο

Ανάλυση της κίνησης δικτύου

Η ανάλυση κίνησης δικτύου, ζωτικής σημασίας για την ασφάλεια του δικτύου και την παρακολούθηση της απόδοσης, μπορεί να εξεταστεί σε διαφορετικά επίπεδα, όπως πακέτα και ροές, το καθένα από τα οποία προσφέρει μοναδικές πληροφορίες για τη συμπεριφορά του δικτύου.

Η ανάλυση της κίνησης δικτύου σε επίπεδο πακέτων περιλαμβάνει τη σύλληψη και τον έλεγχο μεμονωμένων πακέτων που διασχίζουν το δίκτυο, προσφέροντας μια λεπτομερή προβολή της δραστηριότητας του δικτύου. Αυτό επιτρέπει την ακριβή επιθεώρηση της στοίβας πρωτοκόλλων δικτύου και τον εντοπισμό ανωμαλιών ή απειλών ασφαλείας. Ωστόσο, η ανάλυση πακέτων είναι πολύπλοκη, και απαιτεί βαθιά τεχνογνωσία στη δικτύωση, την ασφάλεια και την ανάλυση δεδομένων για να διακρίνει κανείς τα αποτελέσματα και να εντοπίσει χρήσιμες πληροφορίες.

Αντίθετα, η ανάλυση της κίνησης του δικτύου με βάση τις ροές συνεπάγεται τη συγκέντρωση δεδομένων σε επίπεδο πακέτων σε ροές δικτύου υψηλότερου επιπέδου, που αντιπροσωπεύουν την επικοινωνία μεταξύ δύο τελικών σημείων για ένα ορισμένο χρονικό διάστημα. Αυτή η μέθοδος παρέχει μια αφηρημένη, συμπαγή άποψη της δραστηριότητας του δικτύου, διευκολύνοντας την ερμηνεία και την οπτικοποίηση δεδομένων.

Όταν πρόκειται για συστήματα ανίχνευσης εισβολών, αυτά εξαρτώνται από τη λήψη και την ανάλυση δεδομένων κίνησης δικτύου για τον εντοπισμό και την απόκριση σε απειλές ασφαλείας. Αρχικά, τα πακέτα δεδομένων συλλαμβάνονται και μετατρέπονται σε μια χρησιμοποιήσιμη μορφή για περαιτέρω ανάλυση, η οποία είναι ζωτικής σημασίας για τον εντοπισμό και την αποτελεσματική απάντηση σε απειλές ασφαλείας.

Εν τω μεταξύ, η ανάλυση της κυκλοφορίας του δικτύου με βάση τις ροές παρέχει μια πιο λεπτομερή εικόνα της δραστηριότητας του δικτύου από την απλή εξέταση ακατέργαστων πακέτων. Επιτρέπει την παρακολούθηση μεμονωμένων ροών και την ανάλυση της συμπεριφοράς τους με την πάροδο του χρόνου. Ωστόσο, ενδέχεται να μην καταγράφει όλους τους τύπους επιθέσεων, ειδικά αυτούς που περιλαμβάνουν μεμονωμένα πακέτα. Επίσης, βασίζεται σε μεγάλο βαθμό στην ποιότητα των δεδομένων ροής, τα οποία, εάν είναι ανακριβή ή ελλιπή, μπορεί να οδηγήσουν σε ψευδώς θετικά ή αρνητικά.

Ωστόσο, η ανάλυση που βασίζεται στη ροή φαίνεται να είναι η καταλληλότερη για συστήματα ανίχνευσης εισβολής λόγω της ανώτερης επεκτασιμότητας σε σύγκριση με τις μεθόδους που βασίζονται σε πακέτα. Είναι πιο αποδοτικό ως προς τους πόρους, λαμβάνοντας υπόψη την ταχεία αύξηση των ταχυτήτων κίνησης στο δίκτυο. Επιπλέον, μπορεί να καταγράψει πολύπλοκες επιθέσεις που εκτείνονται σε πολλαπλά πακέτα πιο αποτελεσματικά, προσφέροντας μια πιο ολιστική άποψη της κυκλοφορίας του δικτύου.

Συστήματα Ανίχνευσης Εισβολής

Ένα σύστημα ανίχνευσης εισβολής (IDS) διαδραματίζει κρίσιμο ρόλο στην ασφάλεια του δικτύου παρακολουθώντας συνεχώς την κυκλοφορία του δικτύου και τις δραστηριότητες του συστήματος για πιθανές απειλές ασφαλείας. Ο κύριος στόχος του IDS είναι να εντοπίζει τυχόν μη εξουσιοδοτημένες ή επιβλαβείς δραστηριότητες σε ένα δίκτυο ή σύστημα αναλύοντας την κίνηση ή τη δραστηριότητα του συστήματος για μη φυσιολογική ή ύποπτη συμπεριφορά. Το IDS μπορεί να ανιχνεύσει μια σειρά επιθέσεων, συμπεριλαμβανομένων των επιθέσεων Denial of Service (DoS), μολύνσεων από κακόβουλο λογισμικό και προσπαθειών μη εξουσιοδοτημένης πρόσβασης.

Τα δεδομένα που χρησιμοποιούνται για την ανάπτυξη IDS μπορούν να κατηγοριοποιηθούν σε τρεις κύριους τύπους. Ο πρώτος τύπος, που βασίζεται σε πεδία πακέτων της κυκλοφορίας δικτύου, παρέχει λεπτομερείς πληροφορίες για την κίνηση του δικτύου. Ο δεύτερος τύπος βασίζεται σε ροή, που περιέχει πληροφορίες που σχετίζονται με την ανταλλαγή πακέτων εντός του δικτύου. Ο τρίτος τύπος περιλαμβάνει σύνολα δεδομένων που μπορεί να έχουν δεδομένα που βασίζονται σε ροή αλλά είναι επίσης εμπλουτισμένα με δεδομένα από τα ίδια τα πακέτα ή εγγραφές συστήματος δικτύου. Η χρήση δεδομένων που βασίζονται σε ροές γίνεται όλο και πιο δημοφιλής λόγω της συμπυκνωμένης μορφής και της επεκτασιμότητας.

Διαφορετικοί τύποι IDS περιλαμβάνουν συστήματα ανίχνευσης εισβολής δικτύου NIDS (Network-IDS), Συστήματα ανίχνευσης εισβολής κεντρικού υπολογιστή HIDS (Host-IDS), συστήματα ανίχνευσης εισβολής που βασίζονται σε υπογραφή SIDS (Signature-IDS) και συστήματα ανίχνευσης εισβολής που βασίζονται σε ανωμαλίες AIDS (Anomaly-IDS). Το NIDS και το HIDS παρακολουθούν την κίνηση σε στρατηγικά σημεία δικτύου και μεμονωμένες συσκευές, αντίστοιχα. Το SIDS συγκρίνει τα πακέτα δικτύου με μια βάση δεδομένων γνωστών κακόβουλων απειλών, παρόμοια με το λογισμικό προστασίας από ιούς. Το AIDS συγκρίνει την κυκλοφορία δικτύου με μια καθιερωμένη γραμμή βάσης για τον εντοπισμό αποκλίσεων, οι οποίες θα μπορούσαν να υποδεικνύουν μη φυσιολογική δραστηριότητα.

Πρόσφατα, η Μηχανική Μάθηση (ML) εισήχθη στο IDS για την αντιμετώπιση των περιορισμών των παραδοσιακών IDS που βασίζονται σε υπογραφές, τα οποία ήταν ικανά να ανιχνεύουν μόνο γνωστές απειλές. Το ML-IDS χρησιμοποιεί αλγόριθμους εκπαιδευμένους σε δεδομένα για την ανίχνευση μοτίβων στην κίνηση δικτύου ή στη δραστηριότητα του συστήματος. Το ML-IDS έχει γίνει δημοφιλές λόγω του αυξανόμενου όγκου και της πολυπλοκότητας της κίνησης δικτύου και του αυξανόμενου αριθμού εξελιγμένων επιθέσεων που μπορούν να αποφύγουν τα παραδοσιακά IDS που βασίζονται σε υπογραφές. Τα IDS που βασίζονται σε ML μπορούν να ανιχνεύσουν ένα ευρύ φάσμα επιθέσεων και μπορούν να μάθουν και να γενικεύσουν σε νέες απειλές, βελτιώνοντας τις δυνατότητές τους ανίχνευσης και απόκρισης.

Επιπλέον, οι τεχνικές βαθίας μηχανικής μάθησης (DL) έχουν εφαρμοστεί στα IDS. Η βαθιά μάθηση και τα νευρωνικά δίκτυα μπορούν να προσδιορίσουν πιο σύνθετα μοτίβα [1], [2] που μπορεί να λείπουν από τα παραδοσιακά ML-IDS, οδηγώντας στην αυξανόμενη δημοτικότητά τους στον χώρο.

Μηχανική μάθηση και νευρωνικά δίκτυα

Η Μηχανική Μάθηση (ML), ένας κρίσιμος τομέας της τεχνητής νοημοσύνης, παρέχει στα συστήματα τη δυνατότητα να μαθαίνουν αυτόνομα και να βελτιώνονται από την εμπειρία χωρίς να προγραμματίζονται ρητά. Αυτός ο δυναμικός κλάδος εκτείνεται στις σφαίρες της επιστήμης των υπολογιστών και της στατιστικής, χρησιμοποιώντας στατιστικά μοντέλα και αλγόριθμους για την εκτέλεση εργασιών κάνοντας προβλέψεις ή αποφάσεις βάσει δεδομένων.

Η Μηχανική Μάθηση (ML) έχει ουσιαστική αξία σε διάφορους τομείς λόγω της ικανότητάς της να εντοπίζει πρότυπα και τάσεις που διαφορετικά θα παρέμεναν αόριστες στην ανθρώπινη ανάλυση. Αυτή η ικανότητα είναι που δίνει τη δυνατότητα στις επιχειρήσεις να αποκτήσουν κρίσιμες γνώσεις σχετικά με τη συμπεριφορά των πελατών και τη λειτουργική δυναμική, οδηγώντας έτσι στη λήψη στρατηγικών αποφάσεων και ενισχύοντας την καινοτομία.

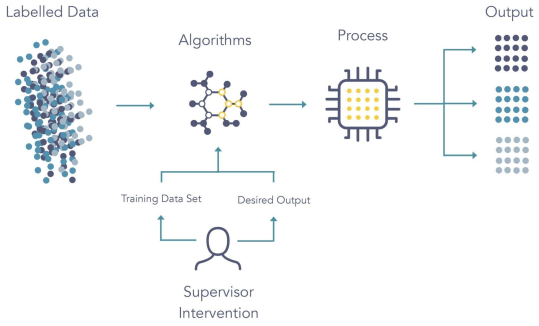
Υπάρχουν τρεις κύριες κατηγορίες μηχανικής μάθησης: το supervised learning, το unsupervised learning και το reinforcement learning. Εν συντομία, το supervised learning συνάγει μια συνάρτηση από δεδομένα εισόδου με ετικέτα για την πρόβλεψη αποτελεσμάτων για μη ορατά δεδομένα. Η μάθηση χωρίς επίβλεψη, αντίθετα, επιδιώκει να εντοπίσει εγγενή πρότυπα και δομές σε δεδομένα χωρίς ετικέτα. Η ενισχυτική μάθηση περιλαμβάνει έναν πράκτορα που μαθαίνει να λαμβάνει αποφάσεις αλληλεπιδρώντας με ένα περιβάλλον, καθοδηγούμενος από σήματα ανταμοιβής και ποινής.

Μια βασική έννοια στις μεθοδολογίες μηχανικής μάθησης είναι η συνάρτηση κόστους, η οποία ποσοτικοποιεί την απόκλιση μεταξύ των προβλέψεων του μοντέλου και των πραγματικών τιμών. Ο κεντρικός στόχος στη μηχανική μάθηση είναι η ελαχιστοποίηση αυτής της συνάρτησης κόστους.

Ωστόσο, η επίτευξη ενός βέλτιστου μοντέλου δεν είναι χωρίς προκλήσεις. Δύο θεμελιώδη ζητήματα είναι η υπερπροσαρμογή, όπου ένα μοντέλο μαθαίνει τον θόρυβο και όχι το μοτίβο, και η υποπροσαρμογή, όπου το μοντέλο είναι υπερβολικά απλοϊκό για να καταγράψει την πολυπλοκότητα των δεδομένων. Στρατηγικές όπως η τακτοποίηση, η έγκαιρη διακοπή και η διασταυρούμενη επικύρωση συμβάλλουν στον μετριασμό αυτών των προβλημάτων, διασφαλίζοντας την ευρωστία και την αξιοπιστία του μοντέλου.

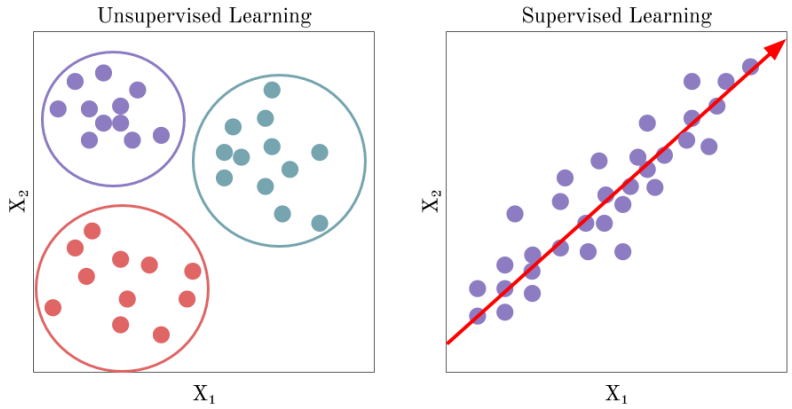
Είδη μηχανικής Μάθησης

Η εποπτευόμενη μηχανική εκμάθηση είναι ένα κρίσιμο συστατικό της ανάλυσης δεδομένων όπου ένας αλγόριθμος εκπαιδεύεται χρησιμοποιώντας ένα προϋπάρχον ¨σύνολο δεδομένων εκπαίδευσης¨. Αυτό το σύνολο δεδομένων παρέχει συγκεκριμένο πλαίσιο και οδηγίες για σωστές και λανθασμένες εξόδους. Το supervised learning είναι ιδιαίτερα εφαρμόσιμη σε προβλήματα ταξινόμησης δεδομένων όπου ο αριθμός των κατηγοριών για ταξινόμηση είναι γνωστός εκ των προτέρων και τα δείγματα έχουν ήδη ταξινομηθεί σε αυτές τις κατηγορίες. Με τη βοήθεια ενός ανθρώπινου επόπτη και επαναληπτικών μεθόδων, ο αλγόριθμος βελτιώνει την απόδοσή του, μειώνοντας το ποσοστό σφάλματος κατά την ταξινόμηση των δεδομένων με την πάροδο του χρόνου.



Σχήμα 1: *supervised learning*

Αντίθετα, η μη εποπτευόμενη μηχανική εκμάθηση χρησιμοποιείται για την εξερεύνηση μη διακριτών δεδομένων όπου ο αλγόριθμος δεν καθοδηγείται από προϋπάρχουσες ετικέτες ή κατηγορίες. Το μοντέλο έχει επιφορτιστεί με την ανακάλυψη της εγγενούς δομής μέσα σε μη επισημασμένα δεδομένα χωρίς ανθρώπινη παρέμβαση. Είναι ιδιαίτερα χρήσιμο όταν δεν υπάρχει προηγούμενη γνώση για πιθανές κατηγορίες ή ομάδες εντός των δεδομένων και ο αλγόριθμος βασίζεται στον εντοπισμό μοναδικών συστάδων με βάση τα εγγενή χαρακτηριστικά τους. Σε αντίθεση με το supervised learning, η μάθηση χωρίς επίβλεψη προσφέρει πληροφορίες για πιθανούς τομείς ενδιαφέροντος, επισημαίνοντας διαφορές, ανωμαλίες ή ακραίες καταστάσεις, επιτρέποντας την ανακάλυψη νέων γνώσεων από τα δεδομένα.



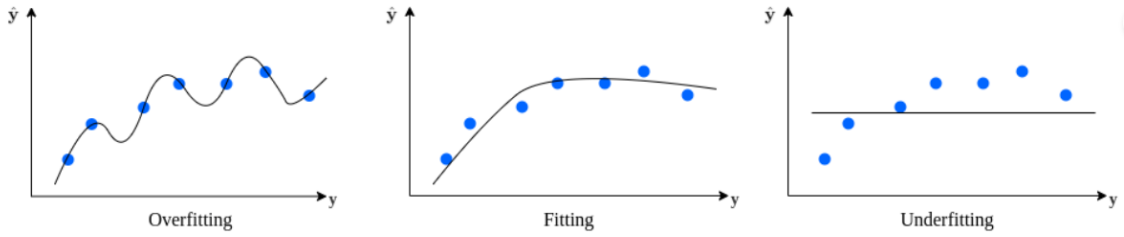
Σχήμα 2: *Διαφορά μεταξύ εποπτευόμενης και μη εποπτευόμενης μάθησης*

Η ενισχυτική μάθηση, ένας άλλος υποτομέας της μηχανικής μάθησης, εστιάζει στην εκπαίδευση ενός πράκτορα για τη μεγιστοποίηση ενός σήματος ανταμοιβής ενώ αλληλεπιδρά με ένα περιβάλλον. Αυτή η προσέγγιση βασίζεται στην επιβράβευση επιθυμητών συμπεριφορών ή/και στην τιμωρία των ανεπιθύμητων, επιτρέποντας στον πράκτορα να μάθει μέσω δοκιμής και λάθους. Ο πρωταρχικός στόχος της ενισχυτικής μάθησης είναι να ορίσει την καλύτερη ακολουθία αποφάσεων που πρέπει να ακολουθήσει ο

πράκτορας για να λύσει ένα πρόβλημα, μεγιστοποιώντας ταυτόχρονα μια μακροπρόθεσμη ανταμοιβή. Βρίσκει εφαρμογές στον σχεδιασμό κίνησης, τη δυναμική διαδρομή, τη βελτιστοποίηση ελεγκτή, τις πολιτικές εκμάθησης βάσει σεναρίων και πολλά άλλα. Ένα κοινό παράδειγμα αποτελεσματικότητας της ενισχυτικής μάθησης είναι η χρήση της στην εκμάθηση πολιτικών αυτόματης στάθμευσης.

Βασικές έννοιες μηχανικής εκμάθησης

Η υπερπροσαρμογή (overfitting) και η υποπροσαρμογή (underfitting) είναι φαινόμενα που συμβαίνουν κατά την εκπαίδευση μοντέλων μηχανικής μάθησης, ζωτικής σημασίας για ένα μοντέλο που γενικεύει καλά σε άγνωστα δεδομένα. Η υπερπροσαρμογή, που χαρακτηρίζεται από ένα υπερβολικά περίπλοκο μοντέλο, έχει καλή απόδοση στα δεδομένα training αλλά κακή σε άγνωστα δεδομένα. Οι τεχνικές για την αποφυγή της υπερπροσαρμογής περιλαμβάνουν τη μείωση της πολυπλοκότητας του μοντέλου, τη χρήση μεθόδων regularization και cross-validation.



Σχήμα 3: *αντιστάθμιση bias-variance*

Η υποπροσαρμογή, αντίθετα, προκύπτει όταν ένα μοντέλο είναι πολύ απλό για να συλλάβει την υποκείμενη δομή δεδομένων. Η υπέρβαση της υποπροσαρμογής μπορεί να περιλαμβάνει την προσθήκη περισσότερων χαρακτηριστικών ή την αύξηση της πολυπλοκότητας του μοντέλου. Η ισορροπία μεταξύ υπερπροσαρμογής και υποπροσαρμογής, που αναφέρεται ως αντιστάθμιση βιασ-αριανσε, είναι κρίσιμη για ένα ισχυρό και αξιόπιστο μοντέλο μηχανικής εκμάθησης.

Οι τεχνικές regularization αντιμετωπίζουν την υπερπροσαρμογή στα μοντέλα μηχανικής εκμάθησης. regularization σημαίνει περιορισμός ενός μοντέλου για την αποφυγή υπερβολικής προσαρμογής προσθέτοντας μια ποινή στη συνάρτηση απώλειας του μοντέλου:

$$Regularization = LossFunction + Penalty$$

Οι τεχνικές κανονικοποίησης(Regularization) περιλαμβάνουν Regularization L2, Regularization L1 και Elastic Net, το οποίο συνδυάζει τους όρους L2 και L1.

$$RidgeRegressionL2CostFunction = LossFunction + \frac{1}{2}\lambda\sum_{j=1}^{m} w_j^2$$

$$LassoRegressionL1CostFunction = LossFunction + \lambda\sum_{j=1}^{m} |w_j|$$

$$ElasticNetCostFunction = LossFunction + r\lambda\sum_{j=1}^{m} |wj| + \frac{(1-r)}{2}\lambda\sum_{j=1}^{m} w_j^2$$

Αυτές οι τεχνικές βοηθούν στον έλεγχο της πολυπλοκότητας του μοντέλου, βελτιώνοντας έτσι την απόδοση του μοντέλου.

Βαθιά μηχανική μάθηση

Η βαθιά εκμάθηση είναι μια επέκταση της κλασικής μηχανικής μάθησης που εισάγει πρόσθετο βάθος, το οποίο σημαίνει βελτιωμένη πολυπλοκότητα εντός του μοντέλου. Περιλαμβάνει τον μετασχηματισμό δεδομένων μέσω πολλαπλών συναρτήσεων, επιτρέποντας την ιεραρχική αναπαράσταση των δεδομένων, σε διάφορα επίπεδα αφαίρεσης. Ένα χαρακτηριστικό πλεονέκτημα της βαθιάς μάθησης είναι η επάρκειά της στην επίλυση περίπλοκων προβλημάτων γρήγορα και με ακρίβεια. Αυτό οφείλεται σε μεγάλο βαθμό στην ικανότητά του να επιτρέπει μαζική παραλληλοποίηση [3].

Η βαθιά εκμάθηση αποτελείται από ένα ευρύ φάσμα στοιχείων όπως συνελίξεις, επίπεδα συγκέντρωσης, πλήρως συνδεδεμένα επίπεδα, πύλες, κελιά μνήμης, λειτουργίες ενεργοποίησης, σχήματα κωδικοποίησης/αποκωδικοποίησης, μεταξύ άλλων, ανάλογα με την αρχιτεκτονική δικτύου που χρησιμοποιείται. Είναι σημαντικό να αναγνωρίσουμε ότι η βαθιά μάθηση επιδεικνύει αξιοσημείωτη ευελιξία και προσαρμοστικότητα, καθιστώντας την κατάλληλη για ένα ευρύ φάσμα σύνθετων προκλήσεων από την οπτική γωνία της ανάλυσης δεδομένων. Αυτό οφείλεται κυρίως στην υψηλή ιεραρχική δομή και την τεράστια ικανότητα μάθησης των μοντέλων βαθιάς μάθησης [3].

Βασικές έννοιες βαθιάς μηχανικής εκμάθησης

Τα τεχνητά νευρωνικά δίκτυα (ANN) είναι υπολογιστικά μοντέλα που βασίζονται σε βιολογικά νευρωνικά δίκτυα σε εγκεφάλους ζώων. Αποτελούνται από μονάδες ή κόμβους, που ονομάζονται τεχνητοί νευρώνες, που μιμούνται τη δομή ενός βιολογικού νευρώνα (Σχήμα 2.4). Αυτοί οι νευρώνες λαμβάνουν μία ή πολλαπλές εισόδους, παρόμοιες με τους δενδρίτες ενός βιολογικού νευρώνα. Οι είσοδοι σταθμίζονται χωριστά (w_1,w_2,...,w_m) και αθροίζονται για τη δημιουργία μιας εξόδου, η οποία στη συνέχεια συγκρίνεται με μια τιμή κατωφλίου μέσω μιας συνάρτησης ενεργοποίησης, Εάν αυτή η τιμή υπερβεί το όριο, ο νευρώνας ενεργοποιείται (Σχήμα 2.5).

Οι νευρώνες σε ένα ANN είναι διατεταγμένοι σε στρώματα και σχηματίζουν συνδέσεις μόνο με νευρώνες στο προηγούμενο και το επόμενο στρώμα. Το δίκτυο λαμβάνει εισόδους μέσω ενός επιπέδου εισόδου και παράγει την τελική έξοδο μέσω ενός

επιπέδου εξόδου. Μπορεί να υπάρχουν μηδέν ή περισσότερα ενδιάμεσα κρυφά επίπεδα (Σχήμα 2.6).

Συναρτήσεις Ενεργοποποίησης Οι συναρτήσεις ενεργοποίησης(activation functions) υπαγορεύουν την έξοδο του νευρώνα στο επόμενο στρώμα. Κατηγοριοποιούνται κυρίως σε γραμμικές και μη γραμμικές, με τις μη γραμμικές συναρτήσεις να χρησιμοποιούνται συχνότερα λόγω της ικανότητάς τους να γενικεύουν σε διάφορους τύπους δεδομένων. Πολλά παραδείγματα λειτουργιών ενεργοποίησης περιλαμβάνουν:

- **Σιγμοειδής συνάρτηση:** Εξάγει μια τιμή μεταξύ 0 και 1.

$$S = \frac{1}{1 + e^{-x}}$$

(Σχήμα 2.7)

- Tanh/Υπερβολική εφαπτομένη συνάρτηση: Εξάγει μια τιμή μεταξύ -1 και 1.

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

(Σχήμα 2.8)

- Συνάρτηση ReLU (Rectified Linear Unit): Επιστρέφει την τιμή εισόδου εάν είναι θετική ή 0 εάν είναι αρνητική.

$$f(x) = \max(x, 0)$$

(Σχήμα 2.9)

- Συνάρτηση Softmax: Μετατρέπει ένα διάνυσμα αριθμών σε διάνυσμα πιθανοτήτων. Χρησιμοποιείται στο τελικό επίπεδο ενός ταξινομητή για προβλήματα ταξινόμησης πολλαπλών κλάσεων.

$$f(x_i) = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}}$$

(Σχήμα 2.10)

- Συνάρτηση δυαδικού βήματος: Εάν η τιμή εισόδου υπερβαίνει ένα συγκεκριμένο όριο, ο νευρώνας ενεργοποιείται και η έξοδος ορίζεται σε 1, διαφορετικά ορίζεται σε 0. (Σχήμα 2.11)

Συναρτήσεις Κόστους Η ενότητα συζητά τις συναρτήσεις κόστους στο πλαίσιο της μηχανικής μάθησης και συγκεκριμένα των νευρωνικών δικτύων. Μια συνάρτηση κόστους ποσοτικοποιεί πόσο αποκλίνει η πρόβλεψη ενός μοντέλου από την πραγματική παραγωγή. Σκοπός του είναι να βοηθήσει το μοντέλο να βελτιώσει την απόδοσή του προσαρμόζοντας τις υπερπαραμέτρους του.

Η ενότητα κατηγοριοποιεί τις κοινές συναρτήσεις κόστους ανάλογα με τη χρήση τους σε διαφορετικές μαθησιακές εργασίες.

Για εργασίες παλινδρόμησης:

1. Το μέσο τετράγωνο σφάλμα (MSE) επιλέγεται συχνά καθώς τετραγωνίζει τη διαφορά μεταξύ των προβλεπόμενων και των πραγματικών τιμών, τιμωρώντας μεγαλύτερα σφάλματα. Παρέχει επίσης ένα σαφές συνολικό ελάχιστο λόγω της κυρτής φύσης του, βοηθώντας τη βελτιστοποίηση της κλίσης κατάβασης. Η εξίσωση του είναι:
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})$$

2. Το μέσο απόλυτο σφάλμα (MAE) υπολογίζει τον μέσο όρο των απόλυτων διαφορών μεταξύ των προβλεπόμενων και των πραγματικών εξόδων. Είναι ωφέλιμο όταν αντιμετωπίζετε μεγάλο αριθμό ακραίων τιμών στα δεδομένα, καθώς μετριάζει τον αντίκτυπό τους. Υπολογίζεται ως εξής:
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Για εργασίες ταξινόμησης:

1. Η απώλεια διασταυρούμενης εντροπίας ή η απώλεια καταγραφής χρησιμοποιείται σε προβλήματα ταξινόμησης δυαδικών και πολλαπλών κλάσεων. Συγκρίνει την πραγματική τιμή με την πιθανότητα η είσοδος να ανήκει σε μια συγκεκριμένη κατηγορία. Η εξίσωση του είναι:
$$L(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\ log(1 - \hat{y}_i)]$$

Για εργασίες που περιλαμβάνουν κατανομές πιθανοτήτων:

1. Η απόκλιση Kullback-Leibler (KL) χρησιμοποιείται για να ποσοτικοποιήσει τη διαφορά μεταξύ δύο κατανομών πιθανοτήτων, που χρησιμοποιούνται γενικά σε μάθηση χωρίς επίβλεψη. Αξιολογεί πώς μια κατανομή πιθανοτήτων αποκλίνει από την αναμενόμενη. Η εξίσωση του είναι:
$$D_{KL}(P||Q) = \sum_{i} P(i) \log \frac{P(i)}{Q(i)}$$

backpropagation Η διαδικασία υπολογισμού των εξόδων είναι γνωστή ως εμπρόσθια διάδοση(forward pass). Εάν η μέτρηση σφάλματος μεταξύ των προβλεπόμενων και των πραγματικών τιμών υπερβαίνει ένα συγκεκριμένο όριο, ξεκινά μια διαδικασία που ονομάζεται αντίστροφη διάδοση (back propagation) για τη βελτιστοποίηση των βαρών στο νευρωνικό δίκτυο για την ελαχιστοποίηση αυτού του σφάλματος (Σχήμα 2.12). Το σφάλμα μεταξύ των προβλεπόμενων και των πραγματικών αποτελεσμάτων ποσοτικοποιείται χρησιμοποιώντας μια συνάρτηση κόστους. Αυτό καθοδηγεί τον αλγόριθμό Βελτιστοποίησης για το πώς να προσαρμόζει τα βάρη και τις προκαταλήψεις κατά τη διάδοση προς τα πίσω.

Αλγόριθμοι Βελτιστοποίησης Οι αλγόριθμοι Βελτιστοποίησης στη μηχανική μάθηση είναι ζωτικής σημασίας για την ελαχιστοποίηση της λειτουργίας απώλειας και τη βελτίωση της απόδοσης του συστήματος. Καθοδηγούν το μοντέλο να τροποποιήσει τα βάρη και τους ρυθμούς εκμάθησης, μειώνοντας τις απώλειες. Το έγγραφο διερευνά διάφορους βελτιστοποιητές, τις λειτουργίες τους, καθώς και τα οφέλη και τις προκλήσεις τους.

- Gradient Descent: Αυτός ο αλγόριθμος βελτιστοποίησης ελαχιστοποιεί τη λειτουργία απώλειας κινούμενος προς την αντίθετη κατεύθυνση από την πιο απότομη κλίση. Χρησιμοποιεί δεδομένα από ολόκληρο το σετ εκπαίδευσης για να υπολογίσει τη διαβάθμιση της συνάρτησης κόστους, η οποία μπορεί να είναι υπολογιστικά ακριβή. Το ποσοστό μάθησης, που καθορίζει το μέγεθος των βημάτων προς το τοπικό ελάχιστο, είναι ζωτικής σημασίας.

$$W_{new} = W_{old} - a * \frac{\partial(Loss)}{partial(W_{old})}$$

- Stohastic Gradient Descent (SGD): Μια παραλλαγή του Gradient Descent, ο SGD ενημερώνει τις παραμέτρους του μοντέλου ξεχωριστά. Ενώ είναι ταχύτερο και απαιτεί λιγότερη μνήμη, μπορεί να οδηγήσει σε θορυβώδεις κλίσεις και υψηλή διακύμανση λόγω συχνών ενημερώσεων.

- Adaptive Gradient Descent (AdaGrad): Εκχωρεί διαφορετικούς ρυθμούς εκμάθησης σε μεμονωμένους νευρώνες μέσα σε κάθε κρυφό επίπεδο, προσαρμόζοντας καθώς προχωρούν οι επαναλήψεις. Παρά το γεγονός ότι είναι αποτελεσματικό με αραιά δεδομένα, μπορεί να υποφέρει από το πρόβλημα του «νεκρού νευρώνα» στα βαθιά νευρωνικά δίκτυα, οδηγώντας σε σημαντικά μειωμένο ρυθμό μάθησης.

$$W_{new} = W_{old} + \frac{a}{\sqrt{cache_{new}} + \varepsilon} * \frac{\partial(Loss)}{\partial(W_{old})}$$

- RMS-Prop: Μια αλλαγή του Adagrad, το RMS-Prop χρησιμοποιεί έναν εκθετικό κινητό μέσο όρο των κλίσεων. Επιτρέπει ανεξάρτητη προσαρμογή των ρυθμών εκμάθησης για κάθε παράμετρο, προσφέροντας μια πιο προσαρμοσμένη προσέγγιση βελτιστοποίησης.

$$cache_{new} = \gamma * cache_{old} + (1 - \gamma)(\frac{\partial(Loss)}{\partial(W_{old})})^2$$

- Adadelta: Μια επέκταση του Adagrad, ο Adadelta αντιμετωπίζει το ζήτημα του μειωμένου ρυθμού μάθησης και λειτουργεί χωρίς να απαιτεί προεπιλεγμένο ρυθμό μάθησης. Ωστόσο, είναι υπολογιστικά ακριβό.

- Adam: Υπολογίζει προσαρμοστικούς ρυθμούς μάθησης για κάθε παράμετρο. Συνδυάζει τον μέσο όρο αποσύνθεσης των προηγούμενων κλίσεων, παρόμοια με την ορμή, και τον αποσυντιθέμενο μέσο όρο των προηγούμενων τετραγωνικών κλίσεων, όπως ο RMS-Prop και ο Adadelta. Ο Adam είναι εύκολο στην εφαρμογή, υπολογιστικά αποδοτικό και απαιτεί ελάχιστη μνήμη.

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{S_{dw_t} - \varepsilon}} * V_{dw_t}$$

$$b_t = b_{t-1} - \frac{\eta}{\sqrt{S_{db_t} - \varepsilon}} * V_{db_t}$$

Για αραιά δεδομένα, συνιστώνται μέθοδοι όπως οι Adam, RMSprop, Adadelta και Adagrad. Οι RMSprop, Adadelta και Adam συχνά αποδίδουν παρόμοια αποτελέσματα καθώς ο Adam είναι ουσιαστικά μια επέκταση του RMSprop, αλλά με πρόσθετα χαρακτηριστικά όπως η διόρθωση μεροληψίας και η ορμή. Σε σενάρια όπου η κλίση γίνεται αραιή, ο Adam γενικά ξεπερνά ο RMSprop.

SVM

Το Support Vector Machine (SVM) είναι ένας ευρέως χρησιμοποιούμενος αλγόριθμος τόσο για προβλήματα παλινδρόμησης όσο και για προβλήματα ταξινόμησης. Στοχεύει να βρει ένα υπερεπίπεδο σε ένα χώρο N-διάστασης (όπου N είναι ο αριθμός των χαρακτηριστικών) που μπορεί να ταξινομήσει τα σημεία δεδομένων. Τα SVM μπορούν να κατηγοριοποιηθούν σε γραμμικά και μη γραμμικά μοντέλα. Τα γραμμικά SVM μπορούν να διαχωρίσουν τα δεδομένα με μια γραμμική γραμμή ή υπερεπίπεδο, ενώ τα μη γραμμικά SVM μετατρέπουν τα δεδομένα σε ένα χώρο χαρακτηριστικών όπου μπορούν να διαιρεθούν γραμμικά.

Μπορεί να υπάρχουν πολλά πιθανά υπερεπίπεδα για να διαχωριστούν δύο κατηγορίες σημείων δεδομένων, αλλά ο στόχος είναι να βρεθεί το υπερεπίπεδο με το μέγιστο περιθώριο ή απόσταση μεταξύ σημείων δεδομένων και από τις δύο κατηγορίες. Τα σημεία δεδομένων που βρίσκονται πιο κοντά στο υπερεπίπεδο ονομάζονται διανύσματα υποστήριξης και επηρεάζουν τη θέση και τον προσανατολισμό του υπερεπίπεδου. Εάν αφαιρεθούν αυτά τα διανύσματα υποστήριξης, η θέση του υπερεπιπέδου θα αλλάξει. Έτσι, το περιθώριο του ταξινομητή μεγιστοποιείται χρησιμοποιώντας αυτά τα Διανύσματα Υποστήριξης.

Η απλούστερη μορφή SVM χρησιμοποιείται για προβλήματα δυαδικής ταξινόμησης. Για τους σκοπούς της ανίχνευσης ανωμαλιών, χρησιμοποιείται μια παραλλαγή που ονομάζεται One-Class SVM (OCSVM). Σε αντίθεση με τα συμβατικά SVM που διαφοροποιούν μεταξύ δύο διακριτών κλάσεων, αυτή η μέθοδος προσδιορίζει ένα διαχωριστικό υπερεπίπεδο που απέχει από την αρχή στον χώρο χαρακτηριστικών, οδηγώντας σε μια συνάρτηση απόφασης που επιστρέφει θετικές τιμές για περιοχές υψηλής πυκνότητας σημείου και αρνητικές τιμές για περιοχές χαμηλής πυκνότητας.

Η εξίσωση για το διαχωριστικό υπερεπίπεδο δίνεται ως εξής:

$$\min_{\mathbf{w},\xi,\rho} \frac{1}{2}||\mathbf{w}||^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i$$
$$s.t. \quad (\mathbf{w}\cdot\phi(\mathbf{x}_i)) \geq \rho - \xi_i \quad and \quad \xi_i \geq 0, \quad i = 1,\ldots,n$$

όπου $\mathbf{w}$ είναι το διάνυσμα βάρους, ρ είναι η μετατόπιση, ξ είναι οι χαλαρές μεταβλητές, $\phi(\mathbf{x}_i)$ είναι το αντιστοιχισμένο σημείο δεδομένων στο χαρακτηριστικό σπαςε, και το ν είναι μια παράμετρος που ελέγχει την αντιστάθμιση μεταξύ της μεγιστοποίησης της απόστασης του υπερεπίπεδου από την αρχή και του κλάσματος των ακραίων τιμών

AutoEncoders

Οι AutoEncoders είναι ένας τύπος νευρωνικού δικτύου που χρησιμοποιείται στην μάθηση χωρίς επίβλεψη για την ανακατασκευή των παρατηρήσεων εισόδου με ελάχιστο σφάλμα. Ο στόχος είναι να μάθουμε μια ενημερωτική αναπαράσταση των δεδομένων [4]. Οι AutoEncoders έχουν τρία κύρια στοιχεία: έναν κωδικοποιητή, μια αναπαράσταση λανθάνοντος χαρακτηριστικού (επίσης γνωστή ως κώδικας, σημείο συμφόρησης ή λανθάνουσα θέση) και έναν αποκωδικοποιητή. Ο κωδικοποιητής δημιουργεί την αναπαράσταση λανθάνοντος χαρακτηριστικού από τα δεδομένα εισόδου και ο αποκωδικοποιητής χρησιμοποιεί αυτήν την αναπαράσταση για να ανακατασκευάσει την είσοδο.

Ο κωδικοποιητής και ο αποκωδικοποιητής μπορούν να αντιπροσωπευτούν από δύο συναρτήσεις, g και f, αντίστοιχα:

$$h_i = g(x_i)$$

$$\hat{x}_i = f(h_i) = f(g(x_i))$$

Η εκπαίδευση ενός AutoEncoder περιλαμβάνει την εύρεση των συναρτήσεων $g(.)$ και $f(.)$ που ικανοποιούν:

$$arg\min_{f,g} < [\Delta(x_i, f(g(x_i))] >$$

Μια κοινή στρατηγική για να αποφευχθεί η εκμάθηση της συνάρτησης ταυτότητας από τον αυτόματο κωδικοποιητή είναι η δημιουργία εμπόδιο μειώνοντας τη διάσταση των λανθάνοντων χαρακτηριστικών ή προσθέτοντας τακτοποίηση. Η διάσταση του

λανθάνοντος χώρου και η ισορροπία μεταξύ της συμπίεσης χαρακτηριστικών και της διατήρησης των απαραίτητων πληροφοριών είναι καθοριστικής σημασίας [4].

Οι αυτοκωδικοποιητές είναι επίσης χρήσιμοι για την ανίχνευση ανωμαλιών, καθώς μπορούν να μάθουν να ανακατασκευάζουν ¨κανονικά' μοτίβα και να παράγουν υψηλά σφάλματα ανακατασκευής για άγνωστα μοτίβα ή ανωμαλίες. Το σφάλμα ανακατασκευής (RE) υπολογίζεται συχνά χρησιμοποιώντας το μέσο τετράγωνο σφάλμα (MSE):

$$RE \equiv MSE = \frac{1}{M}\sum_{i=1}^{M}|x_i - \hat{x}_i|^2$$

Στη ροή κυκλοφορίας δικτύου, ένας AutoEncoder μπορεί να εκπαιδευτεί σε κανονικά δεδομένα κίνησης δικτύου. Οι ανωμαλίες σε αυτό το πλαίσιο θα μπορούσαν να αντιπροσωπεύουν διαφορετικούς τύπους επιθέσεων ή εισβολών δικτύου, οι οποίες θα ανακατασκευαστούν ανεπαρκώς, με αποτέλεσμα ένα υψηλό σφάλμα ανακατασκευής. Αυτή η υπόθεση βασίζεται στο γεγονός ότι οι ανωμαλίες ή οι εισβολές αποτελούν ένα αμελητέο μέρος του συνόλου των δεδομένων. Η τακτοποίηση διαδραματίζει κρίσιμο ρόλο σε αυτή τη ρύθμιση για να αποτρέψει τον αυτόματο κωδικοποιητή από το να μάθει απλώς τη λειτουργία ταυτότητας, κάτι που θα μείωνε την ικανότητά του να ανιχνεύει ανωμαλίες.

Denoising AutoEncoders

Οι Denoising AutoEncoders (DAE) καταστρέφουν σκόπιμα τα δεδομένα εισόδου προσθέτοντας θόρυβο ή τυχαία συγκάλυψη ορισμένων τιμών διανύσματος εισόδου. Ωστόσο, ο στόχος του μοντέλου είναι να ανακατασκευάσει την αρχική, μη αλλοιωμένη είσοδο, όχι την θορυβώδη. Αυτή η διαδικασία διαφθοράς αντιπροσωπεύεται από μια στοχαστική χαρτογράφηση και δεν είναι συγκεκριμένη για κανένα είδος θορύβου. Οι σχετικές εξισώσεις είναι:

$$\tilde{x}^{(i)} \sim M_D(\tilde{x}^{(i)}|x^{(i)})$$

$$L_{DAE}(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_\theta(g_\phi(\tilde{x}^{(i)})))^2$$

όπου το M_D ορίζει την αντιστοίχιση από τα αληθινά δείγματα δεδομένων στα θορυβώδη ή κατεστραμμένα. Ο σχεδιασμός των DAE είναι εμπνευσμένος από την ανθρώπινη ικανότητα να αναγνωρίζει σκοτεινά ή αλλοιωμένα αντικείμενα ή σκηνές. Είναι αποτελεσματικά με υψηλών διαστάσεων, πλεονάζουσες εισόδους όπως εικόνες, καθώς βασίζονται σε πολλαπλές διαστάσεις εισόδου για τον καθαρισμό θορύβου, διασφαλίζοντας έτσι μια ισχυρή λανθάνουσα αναπαράσταση. Οι DAE είναι επίσης χρήσιμα για την ανίχνευση ανωμαλιών, ειδικά σε δεδομένα υψηλών διαστάσεων ή δεδομένα με σύνθετες σχέσεις, όπου οι παραδοσιακές στατιστικές μέθοδοι ενδέχεται να παραπαίουν. Μπορούν να χειριστούν διάφορους τύπους θορύβου και διαφθοράς στα δεδομένα, γεγονός που τα καθιστά ευέλικτα για διαφορετικές εφαρμογές ανίχνευσης ανωμαλιών.

Variational AutoEncoders

Ο Variational AutoEncoder (VAE) [5] είναι μια τεχνική που βασίζεται σε παραλλαγμένες μεθόδους Bayesian και γραφικά μοντέλα [6]. Αντί να μετατρέψει την είσοδο σε στατικό διάνυσμα, η VAE την αντιστοιχίζει σε μια κατανομή p_θ, παραμετροποιημένη από θ. Αυτή η ένωση χαρακτηρίζεται από:

- The prior $p_\theta(z)$
- The likelihood $p_\theta(z|x)$
- The posterior $p_\theta(x|z)$

Η βέλτιστη παράμετρος θ^* είναι αυτή που μεγιστοποιεί την πιθανότητα δημιουργίας γνήσιων δειγμάτων δεδομένων:

$$\theta^* = arg\max_\theta \prod_{i=1}^{n} p_\theta(x^{(i)})$$

Η συνάρτηση απώλειας του VAE αποτελείται από δύο μέρη: την απώλεια ανακατασκευής (RL), που υπολογίζεται χρησιμοποιώντας το Μέσο Τετράγωνο Σφάλμα (MSE) μεταξύ των αρχικών και των ανακατασκευασμένων δεδομένων:

$$RL \equiv MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_i')^2$$

και η KL Divergence (KLD), η οποία μετρά πόσο στενά ευθυγραμμίζεται η κατανομή λανθάνουσας μεταβλητής με μια τυπική κατανομή Gauss. Η συνολική απώλεια VAE είναι:

$$VAELoss = RL + KLD$$

Για να ενεργοποιηθεί η εκπαίδευση με βάση τη διαβάθμιση παρά τη στοχαστική φύση της δειγματοληψίας, χρησιμοποιείται το τέχνασμα επαναπαραμετροποίησης. Αυτό το τέχνασμα εκφράζει την τυχαία μεταβλητή z ως ντετερμινιστική μεταβλητή $z = \mathcal{T}_\phi(x, \epsilon)$, όπου ϵ είναι μια βοηθητική ανεξάρτητη τυχαία μεταβλητή. Για ένα πολυμεταβλητό Gaussian, αυτό εκφράζεται συχνά ως:

$$z = \mu + \sigma \odot \epsilon, where\ \epsilon \sim \mathcal{N}(0, I)$$

όπου το $\odot$ αναφέρεται σε προϊόν βάσει στοιχείων. Το τέχνασμα επαναπαραμετροποίησης λειτουργεί και για άλλους τύπους διανομών, όχι μόνο για Gaussian.

beta-Variational AutoEncoders

Ο β-Variational AutoEncoder (β-VAE) τροποποιεί την αντικειμενική συνάρτηση ενός Variational AutoEncoder για να εξισορροπήσει την απώλεια ανακατασκευής και την απόκλιση KL. Αυτή η προσαρμογή στοχεύει στην ανακάλυψη λανθάνοντων παραγόντων που έχουν ξεμπερδευτεί ή παραγοντοποιηθεί, παρέχοντας ισχυρό πλεονέκτημα ερμηνευσιμότητας και ευκολότερη παρέκταση σε διάφορες εργασίες.

Η αντικειμενική συνάρτηση ενός τυπικού VAE, το κατώτερο όριο αποδεικτικών στοιχείων (ELBO), ορίζεται ως:

$$\text{ΕΛΒΟ} = \mathbb{E}q\phi(z|x)[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

Το β-VAE εισάγει μια υπερπαράμετρο, β, που κλιμακώνει τον όρο απόκλισης ΚΛ:

$$ELBO_\beta = \mathbb{E}q\phi(z|x)[\log p_\theta(x|z)] - \beta \cdot D_{KL}(q_\phi(z|x)||p(z)) \quad (1)$$

Για $\beta = 1$, μειώνεται στο τυπικό VAE. Όταν $\beta > 1$, το μοντέλο δίνει έμφαση στην ευθυγράμμιση με την προηγούμενη διανομή, ενθαρρύνοντας τις ξέμπλεξες αναπαραστάσεις. Ωστόσο, για $\beta < 1$, το μοντέλο εστιάζει περισσότερο στην ανακατασκευή, η οποία μπορεί να είναι ευεργετική για τον εντοπισμό ανωμαλιών λόγω της πιθανότητας ανωμαλιών να έχουν υψηλότερα σφάλματα ανακατασκευής. Έτσι, ένα μοντέλο β-VAE με $\beta < 1$ θα μπορούσε ενδεχομένως να αποφέρει καλύτερη απόδοση σε εργασίες ανίχνευσης ανωμαλιών.

Πειράματα

Αυτό το κεφάλαιο παρέχει μια επισκόπηση της πειραματικής διαδικασίας, των προκλήσεων που αντιμετωπίστηκαν και των γνώσεων που αποκτήθηκαν κατά τη διάρκεια της διπλωματικής εργασίας.

Η αρχική φάση του πειραματισμού περιελάμβανε την εφαρμογή vanilla Autoencoders (AEs) στο σύνολο δεδομένων CICIDS 2017 για ανίχνευση ανωμαλιών. Η χρήση vanilla AE χρησίμευσε ως βάση, καθιερώνοντας θεμελιώδεις μέτρησης απόδοσης και θέτοντας τις βάσεις για τα επακόλουθα πειράματα.

Η επόμενη φάση εισήγαγε τους Deep-Autoencoders (Deep AEs) που επέκτεινε την αρχιτεκτονική του μοντέλου όσον αφορά τα επίπεδα και την πολυπλοκότητα. Ο στόχος αυτού του σταδίου ήταν η βελτίωση της απόδοσης, η βελτίωση της ανίχνευσης ανωμαλιών και η αύξηση της χωρητικότητας του μοντέλου. Κατά τη διάρκεια αυτής της φάσης, η προσαρμογή των υπερπαραμέτρων, όπως ο αριθμός των κρυφών επιπέδων, το μέγεθος των επιπέδων, ο ρυθμός εκμάθησης, batch normalization, ο ρυθμός Dropout και ο αριθμός των εποχών, ήταν το κλειδί για τη βελτιστοποίηση της απόδοσης του μοντέλου.

Μετά τον καθορισμό της απόδοσης των Deep AEs, πραγματοποιήθηκε σύγκριση με ρηχές τεχνικές όπως τα Support Vector Machines (SVM) και το αρχικό βασικό

μοντέλο. Αυτή η σύγκριση επισήμανε τις δυνατότητες των τεχνικών βαθιάς μάθησης και τη σχετική αποτελεσματικότητά τους στο έργο της ανίχνευσης ανωμαλιών.

Περαιτέρω διευρύνοντας το πεδίο της έρευνας, ενσωματώθηκαν παραλλαγές Autoencoder όπως Denoising Autoencoders (DAEs) και β-Variational Autoencoders (β-VAEs). Οι DAE λειτουργούν ανακατασκευάζοντας την αρχική είσοδο από μια θορυβώδη έκδοση ως είσοδο, αναγκάζοντας το μοντέλο να μάθει πιο σημαντικά features από τα δεδομένα. Αντίθετα, οι β-VAE προσθέτουν μια πιθανολογική διάσταση στους παραδοσιακού AE, παρέχοντας μια ερμηνεία της διαδικασίας κωδικοποίησης-αποκωδικοποίησης και ενισχύοντας τις δυνατότητες του μοντέλου.

Η εκπαιδευτική διαδικασία ακολούθησε συστηματική προσέγγιση. Το σύνολο δεδομένων αρχικά χωρίστηκε σε κατηγορίες «κανονική» (normal) και «επιθέσεις» (attacks). Στη συνέχεια, το μοντέλο εκπαιδεύτηκε αποκλειστικά στα «κανονικά» δεδομένα. Μετά την εκπαίδευση, τα δεδομένα «επιθέσεις» ενσωματώθηκαν με το τεστ σετ , το οποίο περιελάμβανε δεδομένα «κανονικής» και «επίθεσης». Η παρουσία περιπτώσεων επίθεσης στο σύνολο δοκιμών επέτρεψε την αξιολόγηση της απόδοσης του μοντέλου στον ακριβή εντοπισμό των δειγμάτων που συνιστούν επίθεση.

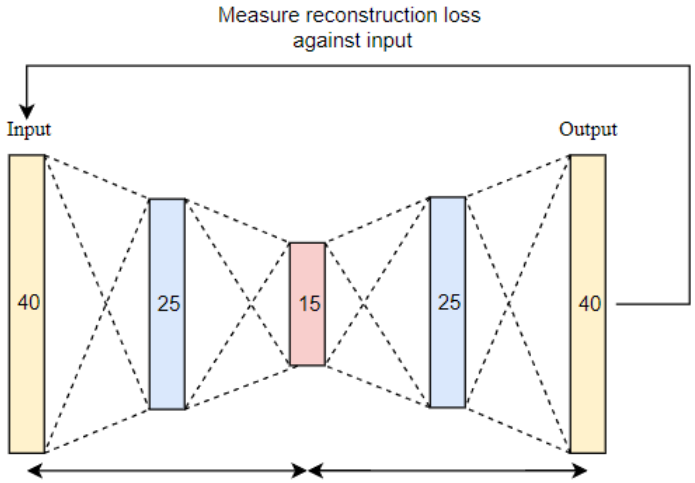
Ένα σημαντικό μέρος της εκπαιδευτικής διαδικασίας ήταν ο υπολογισμός του σφάλματος ανακατασκευής περνώντας τα δεδομένα εκπαίδευσης μέσω του μοντέλου ξανά μετά την ολοκλήρωση της εκπαίδευσης. Αυτό το σφάλμα παρείχε μια εκτίμηση του πόσο αποκλίνουν τα ανακατασκευασμένα δεδομένα από την αρχική εισαγωγή. Με βάση αυτό το σφάλμα ανακατασκευής, καθορίστηκε ένα βέλτιστο όριο(κατώφλι) για την ανίχνευση ανωμαλιών. Αυτό σημαίνει πρακτικά ότι αν ένα δείγμα ξεπερνάει αυτό το κατώφλι στο σφάλμα ανακατασκευής του, τότε χαρακτηρίζεται ως επίθεση και αντίστροφα αν δεν το ξεπερνάει ως κανονική κίνηση δικτύου.

Για την αξιολόγηση της απόδοσης του μοντέλου χρησιμοποιήθηκαν μετρήσεις αξιολόγησης, συμπεριλαμβανομένων της (accuracy), της F1-score, (precision), του recall, του ROC-AUC, του ποσοστού ανίχνευσης επιθέσεων και του ποσοστού ανίχνευσης κανονικής κίνησης δικτύου. Κάθε μέτρηση προσέφερε μια μοναδική προοπτική, βοηθώντας στην κατανόηση της συνολικής αποτελεσματικότητας του μοντέλου και των εγγενών trade-off.

Vanilla AutoEncoders

Αυτή η μελέτη χρησιμοποίησε έναν Vanilla AutoEncoder ως βασικό μοντέλο για τον εντοπισμό ανωμαλιών. Η αρχιτεκτονική του μοντέλου φαίνεται παρακάτω στο Σχήμα 4 ενώ το μέσο τετραγωνικό σφάλμα (MSE) χρησιμοποιήθηκε ως συνάρτηση απώλειας ανακατασκευής.

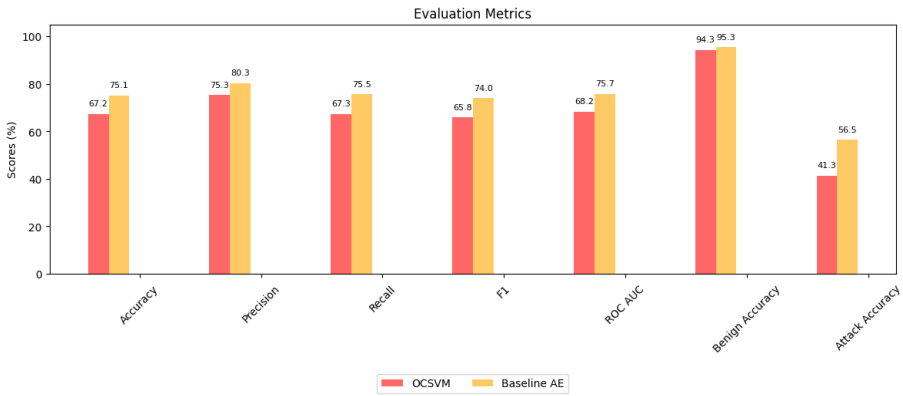
Η απόδοση του Vanilla AutoEncoder αξιολογήθηκε με βάση διάφορες μετρήσεις όπως accuracy, precision, recall, F1-score, ROC AUC, το ποσοστό ανίχνευσης επίθεσης και το ποσοστό ανίχνευσης κανονικής κίνησης δικτύου. Το μοντέλο επέδειξε ακρίβεια 75,1%, σταθμισμένη μέσο precision 80,3%, σταθμισμένη μέσο recall 75,5%,



Σχήμα 4: *Αρχιτεκτονική Vanilla Autoencoder*

σταθμισμένη μέσο F1-score 74,0% και ROC AUC 94,7%. Ενώ το μοντέλο κατάφερε να ανιχνεύσει σωστά το 95,3% των κανονικών περιπτώσεων, είχε σχετικά χαμηλό ποσοστό ανίχνευσης επίθεσης 56,4%.

Σε σύγκριση με το One-Class SVM, ο Vanilla AutoEncoder επέδειξε ανώτερη απόδοση όπως φαίνεται και στο Σχήμα 5, υποδεικνύοντας ότι μπορεί να είναι μια πιο αποτελεσματική λύση για τις ανάγκες αυτής της διπλωματικής εργασίας. Ωστόσο, το σχετικά χαμηλό ποσοστό ανίχνευσης επίθεσης έδειξε ότι ενδέχεται να χρειαστούν πιο εξελιγμένα μοντέλα AutoEncoder για βελτιωμένη ανίχνευση.



Σχήμα 5: *Σύγκριση Vanilla Autoencoder*

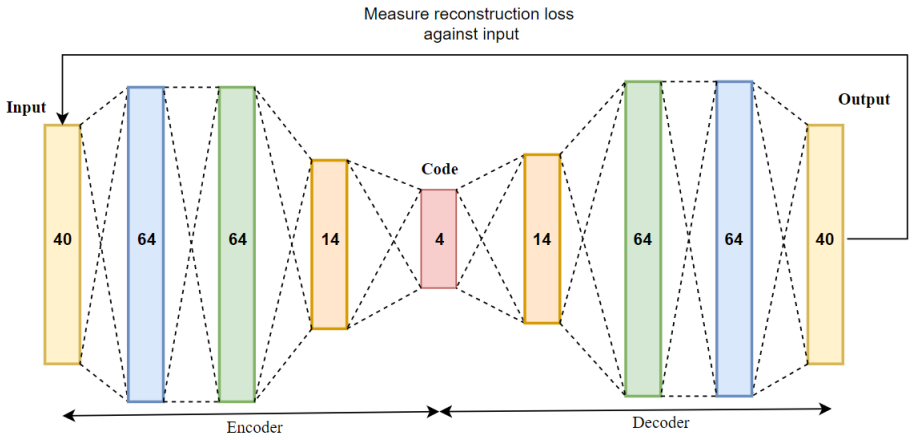
Όσον αφορά τους τύπους επίθεσης, ο Vanilla AutoEncoder πέτυχε την καλύτερη ακρίβεια για DDoS, Portscan, Infiltration, DoS-slowhttptest, DoS-Slowloris & Heartbleed. Από την άλλη πλευρά, δυσκολευόταν να εντοπίσει αρκετές επιθέσεις όπως

Portscan, Botnet, Web Attack-Brute Force, Web Attack-XSS, Web Attack-Sql Injection, SSH-Patator, FTP-Patator & DoS-GoldenEye.

Deep AutoEncoders

Αυτό το πείραμα εστιάζει στην εφαρμογή ενός μοντέλου Deep AutoEncoder, μια επέκταση του παραδοσιακού Vanilla AutoEncoder (AE), για ανίχνευση ανωμαλιών. Οι παράμετροι του μοντέλου, συμπεριλαμβανομένου του αριθμού των επιπέδων, των κόμβων ανά επίπεδο και των διαστάσεων του λανθάνοντος χώρου, ρυθμίστηκαν χρησιμοποιώντας τον αλγόριθμο RandomSearch. Ωστόσο, διαπιστώθηκε ότι ένα χαμηλό validation loss, παρά το γεγονός ότι συχνά υποδεικνύει ένα καλό μοντέλο, δεν εγγυάται ανώτερη απόδοση. Επομένως, προστέθηκε ένα επιπλέον βήμα για την αξιολόγηση της διακριτικής ικανότητας του μοντέλου στο δοκιμαστικό σύνολο, αποκαλύπτοντας ότι το μοντέλο με την καλύτερη απόδοση δεν ήταν αυτό με το χαμηλότερο validation loss.

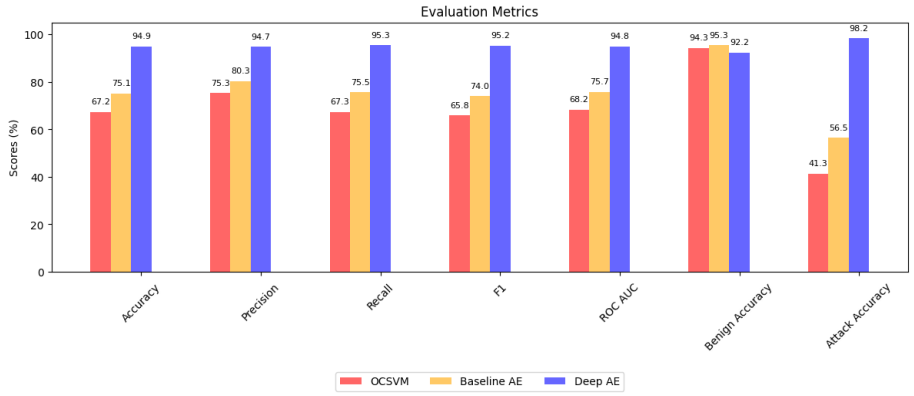
Η αρχιτεκτονική του μοντέλου ύστερα από πειράματα όπως αναφέρθηκαν πριν συνοψίζεται στο Σχήμα 6.



Σχήμα 6: *Αρχιτεκτονική Deep Autoencoder*

Το μοντέλο έδειξε σημαντικό δυναμικό στον εντοπισμό ανωμαλιών. Το μοντέλο πέτυχε συνολική ακρίβεια 94,87%, σταθμισμένο μέσο precision 94,70%, σταθμισμένο μέσο recall 95,31%, F1-score 95,19% και ROC-AUC 94,75%.

Όσον αφορά την ανίχνευση ανωμαλιών, το μοντέλο ξεπέρασε σημαντικά τα βασικά AE και OCSVM όπως φαίνεται στο Σχήμα 7, επιτυγχάνοντας εντυπωσιακό ποσοστό ανίχνευσης επίθεσης 98,17% και ποσοστό ανίχνευσης κανονικής κίνησης δικτύου 92,23%. Οι κατηγορίες επιθέσεων που εντόπισε καλύτερα το μοντέλο περιελάμβαναν DDoS, Portscan, Infiltration & DoS με τις παραλλαγές τους. Από την άλλη πλευρά Botnet, Web Attack-Brute Force, Web Attack-XSS & Web Attack-Sql Injection αποδείχθηκαν πιο δύσκολο να εντοπιστούν.



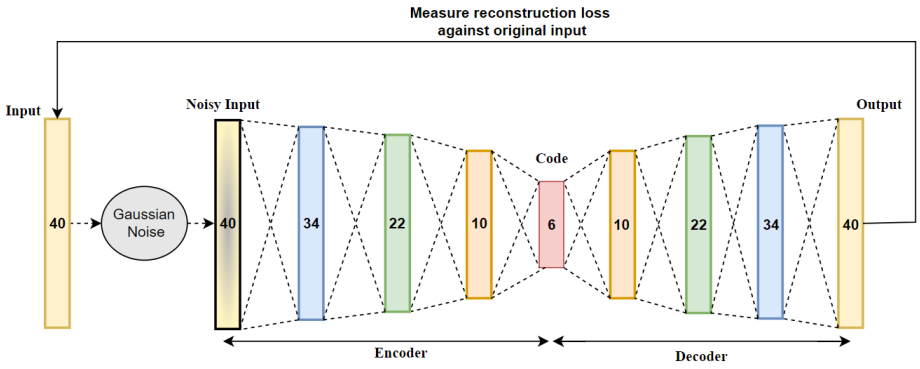
Σχήμα 7: *Σύγκριση Deep Autoencoder*

Συμπερασματικά, το μοντέλο Deep AutoEncoder βελτίωσε σημαντικά την ανίχνευση ανωμαλιών και ελαχιστοποίησε τα ψευδώς θετικά για την καλοήθη κυκλοφορία σε σύγκριση με τα βασικά μοντέλα.

Denoising AutoEncoders

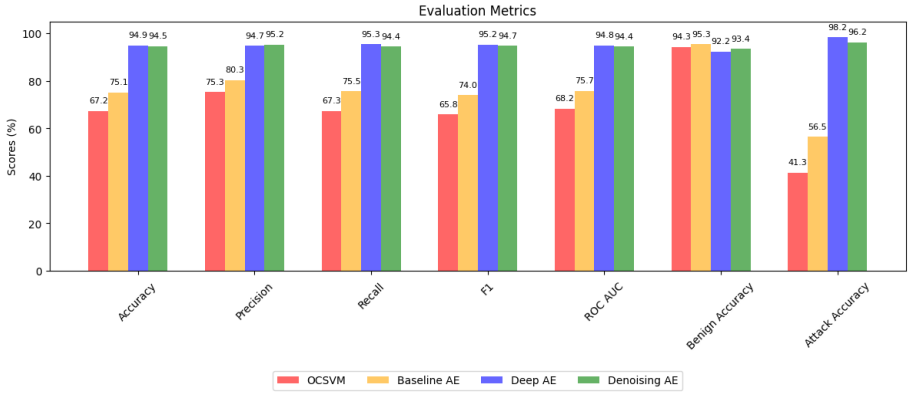
Ο σκοπός αυτού του πειράματος ήταν να εξερευνήσει την επίδραση του θορύβου στην ικανότητα μάθησης των Autoencoders (AEs), ειδικότερα των Denoising Autoencoders (DAEs), για την αντιμετώπιση της υπερεκπαίδευσης. Η υπερεκπαίδευση συχνά προκύπτει λόγω της υπερβολικής πολυπλοκότητας του μοντέλου σε σχέση με τα δεδομένα, το οποίο μπορεί να οδηγήσει τον AE να αναπαράγει την ταυτοτική συνάρτηση αντί να αποτυπώνει βασικά μοτίβα δεδομένων. Η εισαγωγή θορύβου στο μοντέλο μπορεί να βοηθήσει στην αντιμετώπιση αυτού του προβλήματος, αναγκάζοντας το μοντέλο να επικεντρώνεται σε σημαντικά χαρακτηριστικά δεδομένων αντί να απλώς αναπαράγει την είσοδο.

Στο πείραμα, διάφορα επίπεδα θορύβου Gaussian εφαρμόστηκαν σε ένα Deep AE μοντέλο. Ο θόρυβος Gaussian επιλέχθηκε επειδή οι ιδιότητές του συμφωνούν στενά με φυσικές διαδικασίες. Το πείραμα εξέτασε πώς διάφοροι παράγοντες θορύβου (0.1, 0.2, 0.3, 0.4) επηρέασαν το μοντέλο, χρησιμοποιώντας RandomSearch για να προσαρμόσει βέλτιστα τις υπερπαραμέτρους για κάθε επίπεδο θορύβου. Η αρχιτεκτονική του μοντέλου φαίνεται στο παρακάτω Σχήμα 8 ενώ ο παράγοντας θορύβου καθορίστηκε στο 0.2. Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε ήταν ReLU, και το μοντέλο χρησιμοποίησε MSE ως μέτρο απώλειας ανακατασκευής.



Σχήμα 8: *Αρχιτεκτονική Denoising Autoencoder*

Σε σύγκριση, ο Deep AE έδειξε ελαφρώς αποτελεσματικότερος στις περισσότερες μετρικές από το DAE. Ωστόσο, το DAE είχε καλύτερη απόδοση στην ανίχνευση κανονικών στιγμιοτύπων. Η σύγκριση των μοντέλων φαίνεται στο Σχήμα 9. Η εισαγωγή του θορύβου δεν επηρέασε την ικανότητα εκτέλεσης του DAE, δείχνοντας ότι θα μπορούσε να είναι ένα ανθεκτικό μοντέλο για την ανίχνευση ανωμαλιών. Το πείραμα βρήκε το DAE πιο ακριβές στην ανίχνευση DDoS, Portscan, Infiltration, DoS Hulk, DoS slowhttptest, DoS GoldenEye, DoS Slowloris, Heartbleed επιθέσεων, αλλά αντιμετώπισε δυσκολίες στην ανίχνευση Botnet, Web Attack-Brute Force, Web Attack-XSS, και Web Attack-Sql Injection επιθέσεων.

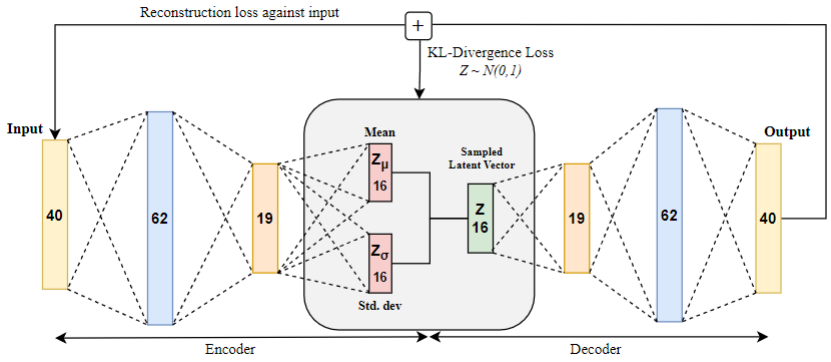


Σχήμα 9: *Σύγκριση Denoising Autoencoder*

beta-Variational AutoEncoders

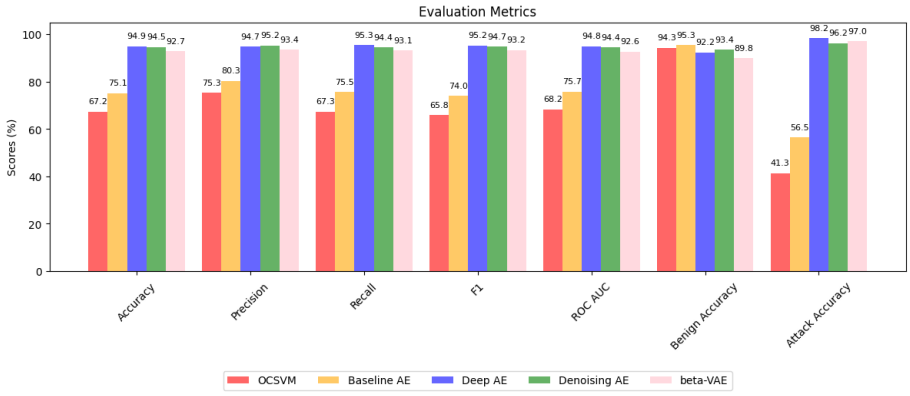
Στο δοθέν πείραμα, παρουσιάζεται την υλοποίηση και τα αποτελέσματα ενός β-Variational AutoEncoder (β-VAE) για την ανίχνευση ανωμαλιών στην κίνηση δικτύου. Το β-VAE διαφέρει από προηγούμενα μοντέλα λόγω των generative δυνατοτήτων του και της μοναδικής του προσέγγισης στη μοντελοποίηση δεδομένων.

Η βέλτιστη διαμόρφωση για αυτό το μοντέλο καθορίστηκε χρησιμοποιώντας έναν αλγόριθμο RandomSearch και συμπεριλάμβανε έναν νέο παράγοντα β, τον συντελεστή KL, στον χώρο αναζήτησης. Οι βέλτιστες υπερπαράμετροι περιλαμβάνουν ένα Dropout rate 0.3, μια τιμή $\beta = 0.1$. Χρησιμοποιήθηκαν συναρτήσεις ενεργοποίησης ReLU και συνάρτηση κόστους τον συνδυασμό MSE, KLD όπως είδαμε και στην θεωρία. Η αρχιτεκτονική του μοντέλου φαίνεται στο παρακάτω Σχήμα 10.



Σχήμα 10: *Αρχιτεκτονική Variational Autoencoder*

Η απόδοση του β-VAE αξιολογήθηκε με βάση την απώλεια ανακατασκευής. Σύμφωνα με τις μετρικές ο β-VAE παρουσίασε εξαιρετική απόδοση, με ακρίβεια 92.70%, precision 93.37% και recall 93.08%. Το ROC AUC 92.55% απέδειξε περαιτέρω την αποτελεσματικότητά του. Το μοντέλο είχε υψηλό ρυθμό ανίχνευσης επιθέσεων 96.95%, αλλά χαμηλότερο ρυθμό ανίχνευσης φυσιολογικών δεδομένων 89.75%, που αποδόθηκε στην μεγαλύτερη εμβέλεια της απώλειας ανακατασκευής. Η σύγκριση με τα υπόλοιπα μοντέλα φαίνεται στο παρακάτω Σχήμα 11.



Σχήμα 11: *Σύγκριση Variational Autoencoder*

Τέλος, ο β-VAE βρέθηκε να αποδίδει την καλύτερη ακρίβεια στην ανίχνευση επιθέσεων DDoS, Portscan, Infiltration, SSH-Patator, FTP-Patator, DoS Hulk, DoS

slowhttptest, DoS Slowloris και Heartbleed, ενώ αντιμετώπισε δυσκολίες με τα Botnet, Web Attack-Brute Force, Web Attack-XSS και Web Attack-Sql Injection.

Ως συμπέρασμα, ο β-VAE παρουσίασε ενθαρρυντική απόδοση στην ανίχνευση ανωμαλιών. Οι μοναδικές ιδιότητες αυτού του μοντέλου και η συγκρίσιμη αποτελεσματικότητα ανοίγουν περαιτέρω ευκαιρίες για εξερεύνηση στον τομέα της ανίχνευσης ανωμαλιών.

Συμπεράσματα

Η παρούσα διατριβή εξερεύνησε την εφαρμογή των AutoEncoders και των παραλλαγών τους (DAE, beta-VAE) στον τομέα της ανίχνευσης ανωμαλιών, επικεντρώνοντας ειδικά στις επιθέσεις Zero-day. Τα αποτελέσματα ήταν ενθαρυντικά, δείχνοντας ότι η μη επιβλεπόμενη μάθηση είναι μια αποτελεσματική προσέγγιση σε αυτόν τον τομέα, συχνά επιτυγχάνοντας υψηλά στάνταρ. Επιπλέον, οι AutoEncoders υπερτερούσαν των παραδοσιακών ρηχών αλγορίθμων μηχανικής μάθησης, όπως το One-Class SVM.

Πιο συγκεκριμένα, ο deep AE αναδείχθηκε ως το καλύτερο μοντέλο, με το Denoising AE να ακολουθεί λίγο πίσω, κατά 0.4%. Αυτό το αποτέλεσμα υποδηλώνει ότι η εισαγωγή θορύβου μπορεί ενδεχομένως να βελτιώσει την απόδοση του μοντέλου όταν συνδυάζεται αποτελεσματικά με αυτήν την προσέγγιση. Αν και ο Denoising AE είναι τεχνικά το δεύτερο καλύτερο μοντέλο, τα αποτελέσματα έδειξαν ότι μια διαφορετική επιλογή κατωφλίου θα μπορούσε να αντισταθμίσει και ακόμα και να υπερβεί το χάσμα απόδοσης του 0.4%. Ωστόσο, αυτή η προσαρμογή δεν ισχύει και για το καλύτερο μοντέλο, καθώς το κατώφλι έτυχα να ήταν βέλτιστα καθορισμένο για αυτό το μοντέλο.

Ένας κρίσιμος παράγοντας σε αυτά τα πειράματα ήταν η επιλογή ενός σταθερού κατωφλίου, βασισμένου στο σφάλμα ανακατασκευής στα δεδομένα εκπαίδευσης. Παρά το γεγονός ότι αυτή είναι μια ευρέως υιοθετημένη μέθοδος (και ο λόγος για τον οποίο επιλέχθηκε για αυτήν την μελέτη), φαίνεται ότι είναι υπο-βέλτιστη. Ενδιαφέρουσα προοπτική αποτελεί η εξερεύνηση πιο δυναμικά προσαρμοζόμενων μεθόδων.

Ένα άλλο σημαντικό στοιχείο αυτής της διατριβής είναι η εφαρμογή των generative μοντέλων για την ανίχνευση ανωμαλιών. Αν και η εφαρμογή generative μοντέλων στην ανίχνευση ανωμαλιών είναι νέο πεδίο, τα αποτελέσματα έδειξαν ότι οι προηγμένες δυνατότητες στην μοντελοποίηση των δεδομένων του VAE θα μπορούσαν να παράσχουν πολύτιμους δείκτες για την ανίχνευση ανωμαλιών. Διαπιστώθηκε ότι ο VAE ήταν λιγότερο ικανός στην ανίχνευση ανωμαλιών. Ωστόσο, αυτό θα μπορούσε να αποδοθεί και στο γεγονός ότι ως generative μοντέλο, η επικέντρωσή του δεν είναι στην τέλεια ανακατασκευή, η οποία χρησιμοποιήθηκε σαν το κύριο κριτήριο κατά την ταξινόμηση των δειγμάτων. Αξίζει να ληφθεί υπόψη ότι ο VAE ίσως να επωφεληθεί από άλλες μεθόδους ταξινόμησης που μπορεί να αξιοποιούν τις ιδιότητές τους πιο αποτελεσματικά.

Chapter 1

Introduction

1.1 Zero Day Attacks

Zero-day attacks are a type of cyber attack that exploit software and/or hardware vulnerabilities that are unknown to developers. These attacks are particularly dangerous because traditional security measures are often ineffective against them, leaving organizations vulnerable to data breaches, financial losses, and reputational damage.

According to a recent report by Symantec, the number of zero-day vulnerabilities discovered in 2020 increased by 55% compared to the previous year. This increase in zero-day vulnerabilities suggests that attackers are becoming more sophisticated and discover new ways to exploit unknown vulnerabilities in software and hardware.

The impact of zero-day attacks may be severe, particularly for businesses and organizations that rely on technology to conduct their operations. Data breaches resulting from zero-day attacks may lead to the theft of sensitive information, including financial and personal data, and may result in significant financial losses for businesses. Furthermore, the reputational damage caused by data breaches can be long-lasting and may result in loss of customers and revenue.

In order to mitigate the impact of zero-day attacks, it is important for organizations to implement effective remediation measures. This includes regularly updating software and hardware to address known vulnerabilities, and implementing advanced security measures such as intrusion detection and prevention systems. Additionally, the use of deep learning techniques such as AutoEncoders can provide an effective defense against zero-day attacks by identifying potential vulnerabilities before they can be exploited.

Overall, the increasing number of zero-day vulnerabilities discovered each year underscores the need for organizations to remain vigilant and implement robust security measures to protect against cyber attacks.

Objectives

The central aim of this diploma thesis is to devise a Network Threat Detection System powered by AutoEncoders, an innovative strategy to effectively mitigate zero-day attacks. Notably, these attacks exploit unidentified vulnerabilities, making them considerably dangerous as traditional security measures often fall short.

A standout feature of the project is the application of unsupervised learning. This approach removes the need for labor-intensive data labeling, offering the added advantage of generalizing to unseen attacks. In other words, the system can identify and respond to threats not present in the initial training data.

The detailed objectives of the thesis extend beyond the standard IDS construction. They involve exploring various types of AutoEncoders and drawing comparisons between them, providing a comprehensive understanding of their application in cybersecurity.

The thesis emphasizes the incorporation of both discriminative and generative models, utilizing Variational AutoEncoders (VAEs). These approaches are considered to enhance the system's predictive accuracy, making it a more robust defense against zero-day attacks.

Performance evaluation forms a crucial aspect of this thesis. The proposed system will be tested against the CICIDS 2017 dataset, an established benchmark for intrusion detection systems. The results will be compared with those of other systems and traditional security measures to assess its effectiveness.

In terms of optimizing the system, factors such as the number of hidden layers in the auto-encoder and the training dataset size will be examined. This exploration will help understand the impact of these parameters on the system's performance.

Ultimately, this diploma thesis seeks to make a significant contribution to cybersecurity. By employing AutoEncoders and leveraging both unsupervised learning and the usage of discriminative and generative models, the project aims to enhance defenses against zero-day attacks and broaden the application of deep learning techniques in cybersecurity.

1.2 Thesis Outline

The current diploma thesis is organized as follows. In Chapter 2, a theoretical background is given regarding the Artificial Intelligence field and the Neural Networks used in the scope of the current diploma thesis. All the models and classifiers that were used are analyzed and explained thoroughly. Chapter 3 discusses the application of both supervised and unsupervised Machine Learning (ML) and Deep Learning (DL) techniques for the detection of zero-day (unseen) cyber attacks, a task often referred to as anomaly detection. The dataset used to conduct all the

experiments is described in Chapter 4 along with the preprocessing procedure followed to transform the data to the appropriate form. Chapter 5 demonstrates the experimental procedure such as training, evaluation, visualization tools, and then the experiments conducted and their results. Finally, Chapter 6 describes the work that should be done along with ways and ideas that can improve our current work on this subject.

Chapter 2

Theoretical background

2.1 Analysis of network traffic

The analysis of network traffic is a critical task in network security. Network traffic can be analyzed at different levels, such as packets and flows, each providing different insights into the network traffic characteristics.

Packet-based analysis involves capturing and examining individual packets that traverse the network. This approach provides a fine-grained view of the network activity, allowing for detailed inspection of the network protocol stack and the identification of potential anomalies or security threats.

On the other hand, flow-based analysis involves aggregating packet-level data into higher-level network flows, which represent the communication between two endpoints over a certain period of time. This approach provides a more abstract and compact view of the network activity, allowing for easier interpretation and visualization of the data. In this section, we will discuss the analysis of network traffic using both packet-level and flow-level approaches

2.1.1 Analysis of network traffic based on packets

Packet analysis involves the examination of individual data units that make up network communications between devices. These units, called packets, consist of a header that includes information about the source and destination addresses, protocol type, and other details, as well as a payload containing the actual data being transmitted.

Packet analysis is a crucial tool for network administrators to gain insights into how data go through their networks, identify potential security threats, troubleshoot network issues, and optimize network performance. By analyzing packets, network administrators can detect patterns of traffic and unusual behavior, such as unusual traffic volume or unusual types of traffic.

Various tools, including network traffic analyzers, packet sniffers, and intrusion detection systems, can perform packet analysis in real-time or through the analysis

of stored packet data. However, packet analysis is a complex process that requires expertise in networking, security, and data analysis to interpret the results and identify actionable insights. The importance of packet analysis in network management and security cannot be overstated, and it remains a critical component of modern network monitoring and analysis.

Intrusion detection systems rely on the capture and analysis of network traffic data in order to identify and respond to security threats. The initial phase of this process involves capturing and transforming packets of data into a usable format for further analysis. Captured data can be stored in raw, pcap, or pcap-ng formats, and then transformed into other formats such as Netflow, sFlow, CSV, JSON, or user-defined structures.

During this transformation process, various header information is extracted from the packets at different layers of the OSI model, depending on the specific analysis being conducted [7]. Netflow and sFlow, for example, focus on layer 3 and layer 4 protocols, while other formats may extract information from all other layers as well. The specific data that is extracted and stored will depend on the type of analysis being conducted.

In order for intrusion detection systems to effectively identify and respond to security threats, it is critical that the captured data be transformed into a usable format that allows for detailed analysis of network traffic patterns and potential indicators of compromise. This process of capturing and transforming data from packets is essential for enabling accurate and effective intrusion detection and response.

2.1.2 Analysis of network traffic based on flows

Network traffic based on flows refers to the flow of data packets between network devices. A flow is a series of packets that share a set of common attributes such as source and destination addresses, protocol type, and port numbers. To give an ID to a flow, the most common approach is to use a combination of the following characteristics: Source IP address, Destination IP address, Source port, Destination port, and Protocol(TCP, UDP, ICMP). Using a combination of these characteristics can create a unique identifier for each flow, allowing network administrators to track individual flows and analyze their behavior over time.

Network traffic based on flows is a way of analyzing network activity by grouping packets into flows and analyzing the characteristics of those flows. This approach provides a more detailed view of network activity than simply looking at the raw packets themselves. Flow-based analysis can be used for a variety of purposes, including troubleshooting network issues, identifying security threats, and optimizing network performance. By analyzing flows, network administrators can gain insights into how data are moving through their networks and identify potential problems or areas for improvement.

Flow-based analysis is generally considered more scalable than packet-based analysis. This is because it requires fewer resources to analyze flows than packets, as

flows are often more aggregated and condensed than packets. Flow-based analysis can also be more efficient, as it can identify anomalous behavior across multiple packets within a flow. This makes it well-suited for identifying complex attacks that span multiple packets.

However, flow-based analysis has some limitations. It may not be able to capture all types of attacks, especially those that involve individual packets rather than flows. Additionally, flow-based analysis relies heavily on the quality of flow data, and if this data are incomplete or inaccurate, it can result in false positives or false negatives.

In conclusion, the analysis of network traffic based on flows appears to be the most suitable approach for intrusion detection systems due to several reasons. Firstly, flow-based approaches offer superior scalability compared to packet-based methods. As network traffic speeds continue to increase rapidly, packet-based methods could demand high computational resources, leading to increased energy consumption. Furthermore, the response time required for packet-based methods may not keep up with the incoming network traffic speeds, leading to potential delays and missed detections.

Secondly, some attacks might not rely on a single packet but instead occur as a sequence of packets, where flow-based approaches can better capture the attack pattern. Flow-based methods can provide a more holistic view of network traffic, taking into account not only individual packets but also the relationship between packets over time.

2.2 Intrusion Detection Systems

2.2.1 Definition

An Intrusion Detection System (IDS) is a critical component of network security that constantly monitors network traffic and system activities for any indications of potential security threats. It is designed to detect, record, and respond to security incidents that may arise in a system.

The primary function of an IDS is to identify any unauthorized or malicious activities taking place on a network or system. This is done through the analysis of traffic or system activity for any abnormal or suspicious behavior. IDS can detect various types of attacks, such as Denial of Service (DoS) attacks, malware infections, and unauthorized access attempts.

2.2.2 Data for Intrusion Detection Systems

The development of intrusion detection systems (IDS) based on network traffic data can be categorized into three main types. The first type is based on packet fields of network traffic. These data sets contain detailed information on network traffic, which can be exported in pcap format files and contain the data of each packet.

The available metadata and packet headers depend on the recording network and protocol used, such as TCP, UDP, and ICMP. Often, beyond the header of the transport protocol, there is also an IP header, with information such as source and destination addresses.

The second type of data are stream-based, containing condensed meta-information collectively interested in the exchange of packets within the network, often without including the data itself. The usual definition of flow in these data sets is the one used in the present IDS. These flows can be unidirectional or bidirectional, taking into account the communication between two systems or users within the network. Standard protocols for exporting flows are NetFlow, IPFIX, sFlow, and OpenFlow. There are many available tools, such as nfdump and YAF, which can convert data based on packets to corresponding data based on flows. One of these tools is the CICFlowMeter-V3, which was used to produce features based on bidirectional flows.

The third type of data is not exclusively categorized by the above distinctions. An example of this type of dataset may contain stream-based data but also enriched with data from the packets themselves or from records of systems on the network. The KDD Cup 1999 [1] data set is representative of this type, which, in addition to aggregate features, also contains the number of failed login attempts. The general structure of this data varies and depends on the analysis being prepared.

It is worth noting that the use of data based on flows has become more popular due to their condensed format and scalability. The use of these data sets allows for more efficient analysis and detection of network anomalies. However, it is important to consider the limitations of each type of data set and their relevance to the specific analysis being conducted.

2.2.3 Different Types of Intrusion Detection Systems

IDSes come in different flavors and detect suspicious activities using different methods, including the following:

- A **network intrusion detection system** (NIDS) is deployed at a strategic point or points within the network, where it can monitor inbound and outbound traffic to and from all the devices on the network.

- A **host intrusion detection system** (HIDS) runs on all computers or devices in the network with direct access to both the internet and the enterprise's internal network. A HIDS has an advantage over an NIDS in that it may be able to detect anomalous network packets that originate from inside the organization or malicious traffic that an NIDS has failed to detect. A HIDS may also be able to identify malicious traffic that originates from the host itself, such as when the host has been infected with malware and is attempting to spread to other systems.

- A **signature-based intrusion detection system** (SIDS) monitors all the packets traversing the network and compares them against a database of attack signatures or attributes of known malicious threats, much like antivirus software [2].

- An **anomaly-based intrusion detection system** (AIDS) monitors network traffic and compares it against an established baseline to determine what is considered normal for the network with respect to bandwidth, protocols, ports and other devices. This baseline is typically created using statistical analysis of network traffic over a period of time, such as a week or a month. The system then continuously monitors network traffic and compares it against this baseline to detect any deviations from normal behavior. Anomaly-based IDS does not rely on known attack signatures or patterns, so it can detect new and unknown types of attacks. However, it can also produce false positives if legitimate traffic deviates significantly from the established baseline. To minimize false positives, anomaly-based IDS often requires fine-tuning and ongoing updates to the baseline as the network evolves.

Machine Learning (ML) came to intrusion detection systems (IDS) as a way to address the limitations of traditional signature-based IDSes. Traditional IDSes were only capable of detecting known threats by comparing network traffic or system activity against a database of known malicious signatures. This approach had limitations in detecting new or unknown threats.

IDSes utilizing ML (ML-IDS), on the other hand, use algorithms that are trained on data to detect patterns in network traffic or system activity. By using statistical methods, ML-IDS can identify anomalies in network traffic or system activity that may be indicative of a security breach.

The use of ML-IDS has grown in popularity due to the increasing volume and complexity of network traffic, as well as the growing number of sophisticated attacks that can evade traditional signature-based IDSes. Contrary to conventional IDS, ML-IDS can learn and adapt to new threats, improving their ability to detect and respond to security threats.

ML-based IDSes are capable of detecting a wide range of attacks, including malware, phishing, and denial-of-service (DoS) attacks. They can also identify patterns of behavior that may indicate an insider threat or other unauthorized access.

Deep Learning (DL) has been applied to IDS as well and it's gaining more attention. The reason behind this is that deep learning techniques and neural networks can identify more complex patterns that may be missed by traditional ML-IDS,

2.3 Machine Learning & Neural Nets

Machine Learning (ML), a critical subdomain of artificial intelligence, provides systems with the ability to autonomously learn and improve from experience without being explicitly programmed. This dynamic discipline straddles the spheres of computer science and statistics, employing statistical models and algorithms to perform tasks by making data-informed predictions or decisions.

Machine Learning (ML) holds substantial value across diverse sectors due to its capacity to identify patterns and trends that would otherwise remain elusive to human analysis. It is this capability that empowers businesses to gain critical insights into customer behavior and operational dynamics, thereby driving strategic decision-making and fostering innovation.

Three primary categories of machine learning exist: supervised learning, unsupervised learning, and reinforcement learning. Briefly, supervised learning infers a function from labeled input data to predict outcomes for unseen data. Unsupervised learning, in contrast, seeks to identify inherent patterns and structures in unlabeled data. Reinforcement learning involves an agent learning to make decisions by interacting with an environment, guided by reward and penalty signals.

One core concept in machine learning methodologies is the loss or cost function, which quantifies the discrepancy between the model's predictions and the actual values. The central objective in machine learning is to minimize this loss.

However, achieving an optimal model is not without challenges. Two fundamental issues are overfitting, where a model learns the noise rather than the pattern, and underfitting, where the model is overly simplistic to capture the data complexity. Strategies such as regularization, early stopping, and cross-validation help mitigate these issues, ensuring the model's robustness and reliability.

Machine learning, characterized by its wide-ranging applicability, has transformative impacts across diverse fields, from natural language processing and computer vision to predictive analytics and personalized recommendation systems. With the continued surge in data availability and computational power, the role of machine learning in shaping our data-driven future is set to become increasingly prominent.

2.3.1 Supervised Learning

Supervised machine learning is an integral part of data analysis, where the algorithm is trained on a pre-existing "training dataset". This dataset provides specific context and instructions for correct and incorrect outputs. The concept extends to a data classification problem, where the number of individual categories for classification is known in advance. Furthermore, the training phase requires samples already classified into these known categories. Employing iterative methods, the algorithm gradually refines its performance, resulting in an ever-decreasing error rate during

data classification. A human supervisor often guides this process, making the necessary corrections and adjustments to the model, akin to safety rails in a bowling game. As the learning progresses, the dependence on the supervisor diminishes, demonstrating the proficiency acquired by the machine learning model.

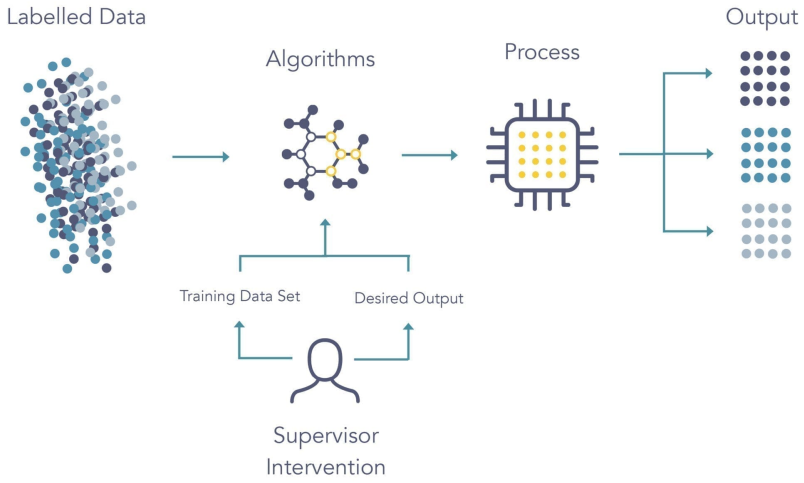


Figure 2.1: *Supervised Learning*

2.3.2 Unsupervised Learning

Unsupervised machine learning is a powerful approach for exploring undistinguished data, where the algorithm is not guided by any pre-existing labels or categories. Unlike supervised learning, the training data in unsupervised learning are unlabelled and the model, therefore, is tasked with discovering inherent structure within this data, with no human intervention. This technique is particularly valuable when there is no prior knowledge about potential categories or groups within the data, and we are instead relying on the algorithm to identify unique clusters based on their intrinsic characteristics. As such, there aren't predefined outcomes such as 'true' or 'false' as in supervised learning, and instead, the unsupervised model offers insight into potential areas of interest, highlighting differences, anomalies, or outliers. This approach allows the algorithm to navigate its journey from unknown data territory towards recognizing and learning complex patterns, thus facilitating the discovery of new insights from the data.

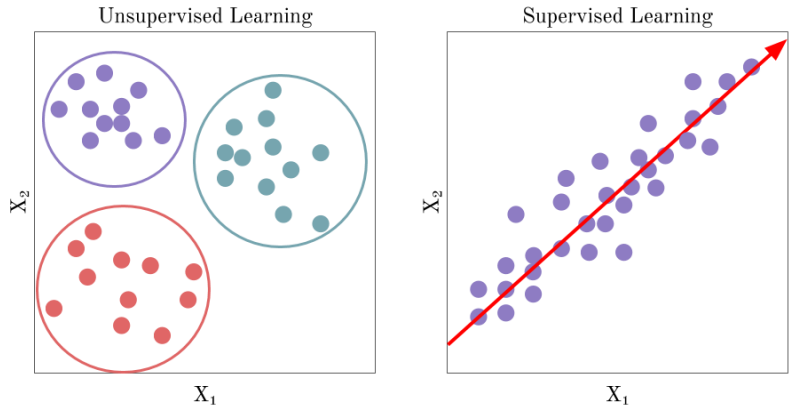


Figure 2.2: *Difference between Supervised and Unsupervised Learning*

2.3.3 Reinforcement Learning

Reinforcement learning is a sub-field of machine learning that deals with the problem of training an agent to maximize a reward signal while acting in an environment. This method is based on rewarding desired behaviors and/or punishing undesired ones, hence a reinforcement learning agent is able to learn through trial and error. The main goal of reinforcement learning is to define the best sequence of decisions the agent has to follow to solve a problem while maximizing a long-term reward. This is why it is primarily applied for motion planning, dynamic pathing, controller optimization, scenario-based learning policies for highways etc. A characteristic example of the adequacy of the method is its use for parking that can be achieved by learning automatic parking policies.

2.3.4 Classification & Regression

Classification and Regression represent the two primary types of supervised learning. These computational techniques, fundamental to Machine Learning (ML), allow us to collect actionable insights and predictions from structured and unstructured data. Though they share similarities, their applications and outcomes are fundamentally different.

Classification, in its core, is a categorization process. It is leveraged to separate input data into specific categories or classes. This technique takes a given dataset, learns from it using an array of ML algorithms such as decision trees, logistic regression, or neural networks, and subsequently makes predictions about new data's classification. For instance, email filtering (spam or not spam), customer churn prediction (will leave or stay), and medical diagnoses (disease or no disease) are typical applications of classification.

Classification can be binary, where the algorithm decides between two classes (e.g., True or False), or multi-class, where more than two classes are present (e.g., classifying types of fruits based on features).

Conversely, **Regression** is a prediction methodology that estimates the relationship between dependent (target) and independent variables (features). This method does not predict a class or category but a continuous value, such as a real number. For instance, predicting housing prices, forecasting the stock market, and estimating the age of a fossil, all leverage regression techniques.

Regression can take various forms, including linear (simple and multiple), logistic, polynomial, ridge, and lasso regression. Each form serves different purposes and is appropriate under distinct circumstances. Understanding which type of regression to use is paramount in ensuring the validity and accuracy of your predictions.

In essence, Classification and Regression underpin the world of predictive analytics and supervised machine learning. They provide a mechanism to understand and predict complex phenomena, empowering researchers to make data-driven decisions and forecasts. It is crucial to acknowledge, however, that while these methods can provide profound insights, they also require thorough data preprocessing, validation, and interpretation to be truly effective.

2.3.5 Overfitting & Underfitting

Overfitting and **Underfitting** are phenomena that occur during the training of machine learning (ML) models. They reflect a model's performance, and understanding them is crucial for achieving a model that generalizes well to unseen data.

Overfitting occurs when a model is excessively complex, typically characterized by having too many parameters relative to the number of observations. An overfitted model often performs exceptionally well on training data because it 'learns' the noise and outliers, capturing minute fluctuations that do not represent the underlying data pattern. However, when introduced to new, unseen data (test data), its performance tends to decline. This differentiation in performance arises because an overfitted model fails to generalize the underlying data pattern but instead memorizes the training data.

Preventing overfitting involves various techniques such as reducing the model's complexity, using regularization (like L1 and L2), increasing the training data, and employing cross-validation methods.

On the other side of the spectrum lies **Underfitting**. Underfitting arises when a model is too simple to capture the underlying data structure and pattern. An underfitted model cannot grasp the complexity of the data, thereby resulting in poor performance on both the training and testing data. Underfitting often indicates that the model can be improved by increasing its complexity or using more suitable features.

Overcoming underfitting might involve adding more features, increasing the model complexity, or utilizing more complex ML algorithms. However, care must be taken to avoid tipping the balance towards overfitting.

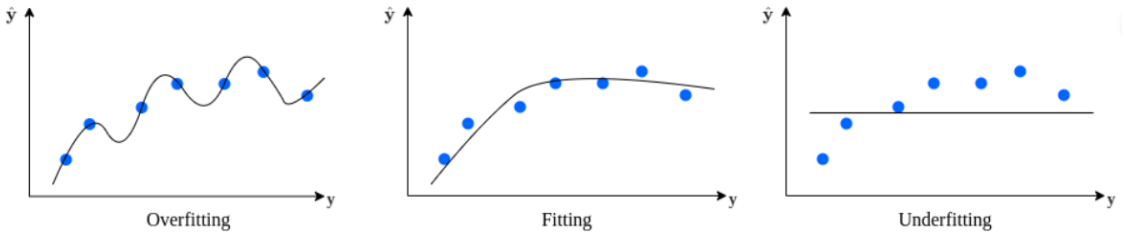


Figure 2.3: *bias-variance tradeoff*

Overfitting and Underfitting are critical considerations that demand a thoughtful balance between the complexity of the model and its simplicity. The objective here is to develop a model that neither resorts to memorization of training data (a symptom of overfitting) nor overlooks the inherent complexities present in the data (a sign of underfitting). This ideal balance, often referred to as the bias-variance tradeoff, serves as a crucial aspect of the pursuit of excellence in machine learning model performance.

While these phenomena offer deep understanding into the operations of machine learning models, it is important to note that handling them effectively necessitates careful steps, such as appropriate model selection, validation techniques, and thorough feature engineering, ensuring a model that is both robust and reliable.

2.3.6 Regularization

Regularization techniques are applied to address overfitting in machine learning models. This situation is depicted by the generalization curve, where despite a continuous decrease in training loss, the validation loss increases after a certain point due to the model's increasing complexity. Essentially, a high variance model captures all details, including noise, increasing the gap between training and validation loss. In contrast, a high bias model has a smaller gap due to its loose fit to the data. Therefore, preventing overfitting is key to improving model performance, which is where regularization techniques come into play.

Regularization means restricting a model to avoid overfitting by shrinking the coefficient estimates to zero. When a model suffers from overfitting, we should control the model's complexity. Technically, regularization avoids overfitting by adding a penalty to the model's loss function:

$$Regularization = LossFunction + Penalty \tag{2}$$

There are three commonly used regularization techniques to control the complexity of machine learning models, as follows:

- **L2 regularization**

A linear regression that uses the L2 regularization technique is called ridge regression. In other words, in ridge regression, a regularization term is added to the cost function of the linear regression, which keeps the magnitude of the model's weights (coefficients) as small as possible. The L2 regularization technique tries to keep the model's weights close to zero, but not zero, which means each feature should have a low impact on the output while the model's accuracy should be as high as possible.

$$RidgeRegressionCostFunction = LossFunction + \frac{1}{2}\lambda \sum_{j=1}^{m} w_j^2 \tag{3}$$

Where λ controls the strength of regularization, and w_j are the model's weights (coefficients). By increasing λ, the model becomes flattered and underfit. On the other hand, by decreasing λ, the model becomes more overfit, and with $\lambda = 0$, the regularization term will be eliminated.

- **L1 regularization**

Least Absolute Shrinkage and Selection Operator (lasso) regression is an alternative to ridge for regularizing linear regression. Lasso regression also adds a penalty term to the cost function, but slightly different, called L1 regularization. L1 regularization makes some coefficients zero, meaning the model will ignore those features. Ignoring the least important features helps emphasize the model's essential features.

$$LassoRegressionCostFunction = LossFunction + \lambda \sum_{j=1}^{m} |w_j| \tag{4}$$

Where λ controls the strength of regularization, and w_j are the model's weights (coefficients). Lasso regression automatically performs feature selection by eliminating the least important features.

- **Elastic Net**

The Elastic Net is a regularized regression technique combining ridge and lasso's regularization terms. The r parameter controls the combination ratio. When $r = 1$, the L2 term will be eliminated, and when $r = 0$, the L1 term will be removed.

$$\text{Elastic Net Cost Function} = \text{Loss Function} + r\lambda \sum_{j=1}^{m} |wj| + \frac{(1-r)}{2}\lambda \sum_{j=1}^{m} w_j^2 \tag{5}$$

Although combining the penalties of lasso and ridge usually works better than only using one of the regularization techniques, adjusting two parameters, λ and r, is a little tricky.

2.3.7 Deep Learning

Deep learning is an extension of classical machine learning that introduces added "depth", which signifies enhanced complexity within the model. It involves the transformation of data via multiple functions, enabling the representation of data hierarchically, across varying levels of abstraction. A distinctive advantage of deep learning is its proficiency in solving intricate problems quickly and accurately. This is largely attributable to its capacity to allow massive parallelization [3].

Deep learning consists of a wide array of components such as convolutions, pooling layers, fully connected layers, gates, memory cells, activation functions, encode/decode schemes, among others, depending on the network architecture being employed. It's crucial to recognize that deep learning exhibits remarkable flexibility and adaptability, making it suitable for a broad spectrum of complex challenges from the perspective of data analysis. This is primarily due to the high hierarchical structure and enormous learning capacity of deep learning models [3].

2.3.8 Artificial Neural Networks structure

Artificial Neural Networks (ANNs) are computational models inspired by the biological neural networks found in animal brains. At the core of these networks are units or nodes known as artificial neurons, mirroring the neurons present in a biological brain. A neuron is an electrically excitable cell that exchanges information with other cells through unique connections, termed synapses.

As depicted in Figure 2.4, a typical neuron comprises a cell body, known as the soma, alongside dendrites and a singular axon. The soma is a compact structure, from which the axon and dendrites extrude. Dendrites branch out substantially, typically a few hundred micrometers from the soma.

Conversely, the axon originates from the soma at the axon hillock, appearing as a swelling, and can extend up to one meter in humans. At the terminal end of the axon are branches referred to as the axon terminals. Here, the neuron can transmit a signal across the synapse to another cell, facilitating intercellular communication.

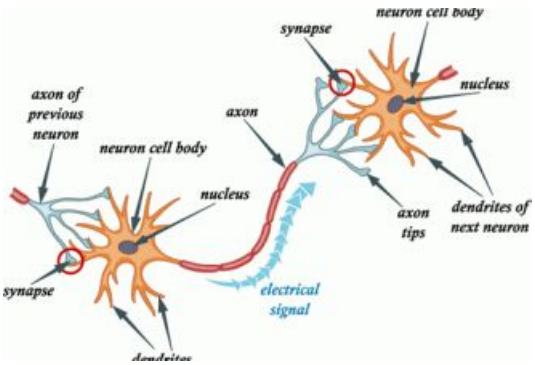


Figure 2.4: *Biological neurons synapse*

Mirroring the structure of a biological neuron, a typical artificial neuron is illustrated in Figure 2.5. An artificial neuron, fundamentally, is a mathematical function designed as a model of a biological neuron, creating a neural network. An artificial neuron accepts one or multiple inputs, simulating the postsynaptic component of a biological neuron— the dendrites. These inputs are summed to generate an output, or activation, which symbolizes a neuron's action potential that is transmitted along its axon.

Typically, each individual input (or edge) is weighted separately (w_1,w_2,...,w_m) to imitate the excitability of the biological neuron. The output is calculated as the sum of the product of each input and its corresponding edge weight. This function used to compute the output is known as the transfer function.

Subsequently, the final value computed by the transfer function is compared with a threshold value through another function referred to as the activation function. If this calculated value exceeds the threshold, the artificial neuron is "fired" or activated, and it passes on its information to the succeeding neuron.

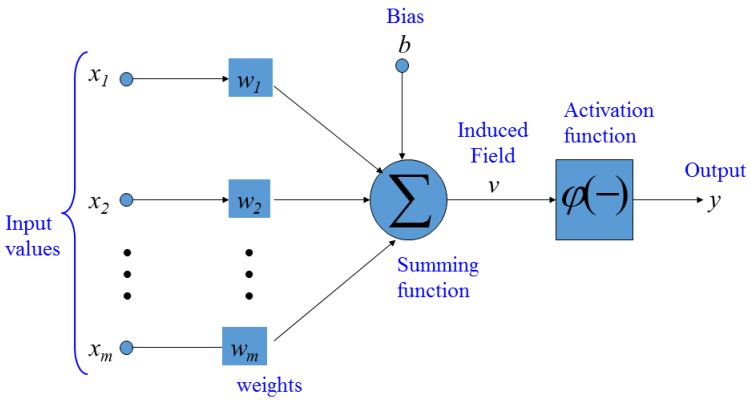


Figure 2.5: *Artificial neuron*

The architecture of an Artificial Neural Network (ANN) is depicted in Figure 2.6. The figure reveals that the neurons are arranged in multiple layers, each potentially containing a diverse number of neurons. Neurons within the same layer remain unconnected to each other; instead, they form connections exclusively with neurons in the layers immediately preceding and immediately following their own.

The network receives inputs through the input layer, while the output layer generates the final output. Intermediary to these, there may be zero or more hidden layers. If the network comprises multiple hidden layers, it is typically referred to as a Deep Neural Network (DNN).

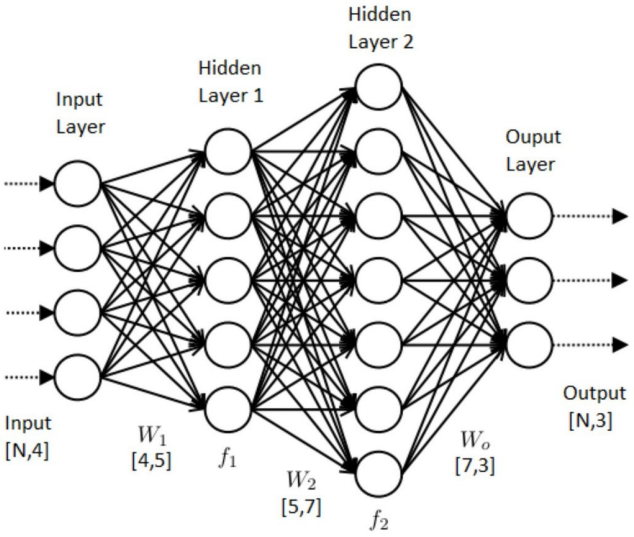


Figure 2.6: *Artificial Neural Network*

2.3.9 Activation Functions

Activation functions play a pivotal role in the efficiency of a neural network. The choice of activation function significantly impacts the model's predictions, as it governs the network's capacity to learn from the dataset. An activation function dictates the neuron's output transmitted to the subsequent layer. An activation function could be as simple as a binary function, which turns the neuron on and off depending on the input. It can also transform the input signal to an output signal within a range of -1 to 1. Activation functions are fundamentally categorized into two types: linear and non-linear. Non-linear functions are most commonly employed because they enable the model to generalize or adapt to a variety of data and to distinguish between the output.

Sigmoid Function

Also known as the logistic function, the Sigmoid function is a mathematical function characterized by an "S" curve. It accepts any real number as an input and yields output values that approach but never reach 0 and 1, making it suitable for models predicting probabilities. The sigmoid function is depicted in figure 2.7 and is defined for all real input values by the formula:

$$S = \frac{1}{1 + e^{-x}} \tag{6}$$

Tanh/Hyperbolic Tangent Function

The tanh function, quite similar to the sigmoid function, also exhibits an "S" curve but generates output values in the range -1 to 1. The tanh function is defined by the formula:

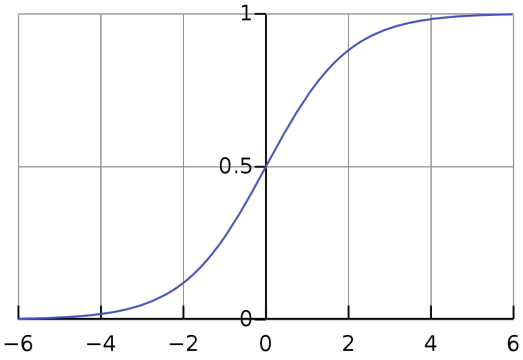


Figure 2.7: *Sigmoid Activation Function*

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{7}$$

The advantage of the tanh function is that negative inputs are mapped as strongly negative, and zero inputs are mapped near zero. The tanh function is depicted in figure 2.8.

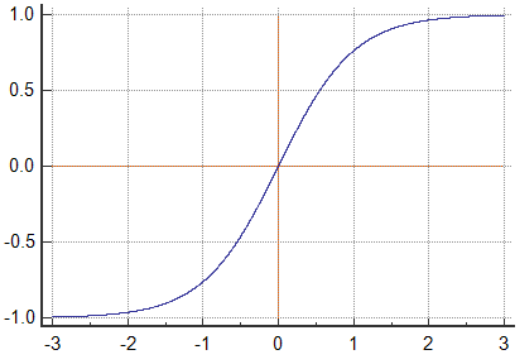


Figure 2.8: *Tanh Activation Function*

ReLU (Rectified Linear Unit) Function

The Rectified Linear Unit (ReLU) function is the most frequently used activation function in hidden layers. The main reasons for its popularity are its simplicity and its ability to overcome the limitations of other widely-used activation functions, such as Sigmoid and Tanh. It simply returns the input value if it's positive or 0 if it's negative. Therefore, the function is formally expressed as:

$$f(x) = \max(x, 0) \tag{8}$$

As illustrated in figure 2.9, the ReLU function is often referred to as the ramp function due to its graph representation.

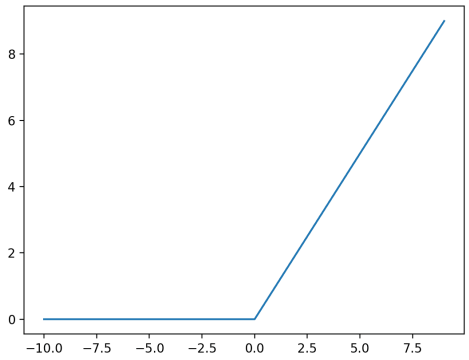


Figure 2.9: *RELU Activation Function*

Softmax Function

The Softmax function is a mathematical function that transforms a vector of numbers into a vector of probabilities, where the probabilities sum up to one. It is commonly used in the final layer of a neural network-based classifier, particularly in multiclass classification problems. The function computes the output for each i_{th} value of the input vector as follows:

$$f(x_i) = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \tag{9}$$

where N represents the length of the input vector or the number of classes in the classification problem. The function is depicted in figure 2.10

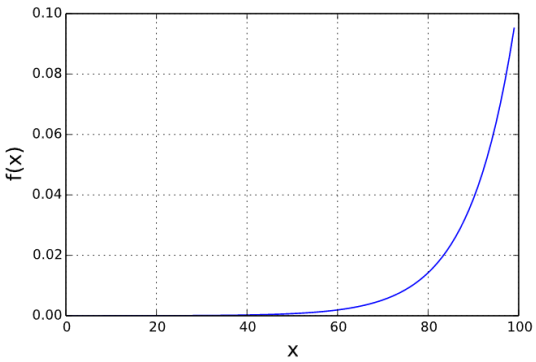


Figure 2.10: *Softmax Activation Function*

Binary Step Function

The Binary Step Function is essentially a threshold-based activation function. If the input value exceeds a certain threshold, the neuron is activated, and the output is set to 1. Conversely, if the input value does not reach the threshold, the neuron is deactivated, and the output is set to 0. The function is illustrated in figure 2.11

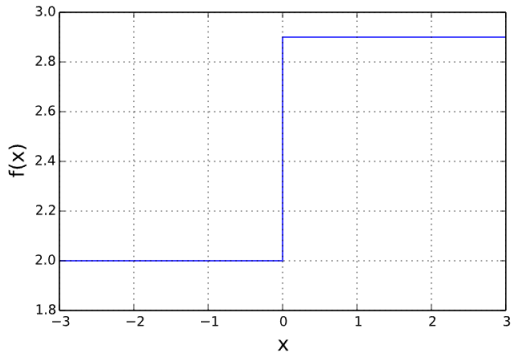


Figure 2.11: *Binary Activation Function*

2.3.10 Back propagation

As mentioned earlier, each input of a neuron is multiplied by its corresponding weight, where weights signify the potency or importance of the given input. A bias variable is then introduced into the equation to ensure that even when all input is zero, the neuron is capable of generating an output - this is shown in figure 2.5. The sum of weighted inputs and bias, represented by the equation $Z = w_1 * i_1 + ... + w_n * i_n$, is then passed through an activation function. The activation function applies a transformation to this sum before it is sent to the next layer of neurons, enabling the model to capture non-linearities and complex patterns in the data.

The process continues until it reaches the output layer, which provides the final prediction. This process is known as forward propagation. An error metric is then calculated based on the difference between the model's prediction and the actual values. If the error exceeds a certain threshold, a corrective process known as back propagation is initiated.

Back propagation is a widely used algorithm designed to optimize the weights of the neural network, with the objective to minimize the error between the predicted and actual values. The particular method used for updating the weights is dependent on the selected optimizer, a function that adjusts various attributes of the neural network, such as learning rate, in order to reduce the cost. The back propagation algorithm computes the gradient (direction and rate of fastest change) of the loss function with respect to each weight by employing the chain rule, a fundamental principle in calculus. It does this efficiently by processing one layer at a time, beginning from the final layer and moving backward. This process can be clearly seen in 2.12

The evaluation of the model's performance (the discrepancy between the predicted 's' and actual 'y' outcomes) is typically quantified using a cost function. In the context of neural networks, 's' usually represents the scores or the raw outputs of the last layer of neurons, and 'y' represents the actual values or labels. The choice of cost function can range from simple methods like Mean Squared Error (MSE) for regression tasks, to more complex ones like cross-entropy for classification tasks.

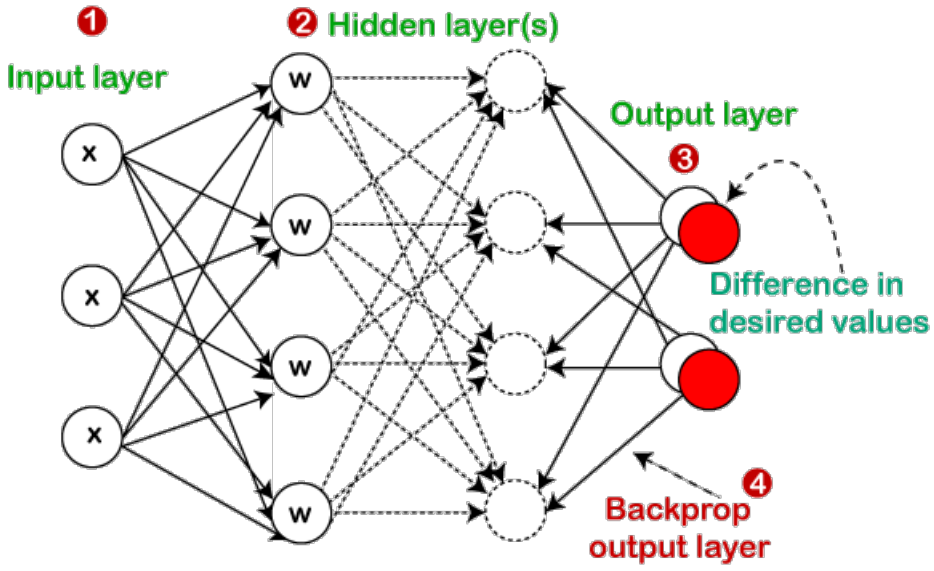


Figure 2.12: *How back propagation works*

These functions provide a measure of how well the network's predictions align with the actual values, thereby guiding the optimizer on how to adjust the weights and biases to minimize the error during back propagation.

2.3.11 Loss Functions

In machine learning, particularly in the realm of neural networks, a key concept that governs the optimization of our model is the "loss function". Broadly speaking, a loss function quantifies the extent to which the predicted output of our model deviates from the actual output. The ultimate goal during the training process is to minimize this loss - the less the loss, the better our model is at accurately mapping inputs to the correct outputs.

In the context of neural networks, each input is processed through various layers of artificial neurons, with each neuron applying specific weights and biases to the input, as depicted in the equation $Z = w_1 * i_1 + ... + w_n * i_n$, This processing stage, known as forward propagation, results in a predicted output.

After obtaining the predicted output, we need to assess how far off our prediction is from the target output. Here's where the backpropagation process, along with the loss function, comes into play. Backpropagation helps adjust the weights and biases of our model such that the output moves closer to the target output in the subsequent iterations.

The loss function plays an integral role in this adjustment process. By comparing the target and predicted output values, it provides a measure of the model's performance - the disparity between what our model predicts and what it should predict. This 'loss' is then used to update the model's hyperparameters, i.e., the weights and biases, thereby nudging the model towards better performance.

Loss functions can be broadly classified into groups depending on the learning task they're used for. Here are some of the most common loss functions categorized by the type of learning task

For Regression Tasks:

1. **Mean Squared Error (MSE):** offers several distinctive features that make it an excellent choice for computing loss. The squaring of the difference ensures that it's irrelevant whether the predicted value surpasses or falls short of the target value; but, it notably penalizes large errors. Moreover, as MSE possesses a convex nature (illustrated in the given diagram), it clearly outlines a global minimum. This characteristic simplifies the application of gradient descent optimization for weight value determination.. If y_i are the actual values and $\hat{y}_i$ are the predicted values, the MSE is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)}) \tag{10}$$

2. **Mean Absolute Error (MAE):** calculates the average of the absolute disparities between the predicted and target outputs. In certain scenarios, it serves as a replacement for MSE. As stated earlier, MSE is considerably influenced by outliers, causing a significant impact on the loss due to the squaring of the distance. When the training data contains a high proportion of outliers, MAE is preferred as it helps alleviate this issue. The MAE is calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{11}$$

   It also has some disadvantages; as the average distance approaches 0, gradient descent optimization will not work, as the function's derivative at 0 is undefined (which will result in an error, as it is impossible to divide by 0).

For Classification Tasks:

1. **Cross-Entropy Loss (Log Loss):** Designed to categorize an input into one of two predefined categories. Classification-oriented neural networks operate by producing a vector of probabilities that represent the likelihood of the given input fitting into each predefined category. The category with the highest associated probability is then chosen as the final output. In binary classification scenarios, only two possible actual values for y exist — either 0 or 1. Consequently, in order to accurately compute the loss between the real and predicted values, the method compares the real value (either 0 or 1) with the probability that the input belongs to that category (p(i) being the probability that the category is 1 and 1 - p(i) representing the probability that the category is 0). The equation for binary cross-entropy loss is:

$$L(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{12}$$

For multi-class classification problems, the cross-entropy loss is extended to sum over all classes.

For Tasks Involving Probability Distributions:

1. **Kullback-Leibler (KL) Divergence:** quantifies the divergence or dissimilarity between two probability distributions. It is most commonly used in situations involving unsupervised learning or probability-based tasks.

   The KL Divergence calculates the difference between two probability distributions by essentially assessing how one probability distribution (P) deviates from a second expected probability distribution (Q). The divergence measure signifies the amount of information lost when Q is used to approximate P.

   In the field of machine learning, this concept can be extremely beneficial in scenarios like model compression where we aim to approximate a complex model (P) with a simpler one (Q) and we wish to quantify the information loss incurred due to this approximation. Thus, minimizing KL Divergence ensures that the simpler model (Q) is a good approximation of the complex model (P). It is calculated as:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{13}$$

   Here, the sum is taken over all possible events i. In the context of neural networks, P(i) is the true distribution and Q(i) is the predicted distribution.

2.3.12 Optimization Algorithms

Optimizers play a paramount role in the domain of machine learning. They're essentially algorithms or methodologies utilized to diminish the error function - often referred to as the loss or cost function - or to enhance the efficiency of the system's output. An intriguing facet of optimizers is their reliance on the model's learnable parameters, predominantly the weights and biases. The purpose of optimizers is to instruct the model on modifying its weights and learning rates, with the ultimate goal being the reduction of losses.

In the subsequent sections, we'll delve into various types of optimizers, exploring their inner workings, and comprehending how they strive to minimize the loss function.

Gradient Descent

Gradient descent is a technique for optimization that is based on a convex function. This algorithm systematically adjusts its parameters in order to minimize the

provided function to its local minimum. The process of gradient descent continually diminishes the loss function by shifting in the direction diametrically opposed to that of the steepest incline. This algorithm relies heavily on the derivatives of the loss function to pinpoint minima. Notably, it employs data from the complete training set to compute the gradient of the cost function relative to the parameters, which demands significant memory and consequently decelerates the process. The equation is :

$$W_{new} = W_{old} - a * \frac{\partial(Loss)}{\partial(W_{old})} \tag{14}$$

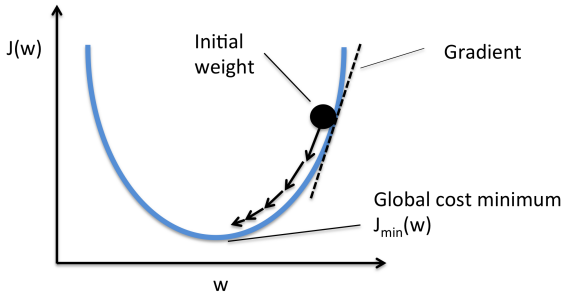


Figure 2.13: *Gradient Descent*

Speaking of the advantages of Gradient Descent, its simplicity in understanding and implementation stands out. However, it does have some disadvantages. Since this method computes the gradient for the entire dataset in a single update, the calculations can be time-consuming. Moreover, it demands considerable memory, making it computationally expensive.

The concept of a Learning Rate plays an integral role in this algorithm. It essentially determines the size of the steps that gradient descent takes towards the direction of the local minimum. It effectively regulates the pace at which we will gravitate towards the optimal weights.

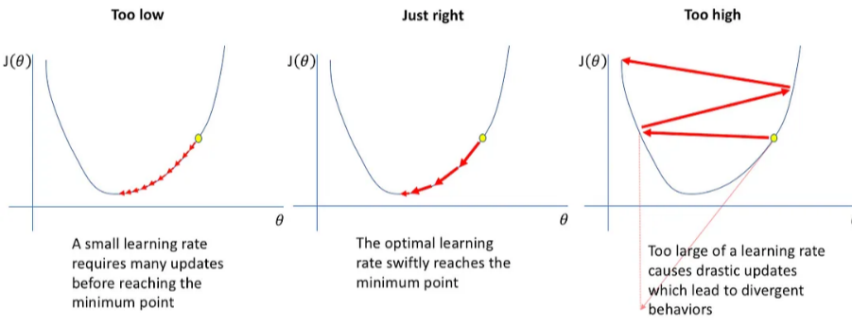


Figure 2.14: *Importance of learning rate*

Stohastic Gradient Descent

Stochastic Gradient Descent (SGD) represents a modification of the Gradient Descent approach, distinct in that it updates the model parameters individually. For instance, if the model encompasses a dataset of 10,000, SGD would initiate an update of the model parameters 10,000 times.

Stochastic Gradient Descent carries numerous advantages. For starters, the model parameters undergo frequent updates, which can accelerate the learning process. Additionally, it demands less memory, making it a more efficient choice for handling large datasets as it updates one example at a time.

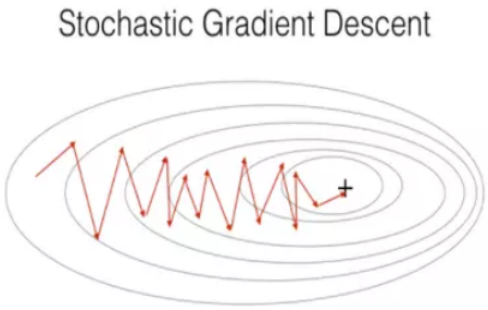


Figure 2.15: *Stochastic Gradient Descent*

However, Stochastic Gradient Descent is not without its challenges. The high frequency of updates can lead to noisy gradients, potentially causing an increase in error rather than the desired decrease. Another hurdle is the high variance that comes along with it. Plus, frequent updates can be computationally expensive, posing challenges when computational resources are limited.

Adaptive Gradient Descent (AdaGrad)

AdaGrad represents a significant shift in our approach to optimization algorithms, particularly in how it deals with the learning rate. While in previous methodologies, we kept the learning rate constant, AdaGrad introduces a fresh perspective. Its underlying principle is to assign distinct learning rates to individual neurons within each hidden layer, and this changes based on the specific iteration. The equation is :

$$W_{new} = W_{old} + \frac{a}{\sqrt{cache_{new}} + \varepsilon} * \frac{\partial(Loss)}{\partial(W_{old})} \tag{15}$$

Examining the benefits of AdaGrad, two primary advantages come to the fore. Firstly, with AdaGrad, the learning rate adaptively changes as iterations progress, making the model more responsive to the learning process. Secondly, it shows remarkable capability when it comes to training sparse data.

However, AdaGrad is not without its flaws. Specifically, when applied to deep neural networks, the learning rate can diminish to a very small number, an issue colloquially termed as the "dead neuron" problem. This arises due to the accumulated squared gradients in the denominator that leads to an exceedingly small learning rate, hampering the overall learning process.

RMS-Prop (Root Mean Square Propagation)

The RMS-Prop algorithm is an alteration of Adagrad. Unlike the traditional Adagrad method where squared gradients are accumulated over time, RMS-Prop utilizes an exponential moving average of the gradients. This subtle yet significant change combines the characteristics of momentum and AdaGrad, offering a unique approach to optimization. The RMS-prop equation is :

$$cache_{new} = \gamma * cache_{old} + (1 - \gamma) * (\frac{\partial(Loss)}{\partial(W_{old})})^2 \tag{16}$$

RMS-Prop shines in its capability to independently adjust the learning rate. This means it can assign different learning rates to each parameter, providing a more customized optimization approach.

On the other side, a potential downside associated with RMS-Prop is its slower learning progression during the model training phase. Nevertheless, in the realm of optimization algorithms, RMS-Prop remains a popular choice, attributable to its flexibility and reliable performance.

Adadelta

Adadelta, often viewed as an extension of Adagrad, aims to address the overly aggressive and monotonically decreasing learning rate often associated with Adagrad. In doing so, it successfully mitigates the issue of a decaying learning rate. Interestingly, Adadelta operates without requiring a default learning rate. Instead, it makes use of a ratio, specifically the ratio of the running average of previous time steps to the current gradient.

When one begins to weigh the pros and cons of Adadelta, a particular advantage stands out: there's no necessity to set a default learning rate. This feature can be a significant advantage in certain scenarios. However, there's no free lunch in machine learning, and this holds true with Adadelta as well. The primary trade-off for using Adadelta is that it's computationally expensive, which might make it less feasible for more complex or larger scale tasks.

Adam(Adaptive Moment Estimation)

Counted among the most famous gradient descent optimization algorithms, the Adam optimizer stands tall. Adam, an acronym for "Adaptive Moment Estimation", is a unique method capable of calculating adaptive learning rates for every individual parameter. Intriguingly, it retains both the decaying average of past gradients, similarly to momentum, and the decaying average of past squared gradients,

reminiscent of RMS-Prop and Adadelta. By doing so, it harmoniously amalgamates the benefits offered by both of these methods. Adam's equation is :

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{S_{dw_t} - \varepsilon}} * V_{dw_t} \tag{17}$$

$$b_t = b_{t-1} - \frac{\eta}{\sqrt{S_{db_t} - \varepsilon}} * V_{db_t} \tag{18}$$

The appeal of the Adam optimizer lies in its advantages. It stands out for its ease of implementation, making it a favorable choice among practitioners. Additionally, it is marked by its computational efficiency, ensuring swift and effective operation. Last but not least, the Adam optimizer is characterized by minimal memory requirements, making it a practical choice for a wide range of applications.

Conclusion :

To encapsulate the insights we've drawn from our examination of optimization methods, we can draw several key conclusions. When we're dealing with sparse data, it's beneficial to lean towards self-applicable methods, which include Adam, RMSprop, Adadelta, and Adagrad.

Often, you might observe that RMSprop, Adadelta, and Adam yield comparable outcomes across a range of scenarios. This shouldn't come as a surprise, given that Adam is essentially an extension of RMSprop, with additional features such as bias-correction and momentum. Consequently, in situations where the gradient becomes sparse, Adam generally outperforms RMSprop, delivering superior results.

Through understanding these characteristics, one can make more informed choices when selecting the most suitable optimizer for their machine learning tasks.

2.3.13 SVM

Support vector machine (SVM) [8] is a highly used algorithm that is used for both regression and classification problems. The objective of the support vector machine algorithm is to find a hyperplane in a N-dimensional space, where N is the number of features that can classify the data points. It uses a simple mathematical model $y = w * x\prime + \gamma$, and manipulates it to allow linear domain division[9]. SVM can be divided into linear and non-linear models [10]. Linear support vector machine can divide the data with a linear line or hyperplane to separate the classes in the original domain. On the other side, non-linear support vector machine indicates that the data domain cannot be divided linearly and can be transformed to a space called the feature space where the data can be divided linearly.

As shown in the Figure 2.16 there are many possible hyperplanes that can be chosen, in order to separate the two classes of data points. The purpose is to find the plane that has the maximum margin, thus the maximum, distance between data points from both classes, as illustrated in the Figure 2.17. The data points with the minimum distance to the hyperplane are called Support Vectors and influence the

position and the orientation of the hyperplane. If we delete the support vectors the position of the hyperplane will change. Using these support vectors we maximize the margin of the classifier, as depicted in Figure 2.18.

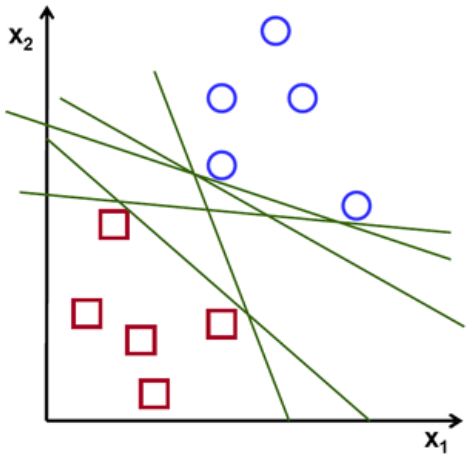


Figure 2.16: *Support vector machine possible hyperplanes*

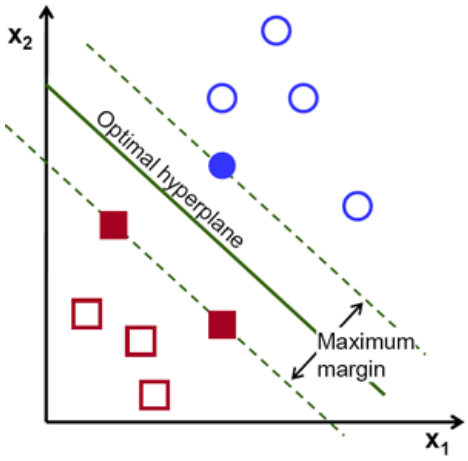


Figure 2.17: *Support vector machine optimal plane*

The simplest type of SVM as explained before is used for binary classification problems dividing the data point into two categories 0 or 1.

In order to use SVM for anomaly detection which is the scope of this thesis we use a variant called **One-class SVM (OCSVM)**

In the deployment of our system, we leverage the One-Class SVM as implemented in Scikit-Learn, which applies the formulation proposed by Schölkopf. This method aims to separate all data points from the origin in the high-dimensional feature space and seeks to maximize the distance from this separating hyperplane to the origin. In contrast to the conventional Support Vector Machines that discriminate between two distinct classes, this approach identifies a separating hyperplane in such a way that it is distant from the origin in the feature space. This leads to a decision

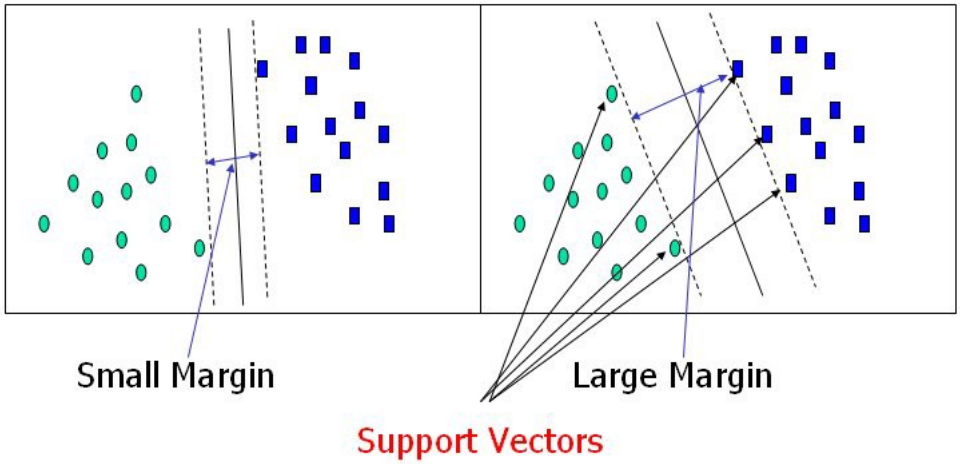


Figure 2.18: *Support vectors*

function that returns positive values for regions with high point density and negative values for those with low density. The hyperplane is defined as follows:

$$\begin{aligned} &\min_{\mathbf{w},\xi,\rho} \frac{1}{2}||\mathbf{w}||^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \\ &s.t. \quad (\mathbf{w}\cdot\phi(\mathbf{x}_i)) \geq \rho - \xi_i \quad \text{and} \quad \xi_i \geq 0, \quad i = 1,\ldots,n \end{aligned} \tag{19}$$

where $\mathbf{w}$ is the weight vector, ρ is the offset, ξ are the slack variables, $\phi(\mathbf{x}_i)$ is the mapped data point in the feature space, and ν is a parameter that controls the trade-off between maximizing the distance of the hyperplane from the origin and the fraction of outliers.

2.3.14 AutoEncoders (AE)

Autoencoders are a type of neural network typically used in an unsupervised setting. The aim of these networks is to learn to reconstruct input observations with minimal error. This concept may seem contradictory at first, as there seems to be no obvious need to produce an output that is a close replica of the input. However, the primary purpose of an autoencoder is not just reconstruction but to learn an "**informative**" representation of the data that can be leveraged for various applications [4].

The typical architecture of an autoencoder consists of three main components: an encoder, a latent feature representation(code or bottleneck or latent space), and a decoder .The encoder and decoder are essentially functions that work together to recreate the input, with the latent feature representation serving as a condensed, insightful representation of the input data.

In most cases, both the encoder and decoder parts of an autoencoder are constructed as neural networks. The encoder function generates the latent feature

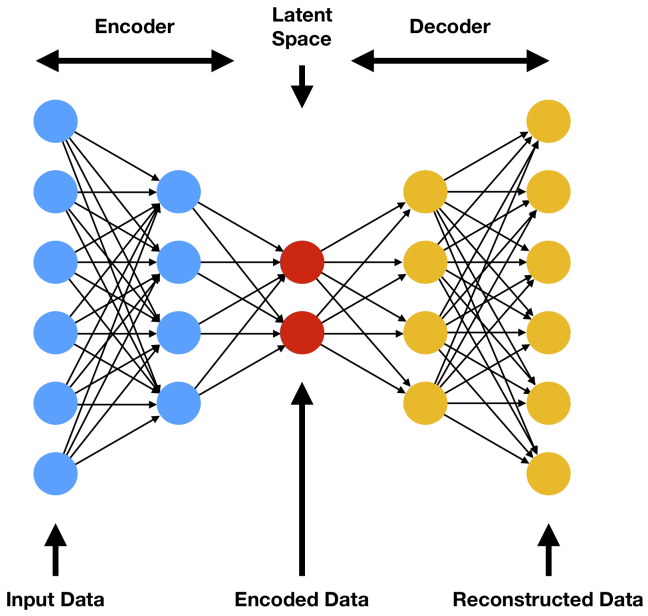


Figure 2.19: *AutoEncoders*

representation from the input data, and the decoder uses this representation to reconstruct the input.

In general, the **encoder** can be written as a function g that will depend on some parameters

$$h_i = g(x_i) \tag{20}$$

Where $h_i \epsilon \mathbb{R}^q$ (the latent space) is the output of the encoder block in Figure 2.19 when we evaluate it on the input x_i, Note we will have $g : \mathbb{R}^n \rightarrow \mathbb{R}^q$, The **decoder** (and the output of the network that we will indicate with $\hat{x}_i$) can be written then as a second generic function f of the latent space

$$\hat{x}_i = f(h_i) = f(g(x_i)) \tag{21}$$

Where $x_i \epsilon \mathbb{R}^n$, Training an AutoEncoder means simply finding the functions $g(.)$ and $f(.)$ that satisfy

$$arg \min_{f,g} < [\Delta(x_i, f(g(x_i))] > \tag{22}$$

Where Δ indicates a measure of how the input and the output of the AE differ (basically our loss function will penalize the difference between input and output), and $< . >$ is the average over all observations.

However, perfect reconstruction, which would mean learning the identity function, is not particularly useful. To avoid this, two strategies are often employed:

creating a bottleneck and adding regularization. A "bottleneck" is created by reducing the dimensionality of the latent features to be lower than that of the input data.

Autoencoders are useful in various tasks, such as dimensionality reduction, classification, denoising, and anomaly detection, primarily due to their ability to learn meaningful representations from unlabeled observations. For the scope of thesis we will focus on the task of anomaly detection.

Reconstruction Error

The reconstruction error (RE) is a metric that gives you an indication of how good(or bad) the autoencoder was able to reconstruct the input observation xi. The most typical RE used is the MSE

$$RE \equiv MSE = \frac{1}{M} \sum_{i=1}^{M} |x_i - \hat{x_i}|^2 \tag{23}$$

That can be easily calculated. The RE is used often when doing anomaly detection with autoencoders, as we will explain later. There is an easy intuitive explanation of the reconstruction error. When the RE is significant, the autoencoder could not reconstruct the input well, while when it is small,the reconstruction was successful.

Dimensionality of Latent space

Autoencoders are capable of learning useful data representations, especially when the dimensionality of the middle layer, also known as the "bottleneck," is less than the input dimensions. This reduction in dimensionality allows autoencoders to extract the essential features of the input data. This capability effectively turns autoencoders into tools for dimensionality reduction, offering a learned representation of the inputs.

However, an interesting aspect of autoencoders is that the quality of their performance is sensitive to the size of the bottleneck. If the dimensionality of the middle layer is reduced excessively, the autoencoder might fail to adequately reconstruct the input data. When the number of neurons in the middle layer is too small, the autoencoder might struggle to encode the necessary information about the input, leading to inaccuracies in the reconstructed output.

This sensitive balance between the size of the latent space and the performance of an autoencoder underscores the careful balance that must be struck during the model design process [4]. It reinforces the notion that while a smaller latent space can encourage the model to learn more robust and general features, shrinking it excessively can impede the model's ability to reconstruct the input effectively. It's a delicate dance between feature compression and the preservation of necessary information, a balance pivotal to the successful deployment of autoencoders.

Anomaly Detection

Autoencoders are a valuable tool for anomaly detection across diverse datasets, including those related to network traffic flow, such as the CIC-IDS 2017 dataset. This model operates under the assumption that when trained only on normal data, it will produce higher reconstruction errors for anomalies due to their unfamiliar nature.

In the context of network traffic flow, "normal data" would be the patterns of regular, non-malicious network activity. The autoencoder learns to reconstruct these normal traffic patterns. Anomalies, in this case, could be patterns that represent different types of network attacks or intrusions, which the model has not encountered during the training phase.

However, it's essential to remember that for this anomaly detection to work, the model must be trained on a dataset that does not contain anomalies, or in this context, the network traffic data must be free of intrusion attempts or malicious activity. The underlying assumption is that these anomalies or intrusions make up a negligible part of the whole dataset, thus, won't significantly influence the autoencoder's learning process.

Regularization plays a crucial role in this setup, as it prevents the autoencoder from simply learning the identity function, which would impair its ability to detect anomalies. Instead, it encourages the autoencoder to learn the essential features of the normal traffic patterns, making it more likely to produce high reconstruction errors for patterns it hasn't seen during training - the anomalies or intrusions.

Hence, when an unusual network traffic pattern is input into the autoencoder, it is likely to be reconstructed poorly, resulting in a high reconstruction error. This high error can be flagged as potential network intrusion, making autoencoders an effective tool for anomaly detection in network traffic data, as exemplified in the use of datasets like CIC-IDS 2017.

2.3.15 Denoising AutoEncoders (DAE)

Denoising Autoencoders are a variation of the standard autoencoder, conceived to enhance model robustness and prevent overfitting. This overfitting can become an issue when the model parameters outnumber the available data points, causing the autoencoder to simply learn the identity function.

The design of Denoising Autoencoders [11] involves intentional corruption of the input data by adding noise or masking some input vector values randomly. Surprisingly, the model's aim is not to reconstruct the noisy input but rather to restore the original, uncorrupted input.

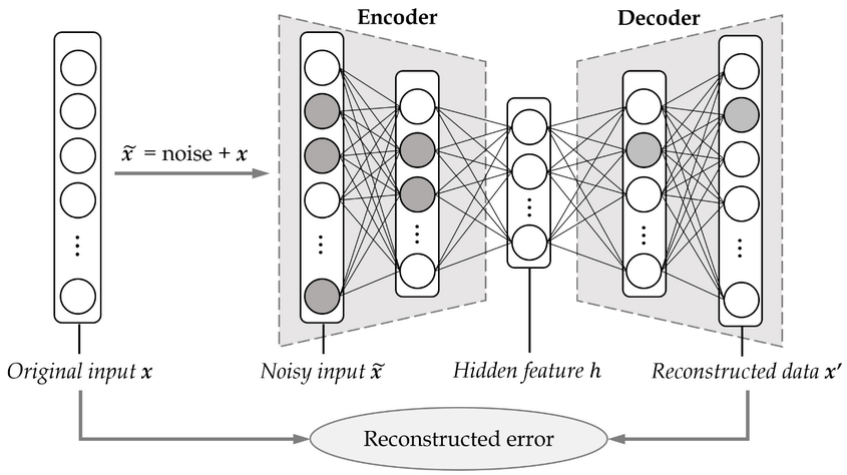


Figure 2.20: *Denoising AutoEncoders*

This corruption process is symbolized by a stochastic mapping, but it isn't tied to any specific type of noise (be it Gaussian, masking, salt-and-pepper, etc.). Furthermore, this process can incorporate prior knowledge.

$$\tilde{x}^{(i)} \sim M_D(\tilde{x}^{(i)}|x^{(i)}) \tag{24}$$

$$L_{DAE}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_\theta(g_\phi(\tilde{x}^{(i)})))^2 \tag{25}$$

where M_D defines the mapping from the true data samples to the noisy or corrupted ones.

The essence of this design is based on human ability to recognize objects or scenes, even when they are partially obscured or altered. The Denoising Autoencoder needs to uncover and comprehend the inter-dimensional relationships within the input to reconstruct the missing parts accurately.

This approach becomes particularly effective with high-dimensional, highly redundant inputs like images. The model typically relies on multiple input dimensions to generate the denoised version, avoiding overfitting to a single dimension and ensuring a robust latent representation. However there are applications also in anomaly detection. DAEs are particularly useful for anomaly detection in high-dimensional data or when the data contains complex relationships, where traditional statistical methods might not work well. They can also handle different types of noise and corruption in the data, making them versatile for various anomaly detection applications.

2.3.16 Variational AutoEncoders (VAE)

The essence of a Variational Autoencoder (VAE) [5] lies less in its similarity to traditional autoencoders and more in its roots in variational Bayesian methods and graphical models. [6]

Instead of transforming the input into a static vector, the idea is to map it onto a distribution. Let's denote this distribution as p_θ, parameterized by θ. The association between the data input $\mathbf{x}$ and the latent encoding vector $\mathbf{z}$ can be entirely characterized by:

- The prior $p_\theta(\mathbf{z})$
- The likelihood $p_\theta(\mathbf{z}—\mathbf{x})$
- The posterior $p_\theta(\mathbf{x}—\mathbf{z})$

Suppose we know the actual parameter θ^* for this distribution. To produce a sample resembling a real data point $\mathbf{x}^{(i)}$, we proceed as follows:

1. First, we draw a sample $\mathbf{z}^{(i)}$ from a prior distribution $p_{\theta^*}(\mathbf{z})$.
2. Then, a value $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(\mathbf{x}—\mathbf{z} = \mathbf{z}^{(i)})$.

The optimal parameter θ^* is the one that maximizes the likelihood of generating genuine data samples:

$$\theta^* = arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)}) \tag{26}$$

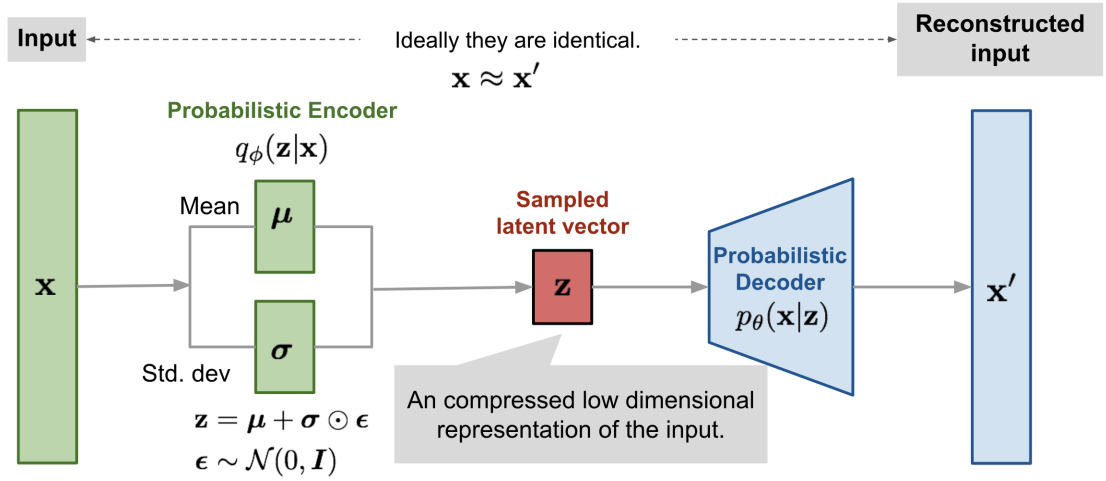


Figure 2.21: *Illustration of variational autoencoder model*

Loss Function: ELBO The loss function of a Variational Autoencoder (VAE) consists of two parts: the reconstruction loss and the KL divergence.

Reconstruction Loss (RL): This part of the loss function measures how well the VAE can reproduce the input data after having encoded it into a latent space and decoded it back, Input data are continuous so the reconstruction loss can be calculated using the Mean Squared Error (MSE). The MSE measures the average squared differences between the original input and the reconstruction.

In the case of using MSE for the reconstruction loss, the equation would be:

$$RL \equiv MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_i')^2 \tag{27}$$

Here, x_i refers to the original input data, and x_i' is the reconstructed data. This loss ensures that the VAE can accurately recreate the inputs in the case of continuous data. This measures how closely the learned latent variable distribution aligns with a standard Gaussian distribution.

KL divergence(KLD): This part of the loss function measures how closely the learned latent variable distribution aligns with a standard Gaussian distribution. The reason for encouraging this alignment is to ensure that the latent space has good properties as a representation, particularly that it is continuous and allows for interpolation.

The VAE loss function is a combination of these two loss terms:

$$\textbf{VAE } Loss = RL + KLD \tag{28}$$

The reconstruction loss ensures that the VAE can accurately recreate the inputs, while the KL divergence ensures that the learned distributions of latent variables adhere to a standard Gaussian distribution. The latter is important as it helps in creating a well-structured latent space where similar data points are grouped together, enabling easier sampling and interpolation.

Reparametrization Trick

The expectation term in the loss function invokes generating samples from $\mathbf{z} \sim \mathbf{q}_\phi(\mathbf{z}|\mathbf{x})$ [6]. Sampling is a stochastic process and therefore we cannot backpropagate the gradient. To make it trainable, the reparameterization trick is introduced: It is often possible to express the random variable $\mathbf{z}$ as a deterministic variable $\mathbf{z} = \mathcal{T}_\phi(\mathbf{x}, \epsilon)$, where ϵ is an auxiliary independent random variable, and the transformation function $\mathcal{T}_\phi$ parameterized by ϕ converts ϵ to $\mathbf{z}$.

For example, a common choice of the form of $q_\phi(\mathbf{z}—\mathbf{x})$ is a multivariate Gaussian with a diagonal covariance structure:

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)}I) \tag{29}$$

$$\mathbf{z} = \mu + \sigma \odot \epsilon, where\ \epsilon \sim \mathcal{N}(0, I) \tag{30}$$

where $\odot$ refers to element-wise product.

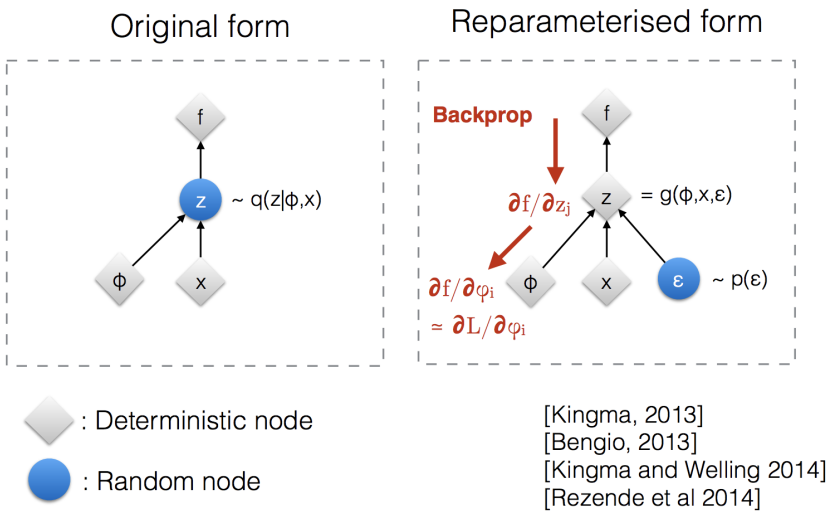


Figure 2.22: *Illustration of how the reparameterization trick makes the sampling process trainable*

The reparameterization trick works for other types of distributions too, not only Gaussian. In the multivariate Gaussian case, we make the model trainable by learning the mean and variance of the distribution, μ and σ, explicitly using the reparameterization trick, while the stochasticity remains in the random variable $\epsilon \sim \mathcal{N}(0, I)$

2.3.17 beta-Variational AutoEncoders (beta-VAE)

The foundational concept behind $\beta - VAE$ is its modification of the Variational Autoencoder's objective function, specifically, the way it balances the reconstruction loss and the KL-divergence.

When each variable in the inferred latent representation $\mathbf{z}$ responds primarily to a single generative factor and remains relatively unaffected by other factors, this representation is described as disentangled or factorized. One advantage that often accompanies disentangled representation is strong interpretability and effortless extrapolation to a variety of tasks.

Consider a model trained on images of human faces. It may capture characteristics like the smile, skin tone, hair color, hair length, expression, presence of glasses, and many other relatively independent factors in separate dimensions. Such a disentangled representation is highly beneficial for facial image generation.

β-VAE [12] is a modification of the Variational Autoencoder that places a particular emphasis on the discovery of disentangled latent factors. As with the VAE,

the aim is to maximize the probability of generating real data while minimizing the distance between the real and estimated posterior distributions.

In a standard VAE, the objective function (also known as the Evidence Lower Bound, or ELBO) can be defined as:

$$\text{ELBO} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{31}$$

Here, the first term represents the expected reconstruction loss and the second term is the KL-divergence that measures how much the learned distribution $q_\phi(z|x)$ over the latent variables z deviates from the prior distribution $p(z)$ (usually a standard normal distribution).

The β-VAE modifies this objective by introducing a hyperparameter, β, that scales the KL-divergence term:

$$ELBO_\beta = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta \cdot D_{KL}(q_\phi(z|x)||p(z)) \tag{32}$$

- For $\beta = 1$, the β-VAE reduces to the standard VAE.
- When $\beta > 1$, the model places more emphasis on making the latent variable distribution align more closely with the prior, encouraging disentangled or factorized representations.
- However, the case of $\beta < 1$ is particularly interesting for anomaly detection. In this scenario, the model reduces the emphasis on the alignment with the prior and focuses more on the reconstruction. This can be beneficial for anomaly detection because the model is encouraged to learn to reconstruct normal data very accurately, and as a result, anomalies, which deviate from the normal data, are more likely to have higher reconstruction errors. Therefore, the β-VAE model with $\beta < 1$ could potentially yield better performance in anomaly detection tasks.

Chapter 3

Related Work

Anomaly Detection refers to a binary classification task, where the goal is to categorize samples as either normal or outliers. While our dataset includes different classes of attacks that could be useful for certain tasks, our primary aim is not just to classify a sample into benign or some type of attack. Instead, we strive to successfully detect zero-day attacks, which represent unseen types of threats. Numerous surveys have explored the capabilities of Machine Learning (ML) and Deep Learning (DL) algorithms in achieving this goal, discussing various learning paradigms. These include supervised learning algorithms such as Support Vector Machine (SVM) for classification problems and Random Forest for classification and regression problems, and unsupervised techniques like Principal Component Analysis (PCA) and clustering techniques like Self-Organizing Map (SOM) [13], [14]. Some studies have specifically highlighted the power of deep learning techniques like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), demonstrating their advantages over shallow techniques of the same learning type. On the unsupervised front, Generative Adversarial Networks (GANs) and Autoencoders are common subjects of investigation [15], [16]. While these surveys provide a comprehensive view of the intrusion detection problem and zero-day attacks, it is crucial to delve deeper into more advanced approaches. To this end, we discuss two primary learning paradigms: Supervised learning and unsupervised learning, elaborating on each in the following subsections.

3.1 Supervised learning

Several supervised learning techniques aim to predict attacks with a higher success rate, especially with fewer false positives. For instance, Yuan et al. [13] proposed a method called DeepDefense, which utilizes historical patterns across various RNN models. Their approach is unique as most existing methods are single-packet-based, which can limit performance due to the lack of historical patterns in the learning model. This makes it challenging for such models to detect low-rate attacks, which may appear similar to legitimate network traffic from the victim's end. In their experiments, Yuan [13] varied the time-stamp window sizes in the memory, trying

out 1, 5, 10, 50, and 100 time-stamps. Larger window sizes can store longer attack time sequences and reflect more integrated attack activities. They treated DDoS detection as a sequence classification problem, transforming packet-based DDoS detection to window-based detection.

An interesting work by Sarhan, Mohanad et al. [16] proposed a unique learning methodology using common architectures. Their Zero-shot Learning (ZSL) approach is designed to evaluate and improve the generalizability of ML models to new or unseen data classes. This technique is based on the assumption that the training dataset may not cover the entire set of classes the ML model could encounter once deployed in the real world. Hence, ZSL aims to address the continuously growing set of classes, making it impractical to collect training samples for each one of them. This technique involves two stages: attribute learning and inference. During the attribute learning stage, the model learns distinguishing semantics of the missing class. In the inference stage, ZSL uses the learned attributes to predict or classify certain samples belonging to the missing data class. ZSL addresses one of the main challenges in building a reliable ML-based Network Intrusion Detection System (NIDS) - evaluating the detection of new attack classes that are unavailable during the training phase, such as zero-day attacks. They also introduce a new metric, the Zero-day Detection Rate, which measures how well a learning model can reconstruct the distinguishing semantics learned from known attack classes to detect unknown attack classes. A study by Pallaprolu et al. [17] proposed a pipeline for intrusion identification in streaming data using semantics and Spark, examining the aspect of deployment in a real-world application.

3.2 Unsupervised Learning

When dealing with data featuring unbalanced classes, such as our case where normal traffic constitutes the vast majority and attacks are a minority, unsupervised learning appears to be a promising option. Tang, Ruming, et al. [18] proposed a method named Zero Wall, an unsupervised approach that works in conjunction with an existing Web Application Firewall (WAF) in the pipeline. The core concept behind Zero Wall is training a self-translation machine learning Autoencoder Recurrent Neural Network (AE-RNN) to capture the syntax and semantic patterns of benign requests. This is based on the idea that HTTP benign requests resemble a language, and by using AE-RNN, we can capture the patterns of this language. As a result, the request as an input goes through the encoder and then the decoder translates it back into an HTTP request with slight differences. By using the BLEU metric as an indicator of the similarity between the original and recovered token sequences, we can determine whether the request belongs to the benign class. If the request is malicious, the translation will fail, and the request will be dropped. The robustness of AE-RNN is apparent in its performance with large volumes of benign requests, which make up the vast majority of Web logs. Hence, malicious requests should

have little impact on the learning of the encoder-decoder network. This is also confirmed by Hindy, Hanan, et al. [19], who compared AE with One Class SVM and found that One Class SVM would fail if contamination occurs. The robustness of AE was examined again by Madani et al. [20], where they intentionally tested AE by contaminating the retraining of AE with malicious examples labeled as benign. Nonetheless, the performance of AE was not significantly affected in comparison with the PCC.

Chapter 4

Dataset

4.1 Introduction to Datasets for Intusion Detection Systems

Intrusion Detection Systems (IDS) are essential for securing computer networks and systems against malicious attacks. Evaluating the performance of IDS is critical in ensuring their effectiveness. One critical aspect of IDS evaluation is the selection of datasets used for training and testing. The availability of reliable and relevant datasets is crucial in ensuring the performance of IDS.

One of the popular datasets for IDS evaluation is the Canadian Institute for Cybersecurity Intrusion Detection Dataset (CICIDS2017), which is a labeled dataset that contains both benign traffic and various types of attacks. The dataset was created by the Canadian Institute for Cybersecurity [21],

Among others that are the NSL-KDD, the UNSW-NB15, and the ISCX 2012 datasets which are well-known resources for evaluating intrusion detection systems. However, each dataset has its limitations and issues, which are addressed more effectively by the CICIDS 2017 dataset.

The NSL-KDD dataset [1] suffers from redundancy, as it contains a significant number of duplicate records that could lead to biased evaluation results. It also has imbalanced classes, with the distribution of normal and attack instances making it difficult for some classifiers to perform well. Furthermore, the dataset includes older attack types that may not accurately represent the current threat landscape.

The UNSW-NB15 dataset [22] also has its drawbacks. It features a limited set of attributes, which can make it harder for intrusion detection systems to identify newer attack types. Additionally, it shares the class imbalance problem found in the NSL-KDD dataset, potentially affecting the performance of some classifiers. Finally, some instances in the dataset are ambiguously labeled, which could lead to inconsistencies in the evaluation of intrusion detection systems.

The ISCX 2012 dataset [23] is limited in terms of attack scenarios, focusing on a small number of them and not covering the entire spectrum of network intru-

sions. Additionally, the dataset is relatively small, limiting its ability to provide a comprehensive evaluation of intrusion detection systems. Furthermore, the documentation for the dataset's features is insufficient, making it difficult for researchers to understand and work with the data.

In contrast, the CICIDS 2017 dataset offers several significant improvements. It covers a wide range of attack types, providing a more comprehensive evaluation of intrusion detection systems. Its class distribution is more balanced, allowing for better evaluation of classifiers' performance. The dataset also includes a comprehensive set of features, enabling intrusion detection systems to better identify new and evolving attack types. Additionally, CICIDS 2017 is regularly updated to include new attack types and scenarios, ensuring that it remains relevant in the evolving threat landscape. Finally, the dataset is well-documented, making it easier for researchers to understand and work with the data.

4.2 CICIDS 2017

The CICIDS2017 dataset includes both benign and up-to-date common attacks, closely resembling real-world data (PCAPs). It also encompasses the outcomes of network traffic analysis using CICFlowMeter, with labeled flows based on time stamps, source and destination IPs, source and destination ports, protocols, and attacks (CSV files). Extracted features definitions are also provided.

The creation of realistic background traffic was a key priority during the dataset's development. The proposed B-Profile system was utilized to model the abstract behavior of human interactions and produce naturalistic benign background traffic. B-Profile for this dataset extracts the abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols. At First, it tries to encapsulate network events produced by users with machine learning and statistical analysis techniques. The encapsulated features are distributions of packet sizes of a protocol, the number of packets per flow, certain patterns in the payload, the size of the payload, and request time distribution of protocols. Then, after deriving the B-Profiles from users, an agent which has been developed by Java is used to generating realistic benign events and simultaneously perform B-Profile on the Victim-Network for predefined five protocols [21].

Data capturing took place from 9 a.m. on Monday, July 3, 2017, to 5 p.m. on Friday, July 7, 2017, spanning a total of five days. Monday served as the normal day, containing only benign traffic. Implemented attacks, such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS, were executed on Tuesday, Wednesday, Thursday, and Friday, both in the morning and afternoon sessions.

Considering the limitations of existing datasets, researchers have identified the need for a comprehensive and well-rounded framework for generating IDS/IPS bench-

marking datasets. In the following section, some of the features of such a framework are defined, which are covered by CICIDS 2017

Complete Network configuration: A complete network topology includes Modem, Firewall, Switches, Routers, and presence of a variety of operating systems such as Windows, Ubuntu and Mac OS X.

Complete Traffic: By having a user profiling agent and 12 different machines in Victim-Network and real attacks from the Attack-Network.

Labelled Dataset: Section 4 and Table 2 show the benign and attack labels for each day. Also, the details of the attack timing will be published on the dataset document.

Available Protocols: Provided the presence of all common available protocols, such as HTTP, HTTPS, FTP, SSH and email protocols.

Attack Diversity: Included the most common attacks based on the 2016 McAfee report, such as Web based, Brute force, DoS, DDoS, Infiltration, Heart-bleed, Bot and Scan covered in this dataset.

Heterogeneity: Captured the network traffic from the main Switch and memory dump and system calls from all victim machines, during the attacks execution.

Feature Set: Extracted more than 80 network flow features from the generated network traffic using CICFlowMeter and delivered the network flow dataset as a CSV file. See our PCAP analyzer and CSV generator.

4.2.1 Network Analysis Tool

CICFlowMeter is a network traffic flow generator which has been written in Java and offers more flexibility in terms of choosing the features you want to calculate, adding new ones, and having a better control of the duration of the flow timeout.

It generates Bidirectional Flows (Biflow), where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence the 83 statistical features such as Duration, Number of packets, Number of bytes, Length of packets, etc. are also calculated separately in the forward and reverse direction.

The output of the application is in CSV file format with six columns labeled for each flow, namely FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol with more than 80 network traffic features. Normally the TCP flows are terminated upon connection teardown (by FIN packet) while UDP flows are terminated by a flow timeout.

The flow timeout value can be assigned arbitrarily by the individual scheme, e.g. 600 seconds for both TCP and UDP. The CICFlowMeter-V3 can extract more than 80 features which are explained in the next section.

4.2.2 Features & Labels

In previous sections, we discussed network analysis tools and provided a general description of the dataset. In this section, we will present the features and labels of the dataset in greater detail. Effective network traffic analysis relies on examining various features that offer valuable insights into the behavior and content of the transmitted data. This section introduces the main categories of features included in the CICIDS2017 dataset and highlights their importance in understanding and characterizing network flows.

Basic Flow Features: Basic flow features describe fundamental characteristics of network traffic flow, such as the maximum, minimum, mean, and standard deviation of packet sizes in both forward and backward directions. These features can provide insights into the overall size and variability of packets transmitted within a network flow, which can help identify anomalies or patterns associated with specific types of attacks or network behaviors.

Flow Rate Features: Flow rate features describe the rate at which data (in bytes) and packets are transmitted in a network flow. These features help understand the network's capacity utilization, identify potential bottlenecks, and detect unusual traffic patterns that may indicate an ongoing attack or network congestion.

Inter-Arrival Time (IAT) Features: IAT features describe the time intervals between consecutive packets or flows in both forward and backward directions. Analyzing these features can help identify potential delays or irregularities in packet transmission, which could signal network issues or ongoing attacks, such as denial-of-service (DoS) or distributed denial-of-service (DDoS).

TCP Flag Features: TCP flag features focus on the different flags set in the TCP header of network packets. These flags, such as SYN, ACK, and FIN, indicate the state of a TCP connection and can reveal important information about the communication between hosts. Analyzing these flags can help detect malicious activities or anomalies, such as port scanning, network reconnaissance, or attempts to exploit specific vulnerabilities in the network.

Header Features: Header features describe the size of packet headers in both forward and backward directions. By analyzing these features, one can gain insights into the protocol overhead and the complexity of network traffic, which can help detect potential anomalies or malicious traffic patterns.

Packet Length Features: Packet length features describe the size distribution of packets within a network flow, including minimum, maximum, mean, standard deviation, and variance. These features can help identify abnormal packet sizes,

which may be indicative of network issues, data exfiltration, or specific types of attacks, such as buffer overflow or fragmentation attacks.

Bulk Features: Bulk features describe the average number of bytes, packets, and rate in bulk transfers for both forward and backward directions. Analyzing these features can help identify large-scale data transfers, which might be indicative of data exfiltration, file sharing, or other activities that may be of interest in a network security context.

Subflow Features: Subflow features describe the average number of packets and bytes in subflows within a network flow in both forward and backward directions. These features can help identify the granularity of network communication and detect potential data fragmentation or unusual subflow patterns, which could be associated with specific attacks or network behaviors.

TCP Window Features: TCP window features describe characteristics of the TCP window size, such as the initial window size in both forward and backward directions, the number of packets with at least 1 byte of TCP data payload, and the minimum segment size observed. Analyzing these features can help identify potential issues or anomalies related to TCP flow control, congestion, or network performance.

Active and Idle Features: Active and idle features describe the time intervals during which a network flow is active (transmitting data) or idle (not transmitting data). Analyzing these features can help identify patterns in network communication, detect potential periods of inactivity or excessive activity, and uncover potential network issues or ongoing attacks, such as low-and-slow attacks or command-and-control (C2) communication.

Miscellaneous Features: Miscellaneous features include various attributes such as the download/upload ratio, average packet size, and average segment size in both forward and backward directions. These features can provide additional insights into the nature and balance of network traffic, which can help identify potential anomalies or malicious activities.

In the table below we are going to show the distribution of classes inside the dataset.

Class	Samples	Sum
Brute Force	15,994	685,673
DoS	252,661	
DDoS	256,054	
Heartbleed Port 444	11	
SQL-injection	21	
Infiltration	36	
Botnet	1,966	
PortScan	158,930	
Benign	1,840,897	1,840,897

Table 4.1: *Class distribution*

One of the reason we chose CICIDS2017 is that it was designed for network security and intrusion detection applications, aiming to cover a wide range of attack scenarios. In this dataset, six attack profiles are created based on the most recent list of prevalent attack families, and these profiles are executed using relevant tools and code.

Brute Force Attack: This popular attack type is not only used for cracking passwords but also for discovering hidden pages and content within web applications. It essentially involves a trial-and-error approach until the attacker achieves success.

Heartbleed Attack: This attack originates from a flaw in the OpenSSL cryptography library, which is a commonly used implementation of the Transport Layer Security (TLS) protocol. It is typically exploited by sending a malformed heartbeat request with a small payload and large length field to a vulnerable target (usually a server) to provoke a response from the victim.

Botnet: This term refers to a collection of internet-connected devices controlled by a botnet owner to carry out various tasks. It can be used to steal data, distribute spam, and provide the attacker with access to the device and its connection.

DoS Attack: In this attack, the aim is to temporarily render a machine or network resource unavailable. It is usually carried out by one source which is inundating the targeted machine or resource with excessive requests, overloading systems and preventing legitimate requests from being processed.

DDoS Attack: This type of attack typically occurs when multiple systems overwhelm a victim's bandwidth or resources. Often, the attack is the result of multiple compromised systems (e.g., a botnet) bombarding the targeted system with massive network traffic.

Web Attack: These attack types are emerging daily, as both individuals and organizations take security more seriously. We use SQL Injection, in which an

attacker creates a string of SQL commands and uses it to force the database to reveal information, Cross-Site Scripting (XSS) that occurs when developers fail to adequately test their code for script injection possibilities, and Brute Force over HTTP, which attempts to crack administrator passwords by trying various password combinations.

Infiltration Attack: This type of attack involves infiltrating a network from within, usually by exploiting vulnerable software like Adobe Acrobat Reader. Once the exploit is successful, a backdoor is executed on the victim's computer, allowing the attacker to carry out various actions on the victim's network, such as IP sweeps, comprehensive port scans using Nmap.

4.3 Data preprocessing

The fundamental step towards reliable data analysis and modeling, data preprocessing stands as the cornerstone in the data science workflow. This subsection meticulously explores the data preprocessing phase of the thesis, underlining its critical importance in ensuring the validity and accuracy of subsequent experimental results.

The quality of the data we feed into our models considerably impacts the insights we can extract. Raw data, especially when dealing with large datasets like the one used in this thesis, often contains inconsistencies, missing values, outliers, and variables of different scales. All these factors can negatively impact the performance of the model and even lead to misleading results.

The purpose of data preprocessing is to cleanse, transform, and standardize data, thus making it suitable for analysis and modeling. This subsection provides a detailed account of the data preprocessing strategies employed, including data cleaning, normalization, feature selection, and dimensionality reduction. The techniques adopted are explained in the context of the specific requirements of the CICIDS 2017 dataset used in this study.

By the end of this section, it will be clear what data were used, how data preprocessing forms a robust foundation for the subsequent steps of the thesis, specifically the application of various autoencoder configurations for anomaly detection. This initial step sets the stage, ensuring the data are of high quality and in the most appropriate format for the sophisticated techniques to follow. Through an understanding of the preprocessing stage, we are better equipped to interpret the results of our experiments and discussions in the later chapters.

The original dataset comes in a format that's prepared and well-structured for Machine Learning applications, provided in CSV files. It contains a substantial amount of data, specifically **2,526,570 entries** spread over **79 features**. There are two ways of labeling included with this dataset.

The first label is used to mark the nature of the data, indicating whether it's benign or an attack, represented as 0 and 1, respectively. Meanwhile, the second label

is more specific, categorizing the particular type of network traffic. This includes references to the various forms of attacks like DDoS, Heartbleed, among others, as well as benign traffic.

Together, these two labels offer both a high-level and detailed view of the network traffic represented in the dataset, delineating between benign instances and various types of malicious activities.

4.3.1 Data Cleaning

The cleaning process includes three steps which are "**drop missing values**", "**drop infinite values**" and "**drop negative values**". These steps will be analyzed further below:

Dropping missing values refers to removing data entries where one or more features lack assigned values. These missing values can lead to inaccuracies or errors in the model's learning process, and they may also introduce bias in the resulting model. Therefore, it is crucial to identify and remove these entries, ensuring a more accurate and robust model.

The **elimination of infinite values** is another crucial data cleaning step. In certain situations, due to calculations like division by zero or logarithm of zero, infinite values can appear in the dataset. These values can drastically skew the model's learning process and distort the final results. Hence, it's necessary to detect any such values and exclude them from the dataset.

The final step in our data preprocessing phase is to **drop negative values**. This action is particularly essential considering the nature of our dataset; we are dealing with statistical features and fields that are inherently non-negative. Negative values not only introduce noise to the data, as they do not convey meaningful information relevant to our analysis, but they can also impede the feature selection process.

In feature selection, methods such as low variance filtering and correlation analysis are used. Negative values can disrupt these calculations, leading to unreliable results. For instance, they can artificially inflate the variance or skew the correlation coefficients, thereby misleading the feature selection process and impacting the final choice of features for the model.

Negative values can also affect subsequent preprocessing stages, such as scaling and standardization. Their presence could distort the calculated ranges, standard deviations, and mean values of the features, thereby skewing the normalized data and compromising the performance of our model. Therefore, the identification and removal of negative values is a critical step, significantly contributing to the integrity and effectiveness of our data preprocessing strategy.

Upon completion of the data cleaning pipeline, a significant reduction in the number of entries in the dataset was observed. Initially containing 2,526,570 entries, the meticulous application of cleaning processes resulted in a final count of 1,302,309 entries.

This sizable reduction, amounting to almost half of the original dataset, underscores the critical role of data cleaning in ensuring the quality of the dataset. Although the cleaning process led to a substantial decrease in the dataset's size, it effectively removed noise and inconsistencies, yielding a dataset of higher integrity and relevance for the subsequent phases of analysis.

This step in data preprocessing sets the stage for the ensuing stages of modeling and analysis, ensuring the results generated are reliable and representative of the underlying patterns in the cleaned data. Thus, the cleaning process, despite its reductive effect on the size of the dataset, is of paramount importance for the validity and credibility of the final results of our study.

4.3.2 Feature selection

The process of feature selection is integral to the successful deployment of machine learning models. In the context of large datasets with a plethora of features, it's often the case that not all features contribute equally to the predictive power of the model. Some features might be redundant, while others could introduce noise into the system. Thus, feature selection plays a key role in identifying the most meaningful and relevant features for the model, thereby improving its performance and interpretability.

With a focus on our dataset, comprising 79 features, two essential methods, low variance filtering and correlation analysis, are employed to streamline the feature set.

1. **Low-variance features**

   Near-zero or low variance features refer to those predictors in the dataset that have only a few unique values occurring with very low frequencies. They often exhibit a highly skewed distribution, with one value dominating the vast majority of the data points. [24]

   Below we demonstrate several reasons why it's important to consider dropping near-zero(low) variance, or low standard deviation, features:

   Influence on Model: Near-zero variance predictors might disproportionately influence the model due to their skewed distribution. A few outlier data points might unduly influence the model's learning, which could distort predictions and reduce overall model performance.

   Data Imbalance: The occurrence of a high ratio of the most common frequency to the second most common frequency in near-zero variance features indicates a severe imbalance in the data. This can bias the model towards the dominant value, leading to less accurate and less generalizable models.

Removing near-zero variance features, therefore, can help improve the model's performance and generalizability by reducing undue influences and addressing data imbalances.

Upon setting the variance threshold at 0.1 and implementing the low-variance filtering method, we were able to reduce the number of features in our dataset. This process resulted in the removal of 16 features. Consequently, the dataset now comprises 63 features, streamlining the dimensionality and potentially increasing the efficiency of our subsequent modeling process.

In the table Table 4.2 we can see the features that have $Variance \leq 0.1$

Table 4.2: *Low-Variance Features (thr=0.1)*

Fwd PSH Flags	Bwd PSH Flags
Fwd URG Flags	Bwd URG Flags
FIN Flag Count	SYN Flag Count
RST Flag Count	URG Flag Count
CWE Flag Count	ECE Flag Count
Fwd Avg Bytes/Bulk	Fwd Avg Packets/Bulk
Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk	Bwd Avg Bulk Rate

2. **Correlation Analysis**

Highly correlated features in a dataset, while seemingly informative, can introduce problems in the modeling process. They are of concern for several reasons, and thus it becomes essential to consider their removal:

Redundancy: If two or more features are highly correlated, they essentially provide the same information. Retaining these redundant features does not add new information to the model and can lead to inefficiencies in computation and storage.

Model Complexity: Highly correlated features can unnecessarily increase the complexity of the model, making it harder to interpret and explain. Simplifying the model by removing these correlated features can lead to more straightforward, interpretable models without compromising predictive accuracy.

Overfitting: Highly correlated features can contribute to overfitting, where the model performs well on the training data but poorly on unseen data. This happens because the model may rely too heavily on these features, which may not behave the same way in the test data.

By identifying and removing highly correlated features, we can alleviate these issues and improve the stability, interpretability, and generalizability of our models.

We carried out a correlation analysis, the results of which are visually represented in the correlation matrix presented below.

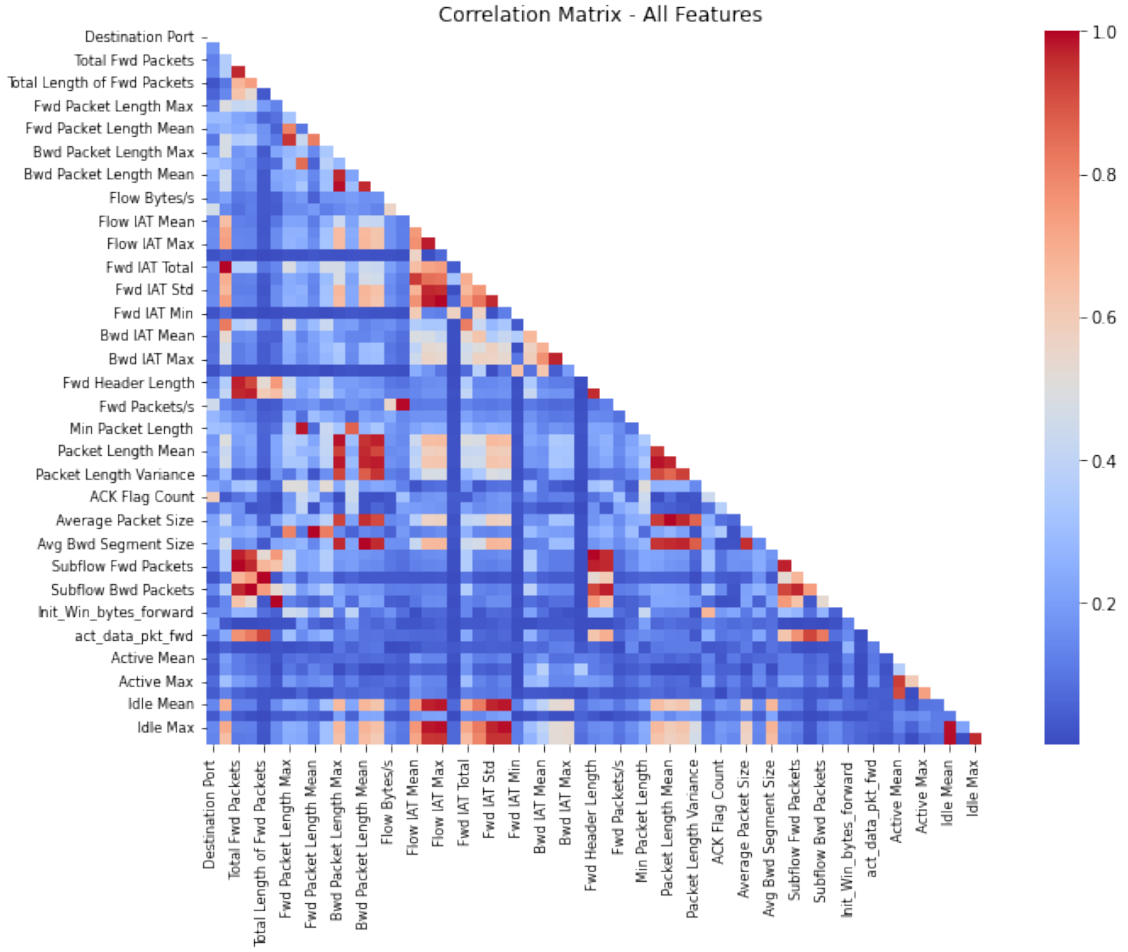


Figure 4.1: *Correlation Matrix*

In light of the results derived from the correlation analysis, a threshold of 0.9 was employed as a marker to identify features that exhibited high correlation. Notably, not all of these high correlation features were slated for removal.

This decision was informed by the significance of these features as delineated in the CICIDS 2017 paper. Certain features, despite their high correlation with others, were underscored as critically important for classifying the data. Consequently, these particular features were retained in our dataset.

The reasoning behind this selective feature retention revolves around the value of the information these features contribute. As per the assertions of the CICIDS 2017 creators, these attributes impart essential information which cannot be disregarded for an accurate and comprehensive data analysis. Hence, in our endeavor to balance feature correlation and importance, these specific high-correlation features were deemed necessary and thus preserved in the dataset.

As per our analysis, and considering the insights provided by the dataset creators regarding feature importance, we identified 23 features for removal. These features exhibited a high correlation exceeding the designated threshold of 0.9. Table 4.3 enumerates all such features that were dropped due to their high correlation.

Table 4.3: *Highly Correlated Features (thr=0.9)*

Fwd IAT Max	Fwd IAT Total
Bwd IAT Max	Avg Bwd Segment Size
Subflow Fwd Packets	Packet Length Std
act_data_pkt_fwd	Max Packet Length
Packet Length Variance	Idle Max
Fwd IAT Std	Bwd Packet Length Mean
Fwd Header Length	Total Backward Packets
Idle Mean	Min Packet Length
Idle Min	Active Max
Fwd Packet Length Std	Subflow Bwd Packets
Packet Length Mean	Avg Fwd Segment Size
Bwd Header Length	

Finalizing the feature selection process has yielded a refined dataset. Now, our dataset comprehensively encapsulates **1,302,309 entries** and is characterized by **40 features**, along with the 2 labels that we described earlier.

4.3.3 Scaling

Deep Neural Network models often benefit significantly from data inputs that adhere to a normal distribution. This is because the normalization of inputs can aid in enhancing the model's performance and generalization capabilities.

A critical component in realizing this normalization is the usage of **Scalers**. Scalers are deployed in deep learning to standardize the values of input features, ensuring that they operate on a similar scale. This is a fundamental requirement in the optimization of the learning process. Through standardization, the convergence of training is facilitated, and all features are enabled to contribute evenly and proportionately to the model's predictions.

In the context of our study, we opted to use the **Power-Transformer**, specifically the **Yeo-Johnson** transformation, as the scaler. Specifically the decision of Yeo-Johnson over another type of power transformer relies on the fact that the first one can perform only on positive and negative values while others on positive only. The decision to employ this method was largely informed by the skewed nature of our dataset. Yeo-Johnson transformation is a flexible technique that can manage

both positive and negative values and is effective in normalizing data and reducing skewness.

When comparing the Yeo-Johnson transformation to other potential scalers that are not Power transformer, the advantages become clear. The **Standard Scaler**, for example, operates under the assumption that data are normally distributed. However, it fails to effectively handle data with significant skewness and high ranges, which results in unsatisfactory performance in such cases. The **MinMax** scaler, another commonly used method, is also found inadequate when it comes to handling skewed data. It is prone to information loss due to the transformation of data to a specific range, leading to a potential flattening of the features where the standard deviation may equate to zero.

In sum, for data that exhibits skewness, as is the case with our dataset, the Yeo-Johnson transformation scaler presents as an optimal choice to ensure effective standardization and normalization, thereby enabling more accurate modeling and improved predictive performance.

Below there will be plots that shows how different scalers operate on the factors of **Mean**, **Std vector**, **Ranges** and **Skewness**.

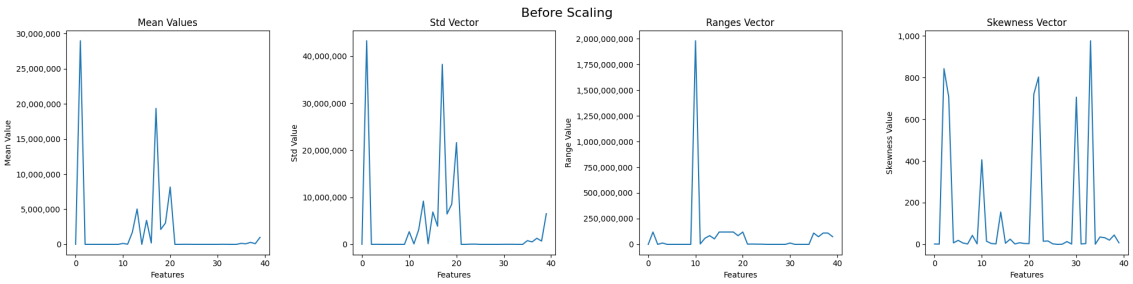


Figure 4.2: *Before Scaling*

As illustrated in the Figure 4.2, it is notable that the mean vector for certain features is exceptionally high. Simultaneously, the standard deviation vector exhibits a similar trend of elevated values. Furthermore, the range vector and the skewness vector also appear to follow a comparable pattern, reinforcing the highly skewed nature of these specific features.

Applying the MinMax scaler Figure 4.3 has helped in resolving certain issues such as normalizing the range vector and somewhat mitigating the excessive values in the mean vector. However, this approach has also led to a significant decrease in the standard deviation vector, with values now falling below 0.5 for all features. This reduction could potentially result in the loss of valuable information due to the features being overly flattened. Moreover, the MinMax scaler has made no substantial improvements in addressing the skewness issue, with values remaining near to or exceeding 200.

The implications of such skewed and flattened feature distribution could be detrimental for the performance of our machine learning model. Firstly, models trained

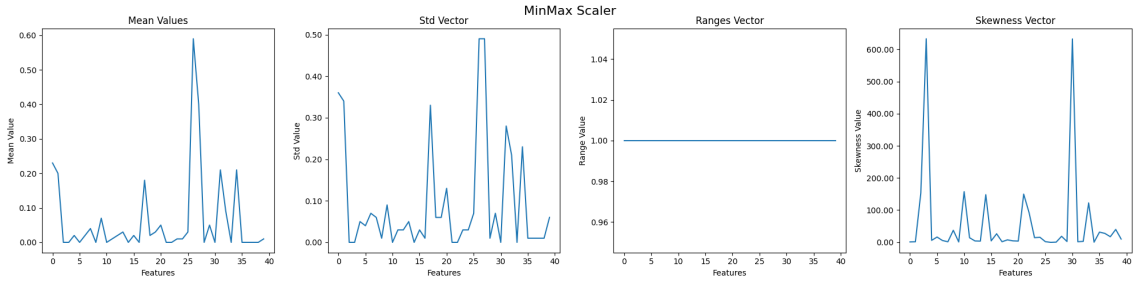


Figure 4.3: *MinMax Scaler*

on skewed data can be biased and result in inaccurate predictions. Secondly, low standard deviation, resulting from the data being overly flattened, could cause loss of unique information about each feature, leading to decreased model sensitivity to variations in those features. This could potentially reduce the model's ability to accurately learn and generalize from the data, thereby negatively impacting its overall predictive performance.

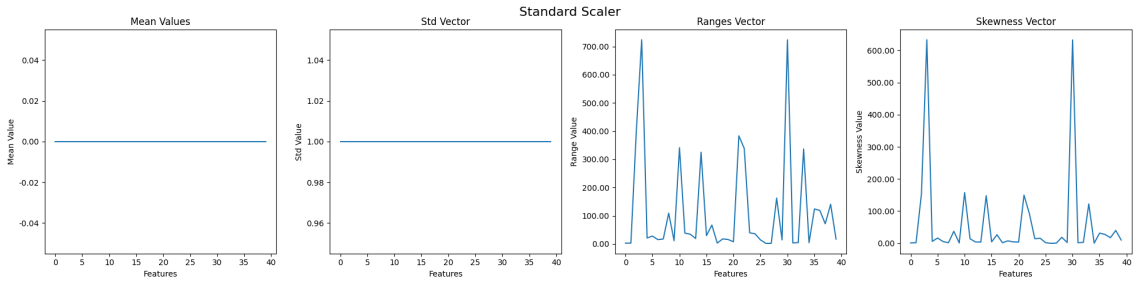


Figure 4.4: *Standard Scaler*

The application of the Standard Scaler algorithm Figure 4.4 demonstrates a superior ability to manage the Mean Vector and Standard Deviation when compared to MinMax. However, despite its strengths, this technique falls short in effectively addressing the range vector; the values still remain strikingly high. Additionally, the skewness of the data persists in its exceedingly high state.

This is largely due to the fact that the Standard Scaler assumes the data to be normally distributed, without considering the skewness. This assumption does not align well with the reality of our data, making the Standard Scaler unsuitable in this context.

The most recent Figure 4.5 demonstrates the superior performance of the Power Transformer, specifically the Yeo-Johnson algorithm, compared to other methods under consideration. It showcases a remarkably consistent mean vector hovering around the zero mark and an equally standardized standard deviation vector, fixed at 1 across all features. This demonstrates a successful standardization of the data, crucial for effective machine learning model training.

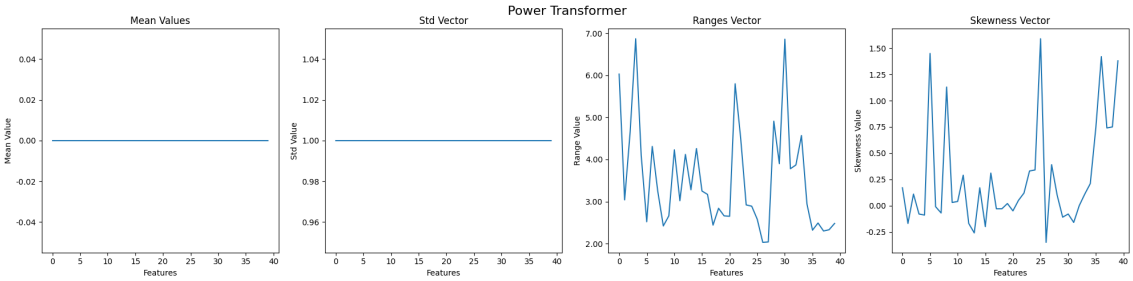


Figure 4.5: *Power Transformer*

Moreover, the range vectors, initially spanning widely varying scales, are now successfully condensed to a similar scale between 2 and 7 across all features, indicating a harmonized data spread. The skewness vectors also show dramatic improvements; their initial high values nearing or even surpassing 200 are now greatly reduced, varying between 0 and 2.

This accomplishment is particularly significant considering the initial state of the data, highlighting the efficacy of the Yeo-Johnson algorithm. It underscores the value of this algorithm in transforming data, thereby ensuring that the data inputs into our model are optimally preprocessed for enhanced model performance and generalization.

This achievement is not merely technical, but it also carries important implications for our subsequent analyses. By creating a data set that maintains the diversity of the original information while ensuring that the scales and distributions are balanced, we lay a solid foundation for the upcoming modeling and prediction tasks.

Chapter 5

Experiments

In this chapter, we discuss briefly the experiments and testing conducted in the scope of the diploma thesis. In addition, difficulties and challenges that we faced during the project are mentioned and of course what are the conclusions and what knowledge was earned by the results of these experiments.

5.1 Experimental procedure

In this series of experiments, the focus was on the application of various autoencoder (AE) configurations for anomaly detection on the CICIDS 2017 dataset. The baseline autoencoders, also known as vanilla AEs, were first implemented. This fundamental setup was crucial in establishing the base performance and set the ground for deeper exploration.

Subsequently, Deep Autoencoders (Deep AEs) were introduced, expanding the architecture in terms of layers and complexity, aiming for increased performance, greater capacity and better anomaly detection. During this stage, hyperparameter tuning played an important role in enhancing the model's capability and effectiveness. Specific parameters such as number of hidden layers, size of layers, learning rate, batch normalization, dropout rate and the number of epochs were fine-tuned using RandomSearch to optimize the model's performance.

The results were then compared to those of shallow techniques, such as Support Vector Machines (SVMs), and the initial baseline model. This comparison offered insights into the extent of improvement and the relative effectiveness of these deep learning techniques for the task at hand.

The research was then expanded to include AE variants, specifically the Denoising Autoencoders (DAEs) and β-Variational Autoencoders (β-VAEs). DAEs aim to reconstruct the original input from a noisy version, which forces DAE to learn meaningful properties in the latent space ignoring noise. On the other hand, β-VAEs bring the aspect of latent variable modeling to traditional AEs, providing a probabilistic interpretation of the encoding-decoding process and offering additional capabilities to the model.

5.1.1 Training Process of Autoencoders

The training procedure of the autoencoder for anomaly detection involves a series of steps. Initially, we need to partition the dataset into 'normal' and 'attacks' categories. This segregation is important as it forms the basis for training our model exclusively on the 'normal' data.

Once we've separated our dataset into these two broad classes, the next step is to further split the 'normal' data into a training set, a validation set, and a test set. This is a crucial step that allows us to effectively train our model and to evaluate its performance on unseen data.

Following this, we integrate the 'attacks' data with the previously created test set. This amalgamation results in a new test set that comprises both 'normal' and 'attack' data. The presence of 'attack' data points in the test set enables us to evaluate the performance of our model in accurately detecting anomalous instances.

The model training process is carried out exclusively on the 'normal' data. The fundamental premise here is that the autoencoder learns to reconstruct normal data accurately and thus, any deviation from this norm is flagged as an anomaly.

Upon completion of the training, we pass the training data through the model once again to compute the reconstruction error. This error gives us an estimate of how much the reconstructed data deviates from the original input.

An essential aspect of this process is the determination of an optimal threshold for flagging an instance as an anomaly based on its reconstruction error. The distribution of reconstruction errors provides us a guideline for setting this threshold. Typically, between 90_{th}, 95_{th} or 99_{th} percentile of the distribution of reconstruction errors is the choice of the threshold. However, based on experience and the specificities of the problem at hand, we've found that the 90th percentile often strikes the right balance and will be the chosen threshold for the remainder of our trials.

5.1.2 Evaluation Metrics

In evaluating the performance of anomaly detection models, a number of common metrics can be utilized, including accuracy, F1 score, precision, recall, ROC-AUC, Attacks Detection Rate and Normal Detection Rate. These metrics provide a comprehensive picture of the model's performance and help in understanding the trade-offs involved.

Definitions:

- True Positives (TP): attack examples classified as attack
- False Positives (FP): benign examples classified as attack
- True Negatives (TN): benign examples classified as benign

- False Negatives(FN): attack examples classified as benign

Metrics:

- **Accuracy**: Accuracy represents the overall correctness of the model's predictions. Mathematically, it is the ratio of correct predictions (both true positives and true negatives) to the total number of predictions. In the context of anomaly detection, accuracy can be misleading if the dataset is imbalanced, as the model may achieve high accuracy by simply classifying everything as the majority class.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{33}$$

where TP, FP, FN,and TN stand for True Positives, False Positives, False Negatives, and True Negatives respectively.

- **weighted Precision**: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. In the context of anomaly detection, a high precision means that the model rarely labels a normal point as an anomaly, but may miss many actual anomalies.

$$\text{Weighted Precision} = \frac{\sum_{i=0}^{n}(W_i * TP_i)}{\sum_{i=0}^{n}(W_i * (TP_i + FP_i))} \tag{34}$$

Where W_i is the weight of class i

- **Weighted Recall**: Recall (also known as sensitivity or true positive rate) is the ratio of correctly predicted positive observations to all actual positive observations. In terms of anomaly detection, a high recall means that the model catches a large portion of anomalies, but may also incorrectly flag many normal points as anomalies.

$$\text{Weighted Precision} = \frac{\sum(W_i * TP_i)}{\sum(W_i * (TP_i + FN_i)} \tag{35}$$

Where W_i is the weight of class i

- **F1 Score**: The F1 Score is a measure of a model's precision and recall. The F1 Score is the harmonic mean of precision and recall, and it reaches its best value at 1 (perfect precision and recall) and worst at 0. It is a good way to summarize the evaluation of the model when dealing with imbalanced classes. Weighted F1 score adjusts the F1 scores based on the number of samples from each class. It is defined as

$$\text{Weighted F1 score} = 2 * \frac{\text{weighted Precision} * \text{weighted Recall}}{\text{weighted Precision} + \text{weighted Recall}} \tag{36}$$

- **ROC-AUC**: The Receiver Operating Characteristic Area Under the Curve (ROC-AUC) score represents the probability that a random positive (anomaly) example is placed before a random negative (normal) example. The AUC provides a robust measure that can handle imbalanced classes. An AUC of 1.0 indicates a perfect model, whereas an AUC of 0.5 suggests a model no better than random chance.

- **Attacks Detection Rate**: This metric reflects the model's ability to accurately detect attack examples, regardless of their specific class(DDoS or botnet). It is particularly crucial in contexts where all anomalies (attacks) are equally significant, irrespective of their individual types.

- **Normal Detection Rate**: Conversely, the Normal Detection Rate evaluates the model's proficiency in identifying benign or non-anomalous instances. A high Normal Detection Rate ensures that the model minimizes false alarms, thereby reducing potential unnecessary interventions.

Each of these metrics offers a different perspective on the performance of the model, and it's important to consider all of them when assessing the efficiency of an anomaly detection system. Furthermore, the choice of metric may depend on the specific cost associated with different types of errors in the application domain. For example, in some contexts, missing an anomaly (low recall) may be more costly than raising a false alarm (low precision), and the evaluation metric should reflect that.

5.1.3 Visualization tools and Techniques

To effectively illustrate the results of the experiments and assist in understanding the performance of our anomaly detection model, we'll employ several visualization tools and techniques. These will aid in both the presentation and interpretation of our experimental results.

Reconstruction loss: One of the primary visualization tools that we will use is the Kernel Density Estimation (KDE) plot. This specific plot will be used for visualizing the distribution of the reconstruction loss values. KDE fitting a histogram with Gaussian approximations (also known as kernels) for each bin provides us with a smooth and continuous estimate of the distribution. This makes KDE an incredibly powerful tool, providing a more informative view of the data compared to a simple histogram.

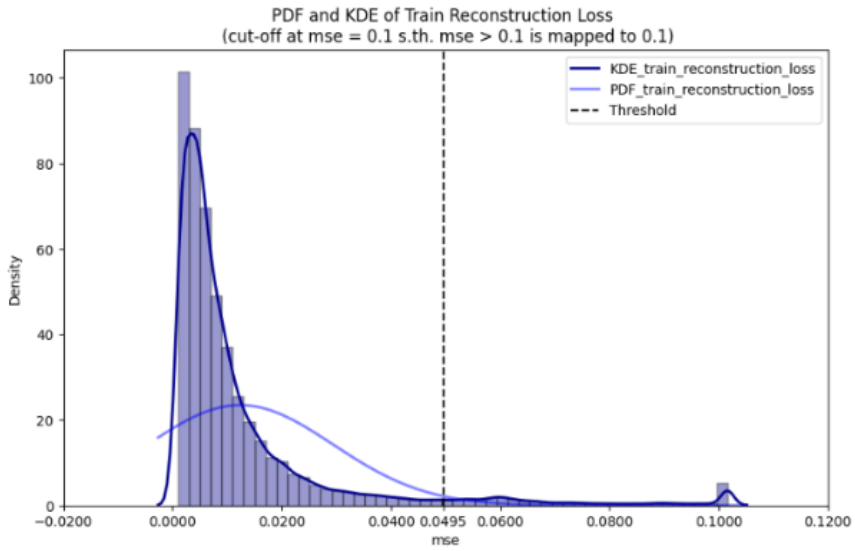


Figure 5.1: *Train set Reconstruction Loss and threshold*

An important feature integrated into these plots is a vertical threshold line. This line serves as a visual aid, clearly indicating which instances our model considers anomalous based on their reconstruction loss value. Instances with a reconstruction loss value beyond(in the right of) this threshold line are considered anomalies, simplifying the identification process.

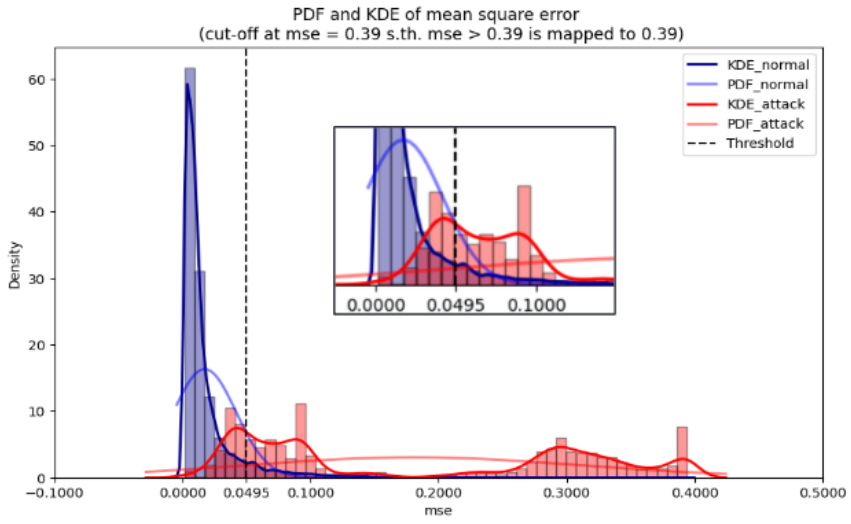


Figure 5.2: *Test set Reconstruction Loss*

To ensure that our visualization doesn't get excessively influenced by extreme outliers, a strategic cutoff value at the 95_{th} percentile of the test set reconstruction loss values is implemented and at the 99_{th} percentile for the training set reconstruction error. This approach ensures that our focus remains on the most relevant part of the distribution, providing a clear view without distortion from extreme values.

Latent Space:

In order to provide a thorough and clear interpretation of the experimental results, the utilization of effective visualization tools is paramount. Among these tools, one of the most insightful methods is the visualization of the model's latent space in two dimensions.

The concept of "latent space" is integral to autoencoder models. The latent space refers to the lower-dimensional representation of the input data that an autoencoder model learns to map. In the context of zero-day attacks, during the learning phase, the model predominantly encounters benign data, while attacks materialize as anomalies exclusively during the testing phase.

To project the multi-dimensional latent vectors onto a 2D plane, we employ Principal Component Analysis (PCA). PCA is a widely used technique for dimensionality reduction that preserves the variability in the data and helps to identify patterns and structures within it.

By visualizing the latent space, we can gain insights into the internal representations that the model has learned. It provides an intuitive way to explore how the model differentiates between various types of input data. The distribution and arrangement of points in this 2D space are telling of the model's underlying data comprehension.

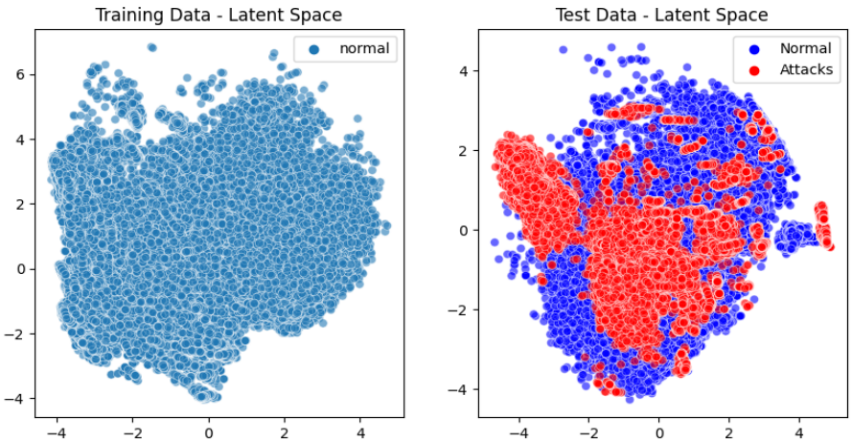


Figure 5.3: *Plot Latent space*

Latent space visualizations like this one in Figure 5.3 are a powerful tool for identifying clusters and outliers, especially in the realm of anomaly detection. Typically, we would expect benign data points to form a dense cluster in the latent space, depicted by blue points in our graph. However, it's critical to underscore that distinct separation in the latent space, while desirable, is not a prerequisite for a well-performing anomaly detection model.

Anomalous or attack data points, represented by red dots, might appear close to or even overlapping with the normal data in the latent space. This occurrence is

especially likely when their statistical properties resemble those of the normal data. Despite this overlap, these attack points can still be effectively distinguished based on the higher reconstruction errors generated by our models.

5.2 Baseline Autoencoder

In this study, the baseline model is a Vanilla AutoEncoder. Its architecture includes one hidden layer per Encoder and Decoder, as well as a latent space layer, commonly referred to as the "code". Following a series of experiments, the final model configuration is established as follows:

- Input Layer: 40 units
- Encoder's Hidden Layer: 25 units
- Latent Space (Code): 15 units
- Decoder's Hidden Layer: 25 units
- Output Layer: 40 units

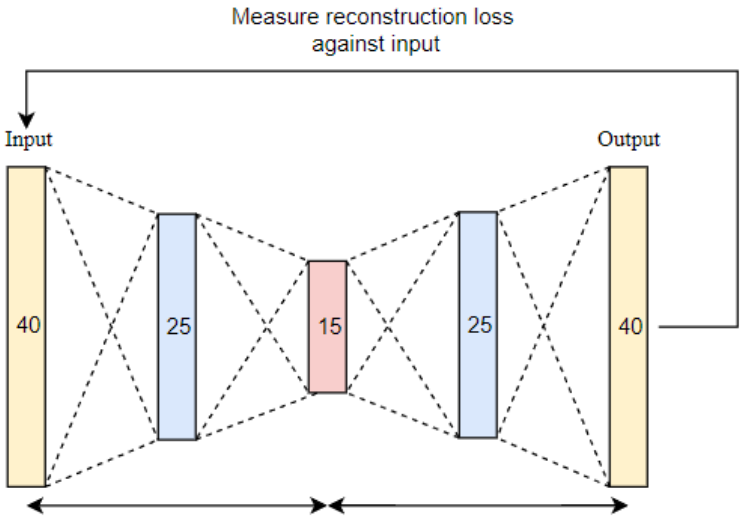


Figure 5.4: *Vanilla AutoEncoder Architecture*

The activation function selected for this model is the Rectified Linear Unit (ReLU), while the Mean Squared Error (MSE) serves as the reconstruction loss function. The hyperparameters of the model are set as:

- Batch size: 1024
- Epochs: Initially 100 (though early stopping is employed during training)
- Optimizer: RMSprop
- Output Layer's Activation function: Linear
- Other Layers' Activation function: ReLU
- Learning Rate: 0.001

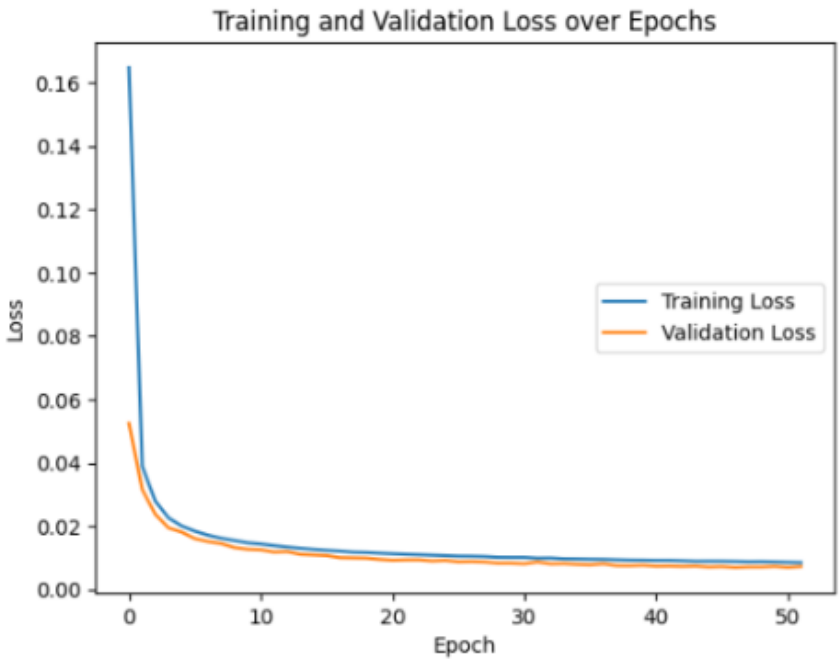


Figure 5.5: *Vanilla AE learning curves*

Instances with a reconstruction error that surpasses a predetermined threshold are classified as attack instances, with the remaining instances being considered normal. This threshold is determined during training, and it is subsequently applied during the evaluation on a previously unseen test set that comprises a balanced collection of normal and attack data.

The evaluation includes was conducted using reconstruction loss at the 90th and 95th percentile as thresholds. After reviewing these, the decision is made to proceed using only the 90th percentile as the threshold for further tests Figure 5.6.

In terms of model performance comparison, our Vanilla AutoEncoder appears to outperform the One-Class SVM Figure 5.7. This points towards a positive direction for further exploration. However, it's noteworthy that the Vanilla AutoEncoder exhibits a low attack detection rate, suggesting a potential need for more sophisticated AutoEncoder models.

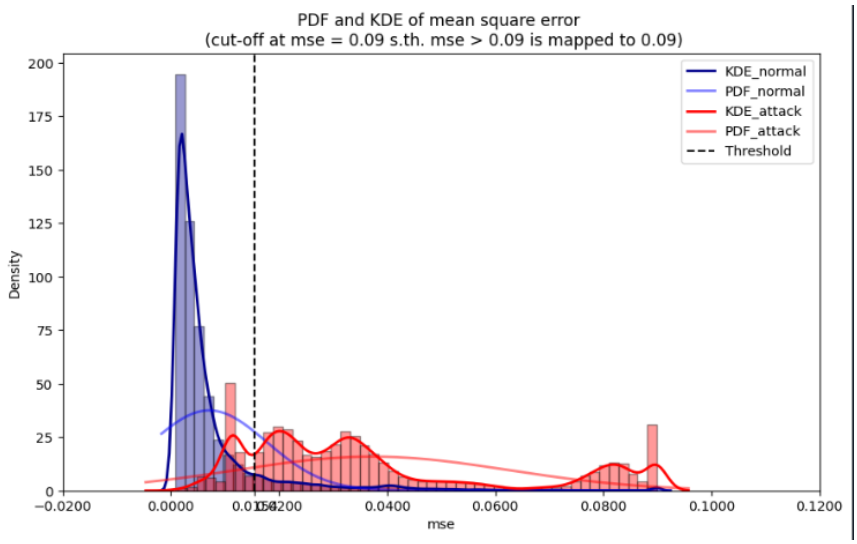


Figure 5.6: *Vanilla AE test set reconstruction loss*

Metric	Value
Threshold	0.0267
Accuracy	0.751
W.Avg Precision	0.803
W.Avg Recall	0.755
W.Avg F1-score	0.740
ROC AUC	0.947
Attacks Detection Rate	0.564
Normals Detection Rate	0.953

Table 5.1: *Performance Metrics of the Model*

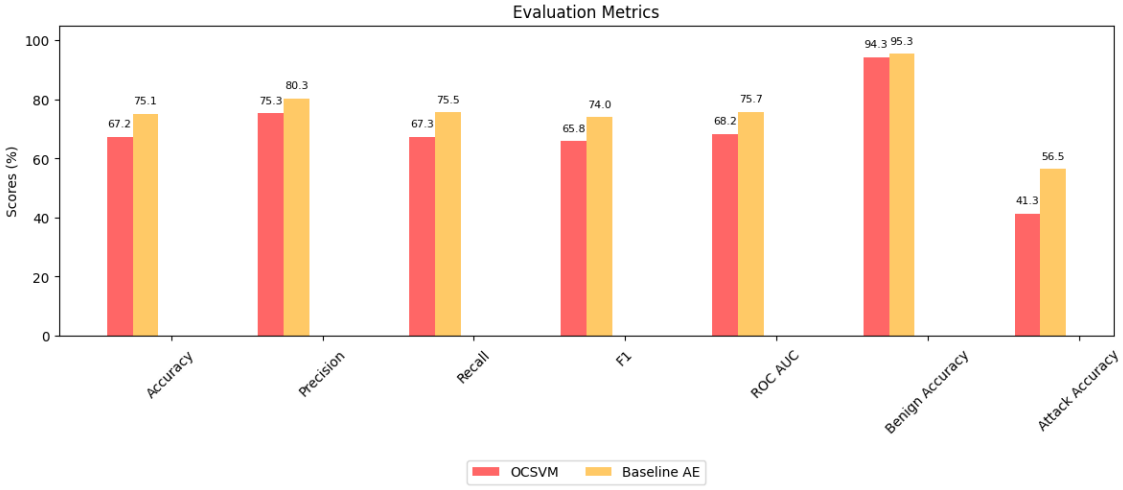


Figure 5.7: *Vanilla AutoEncoder against SVM*

As we can see in the Figure 5.8 among the attack Categories we have

- **Best Accuracy:** DDoS, Portscan, Infiltration, DoS-slowhttptest, DoS-Slowloris, Heartbleed
- **Hard to Detect:** Portscan, Botnet, Web Attack-Brute Force, Web Attack-XSS, Web Attack-Sql Injection, SSH-Patator, FTP-Patator, DoS-GoldenEye

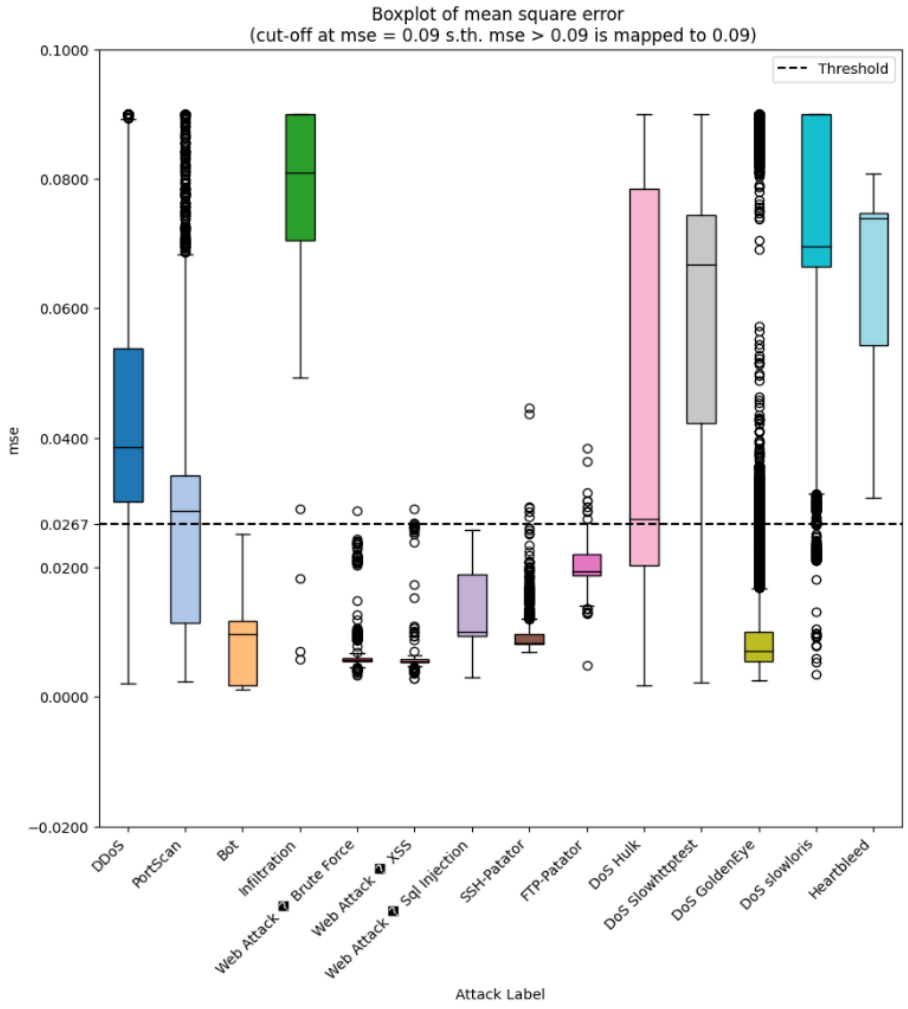


Figure 5.8: *Vanilla AutoEncoder Box plot of MSE*

5.3 Deep AutoEncoder

In the current implementation, we have included more layers in contrast to a traditional Vanilla AutoEncoder (AE). We've utilized the RandomSearch algorithm for tuning hyperparameters, which entails a series of trials during which we've chosen to use validation loss as the objective to choose the best hyperparameters for our case. We've selected this measure because it can serve as a strong indicator of model performance.

However, it's important to note that a low validation loss does not guarantee superior performance, as there's a risk that the model may simply learn to replicate the input to the output without successfully capturing the underlying representation of benign traffic.

As a result, we introduced an additional evaluation step where the top 10 models, selected based on the validation loss criterion, were tested on the test set. This was done to determine the model with the best discriminatory ability. Interestingly, the best performing model was not the one ranked first based on the validation loss criterion.

Category	**Parameters**
Architecture	#Layers, #nodes per layer, Latent Space dimension, Batch Normalization
Regularization	Dropout, Dropout rate
Learning	Learning rate

Table 5.2: *Parameters explored by RandomSearch*

While the RandomSearch algorithm was utilized for hyperparameter tuning, some parameters were deliberately kept fixed to decrease the search space. These parameters, chosen based on practical experience, remained constant throughout the process. The list of fixed hyperparameters along with their chosen settings are as follows:

- Reconstruction Loss: MSE
- Activation Function: ReLu
- Output layer Activation: Linear
- Optimizer: RMSprop

The output of the RandomSearch algorithm yielded an optimal architecture for our Deep AutoEncoder model, detailed as follows:

- Input Layer: 40 units.
- Encoder's Hidden Layers: 64 units, 64 units, and 14 units, respectively.

- Latent Space (Code): 4 units.
- Decoder's Hidden Layers: 14 units, 64 units, and 64 units, respectively.
- Output Layer: 40 units.

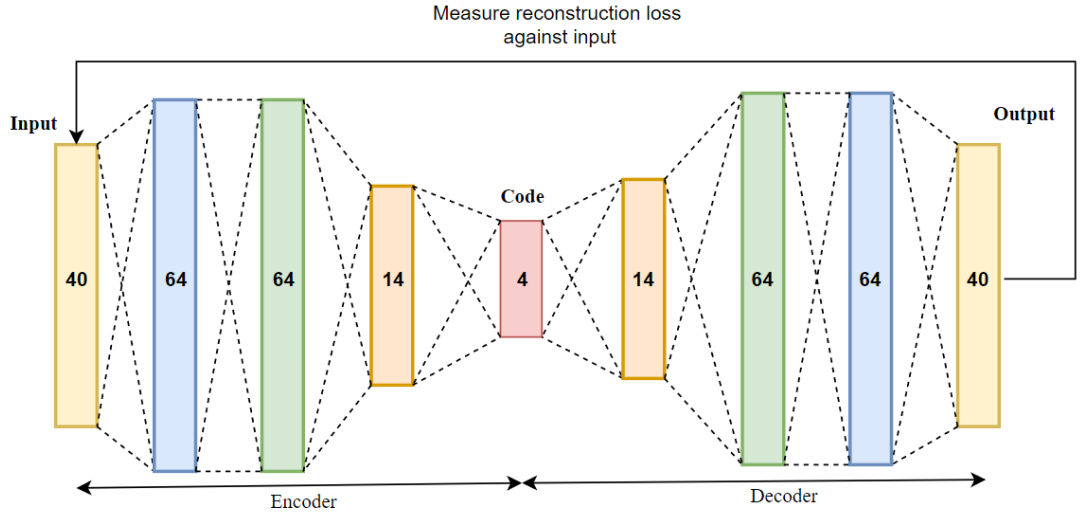


Figure 5.9: *Deep AutoEncoder Architecture*

The model's hyperparameters are configured as follows:

- Batch size: Set to 1024. This offers a good balance between computational efficiency and the granularity of the gradient estimation.
- Epochs: Initially set to 100, however, early stopping is employed during training. Early stopping helps to avoid overfitting by terminating the training process if the model's performance on the validation set does not improve after a number of epochs.
- Batch Normalization: Helps to accelerate training and provide some regularization effect, thus reducing generalization error.
- Early stopping: Incorporated to monitor validation loss and stop training when validation loss ceases to decrease, thus preventing overfitting.

Based on our observations from the latent space and test set's reconstruction loss plots, our model has demonstrated significant potential in anomaly detection. Although there is a tight representation in the latent space, a degree of overlap is also observed. This overlap signifies that our model has learned shared representations between normal and anomalous traffic. Nevertheless, despite this overlap, our model can discriminate effectively between normal and anomalous traffic when the reconstruction loss is considered.

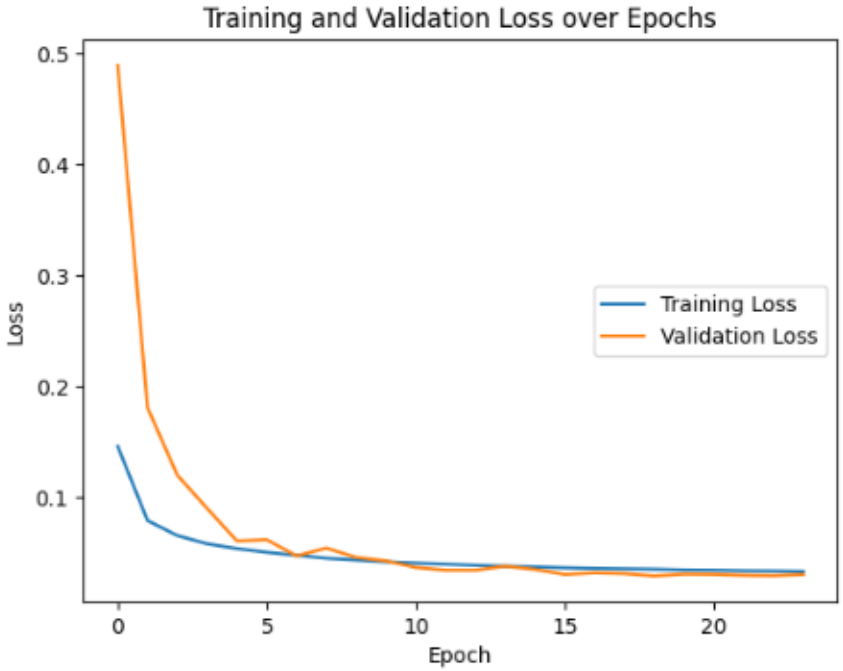


Figure 5.10: *Deep AE learning curves*

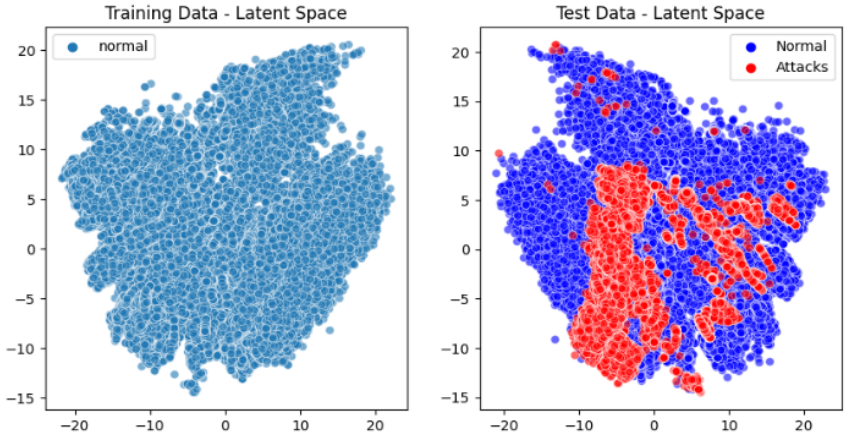


Figure 5.11: *Deep AE latent space*

Notably, the model's threshold is set at 90th percentile of train reconstruction loss and value is $thr = 0.0613$, which perfectly separates the distributions of the normal and anomalous traffic in the reconstruction loss plot. Such optimal threshold has resulted in excellent detection rates. Specifically, the model achieved an accuracy of 94.87%, a weighted average precision of 94.70%, and a weighted average recall of 95.31%. The weighted average F1-score, which gives a balanced measure of the model's precision and recall, is at a high of 95.19%. Furthermore, the model boasts an ROC AUC score of 94.75%, indicating its robustness in classifying benign and attack traffic.

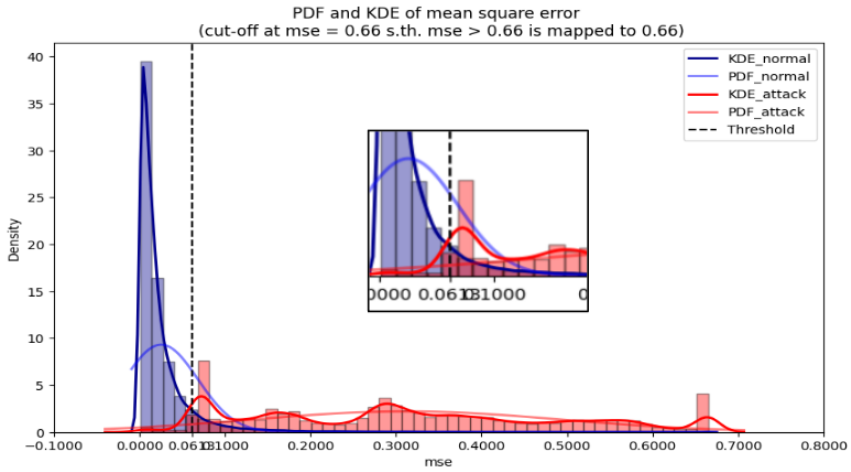


Figure 5.12: *Deep AE test set reconstruction loss*

In terms of detecting anomalies, our model outperforms the baseline AE and OCSVM by a substantial margin. It achieved an impressive attack detection rate of 98.17%. Simultaneously, the model managed to maintain a high normal detection rate of 92.23%, signifying a minimal trade-off between attack detection and benign traffic recognition.

In conclusion, our Deep AutoEncoder model, with its current configuration and settings, has demonstrated substantial improvements over the baseline models, striking a remarkable balance between accurately detecting anomalous traffic and minimizing the false-positive rate for benign traffic

Metric	Value
Threshold	0.0613
Accuracy	0.948
W.Avg Precision	0.947
W.Avg Recall	0.953
W.Avg F1-score	0.952
ROC AUC	0.948
Attacks Detection Rate	0.982
Normals Detection Rate	0.922

Table 5.3: *Performance Metrics of the Deep AutoEncoder using* 90_{th} *percentile as threshold*

As we can see in the Figure 5.14 among the attack Categories we have

- **Best Accuracy:** DDoS, Portscan, Infiltration, SSH-patator, FTP-patator, DoS Hulk, DoS slowhttptest, DoS GoldenEye, DoS Slowloris, Heartbleed

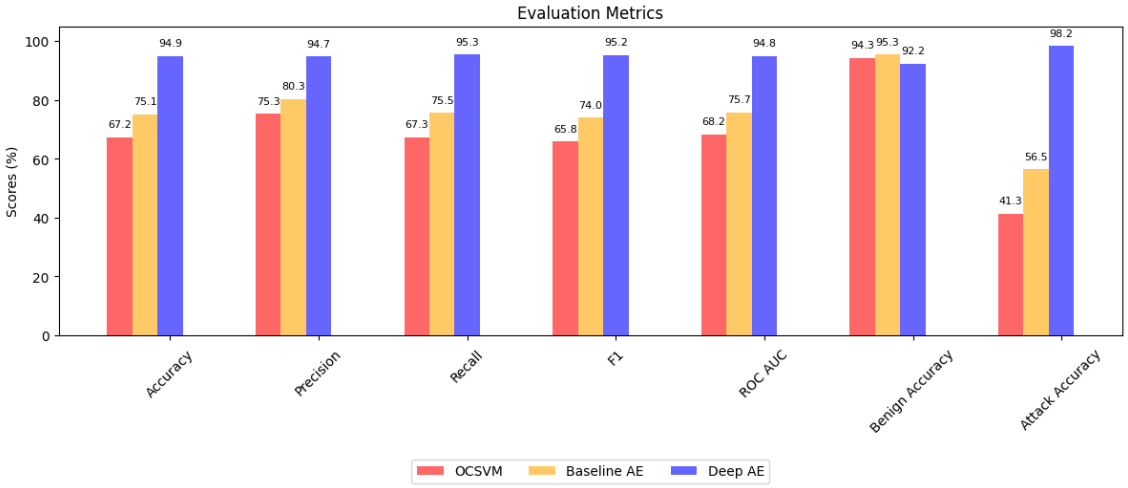


Figure 5.13: *Deep AutoEncoder against Baseline & SVM*

- **Hard to Detect:** Botnet, Web Attack-Brute Force, Web Attack-XSS, Web Attack-Sql Injection

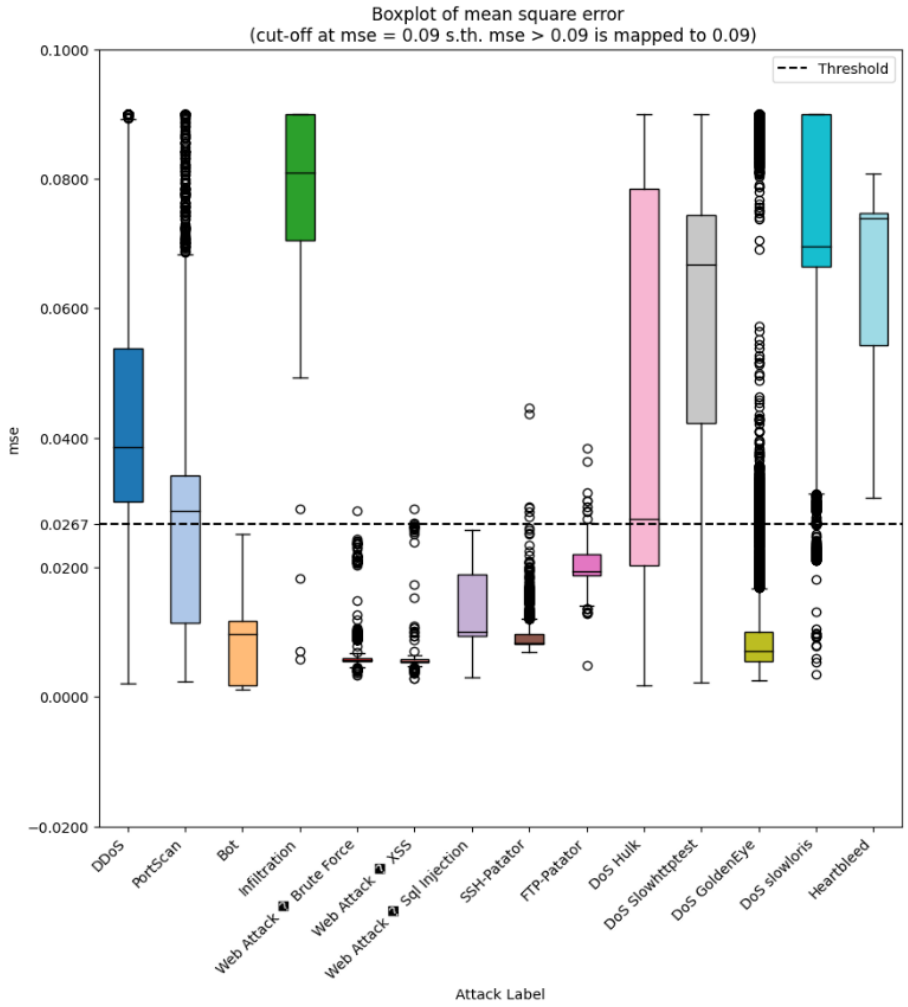


Figure 5.14: *Deep AutoEncoder Box plot of MSE*

5.4 AE Variants

5.4.1 Denoising AEs

The aim of this experiment is to explore the potential benefits of noise on the learning capacity of Autoencoders (AEs). Specifically, we focus on Denoising Autoencoders, a variant of the standard autoencoder that's designed to improve model robustness and mitigate overfitting.

One common issue that typical AutoEncoders face is Overfitting. Overfitting occurs when the model becomes overly complex in relation to the data at hand. This complexity can lead the autoencoder to simply reproduce the **identity function**, instead of capturing the essential, underlying patterns in the data. A potential countermeasure to this issue is the introduction of noise into the model. The noise can guide the model to pay more attention to the significant features of the data, rather than simply reproducing the input. In this experiment, we are assessing the

effect of different levels of Gaussian noise on our previously configured and tested Deep AutoEncoder model. Gaussian noise is a common choice in such situations due to its properties that closely align with natural processes.

While adding noise can indeed help a model learn more significant features, adding excessive noise may confound the model and lead to deteriorated performance. Therefore, it's essential to experiment with the degree of noise we introduce to the input data. However, it's equally crucial to adjust the model's hyperparameters for each level of noise. This is because varying amounts of noise may necessitate different hyperparameters for optimal performance. As a result, we employed RandomSearch for each noise level we experimented with, adjusting hyperparameters accordingly.

- Input Layer: 40 units
- Encoder's Hidden Layer: 34 units, 22 units, and 10 units, respectively
- Latent Space (Code): 6 units
- Decoder's Hidden Layer: 10 units, 22 units, 34 units, respectively
- Output Layer: 40 units
- Batch Normalization (Training)
- Learning rate: 0,000615
- Dropout rate: 0.4
- Noise-factor: 0.2
- Activation function: ReLU (rectified linear unit)
- Reconstruction Loss: MSE (Mean Squared Error)

We performed experiments for various noise factors, namely $Noise\ factor = 0.1, 0.2, 0.3, 0.4$. The best results were observed for $Noise\ factor = 0.2$. The results of these experiments are outlined below in the Figure 5.16. Moving forward, our exploration will primarily focus on the most effective Denoising Autoencoder, and we'll draw comparisons with the results of previous experiments.

Upon analyzing the test set reconstruction loss plot in Figure 5.18 and the latent space plot below, some key observations can be made. Notably, the Denoising AutoEncoder exhibits a remarkable capability of maintaining a close reconstruction loss for benign data while also broadly spreading the attacks, an improvement over other AutoEncoders. This feature suggests an enhancement in its discriminative ability for anomalies, thereby making the detection of attacks more effective.

The peaks of attack losses are found to be further away from the threshold as compared to other AutoEncoders. This indicates a higher level of robustness, which

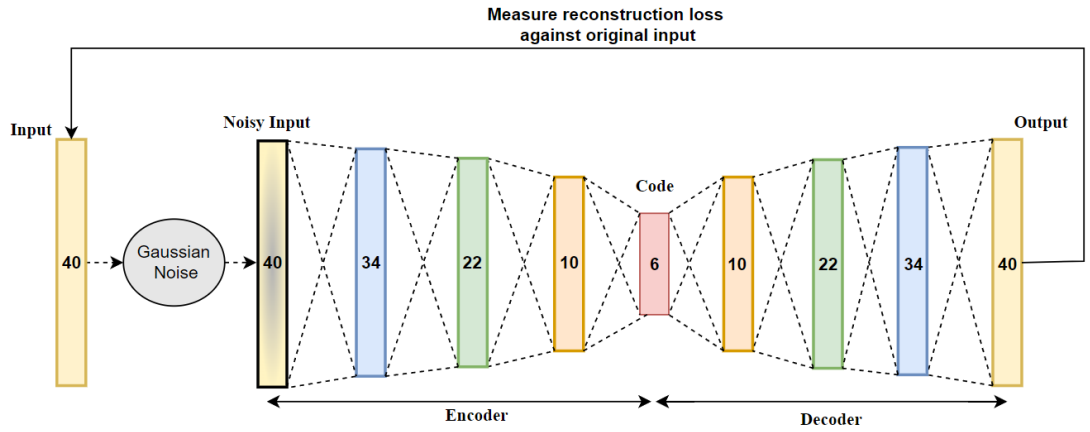


Figure 5.15: *Denoising AutoEncoder Architecture*

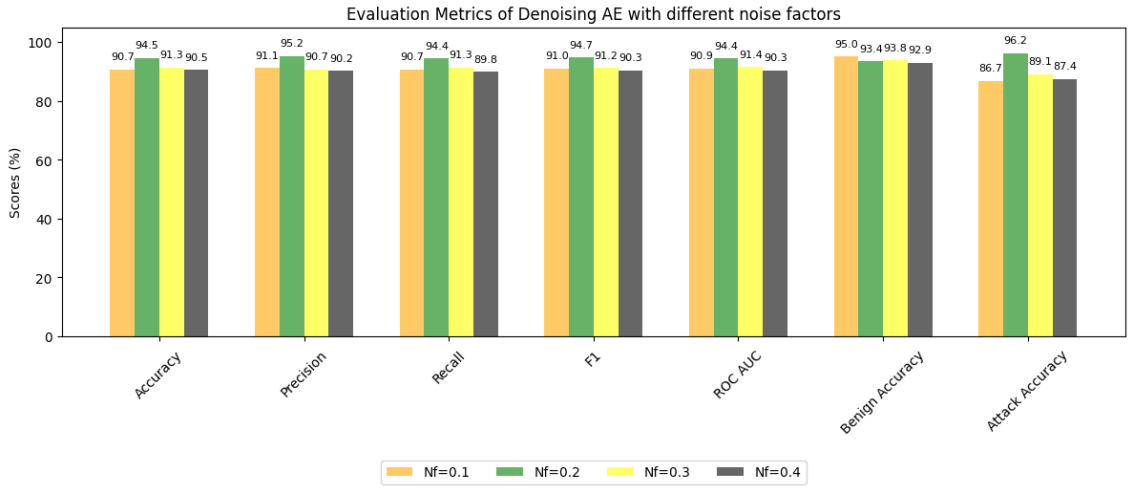


Figure 5.16: *Denoising AE evaluation metrics for different noise factors*

is an essential characteristic for a model aimed at anomaly detection. Inspecting the latent space in Figure 5.19, it is observed to have a tight representation. However, there is a significant overlap observed within this space. This overlap could be an indicator of the model's capacity to encode different classes of data in a similar latent space, which might affect the discrimination between benign and attack instances. Despite this, the benefits of a more robust and discriminative model, as indicated by the broader spread of attacks and close clustering of benign data, may outweigh the potential disadvantages of this overlap.

Given the metrics of the Denoising AutoEncoder (DAE) and the rest models, we can see that Deep AE and DAE models both perform similarly well, but there are some differences.

The Deep AutoEncoder is slightly more accurate, with an accuracy score of

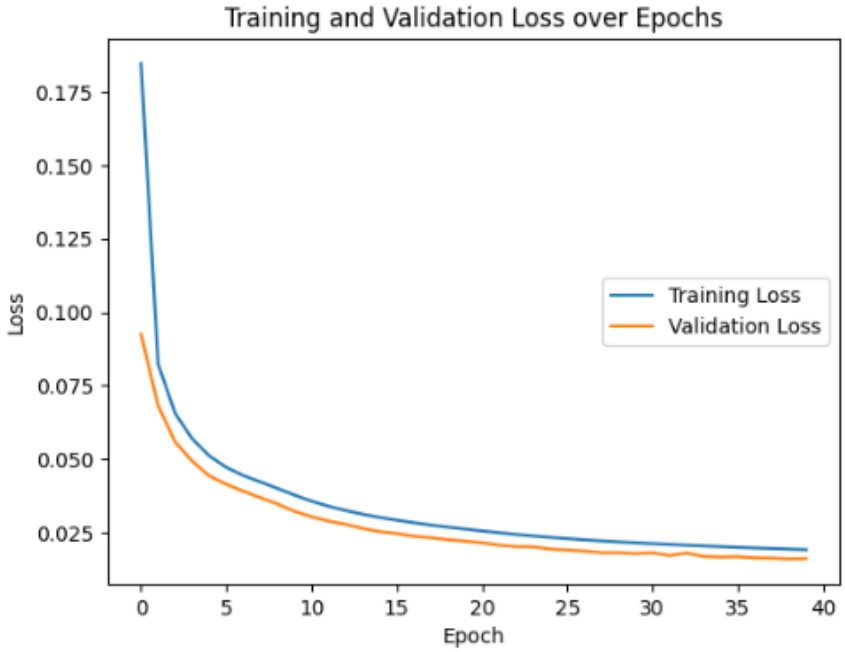


Figure 5.17: *Denoising AE learning curves*

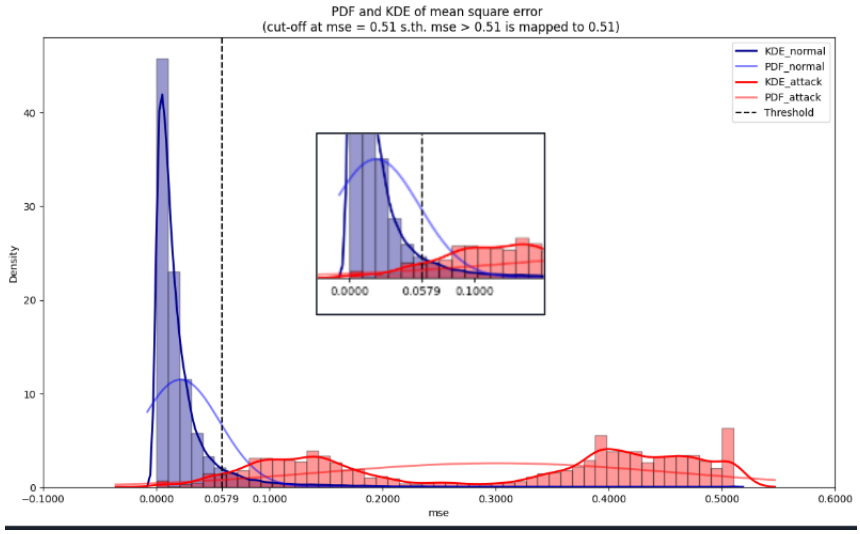


Figure 5.18: *Denoising AE test set reconstruction loss*

94.87% compared to the DAE's 94.47%. It also scores higher in terms of precision, recall, F1-score, and ROC AUC.

The Deep AutoEncoder is particularly better at detecting attacks, scoring 98.17% compared to the DAE's 96.15%. However, the DAE is slightly better at detecting normal instances, with a detection rate of 93.36% versus the Deep AutoEncoder's 92.23%.

In conclusion, the performance of Deep AutoEncoder (DeepAE) and Denoising AutoEncoder (DAE) are largely equivalent, demonstrating that the introduction of

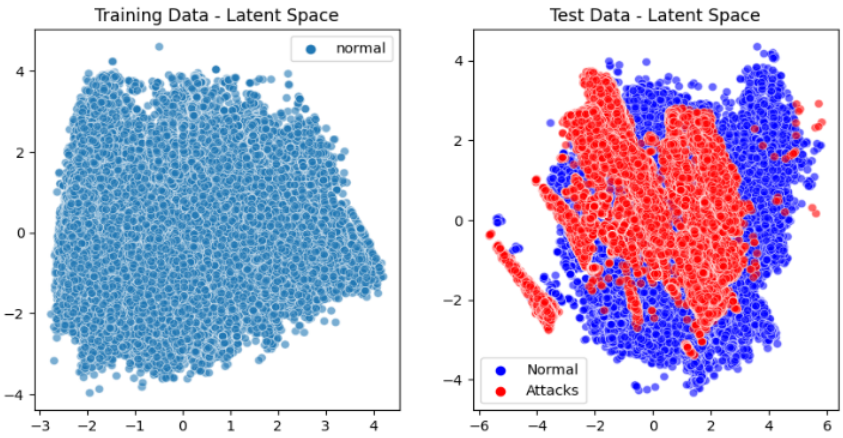


Figure 5.19: *Denoising AE latent space*

Metric	Value
Threshold	0.0579
Accuracy	0.945
W.Avg Precision	0.952
W.Avg Recall	0.944
W.Avg F1-score	0.947
ROC AUC	0.944
Attacks Detection Rate	0.962
Normals Detection Rate	0.934

Table 5.4: *Performance Metrics of the Denoising AE using* 90_{th} *percentile as threshold*

noise didn't impair the DAE's ability to achieve high performance. In fact, as we observed in the reconstruction loss discussion, anomaly peaks in DAE were further from the decision boundary (threshold), indicating that the model is more robust to minor changes. This suggests that using a DAE for anomaly detection is a promising approach and warrants further exploration.

As we can see in the Figure 5.21 among the attack Categories we have

- **Best Accuracy:** DDoS, Portscan, Infiltration, DoS Hulk, DoS slowhttptest, DoS GoldenEye, DoS Slowloris, Heartbleed
- **Hard to Detect:** Botnet, Web Attack-Brute Force, Web Attack-XSS, Web Attack-Sql Injection

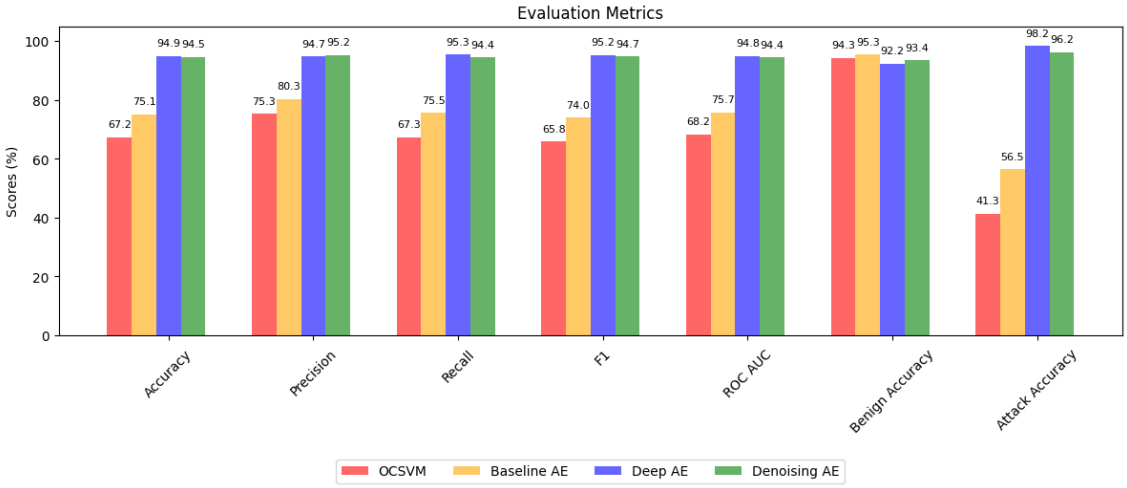


Figure 5.20: *Denoising AutoEncoder against other Models*

5.4.2 beta-Variational AEs

Variational AutoEncoders (VAEs) possess distinct properties and architectures compared to Deep AEs and Denoising AEs. Consequently, we need to determine the optimal configuration for VAEs from scratch. The main advantage of VAEs is their capability to capture meaningful patterns in their latent space due to their generative model nature. This could help avoid the model merely learning to replicate the input at the output, though it might also result in a decrease in reconstruction accuracy.

To mitigate this issue, we employ the parameter β to regulate the impact of KL-loss, thereby concentrating more on minimizing the reconstruction loss while retaining as many of the VAE's properties as possible. We'll include the parameter β, also known as the KL-coefficient, in the search space. This parameter modifies the original VAE into a beta-VAE. Notably, when β equals 1, we return to the original VAE

Similarly with the previous experiments we use RandomSearch algorithm to find the parameters that suits the most in our case but here another parameter is added to the search space the β, The optimal model configuration was identified as follows:

- Input Layer: 40 units
- Encoder Layers: 62 units, 19 units
- Latent Space: 16 units
- Dropout rate: 0.3
- Beta: 0.1
- Learning rate: 0.008623

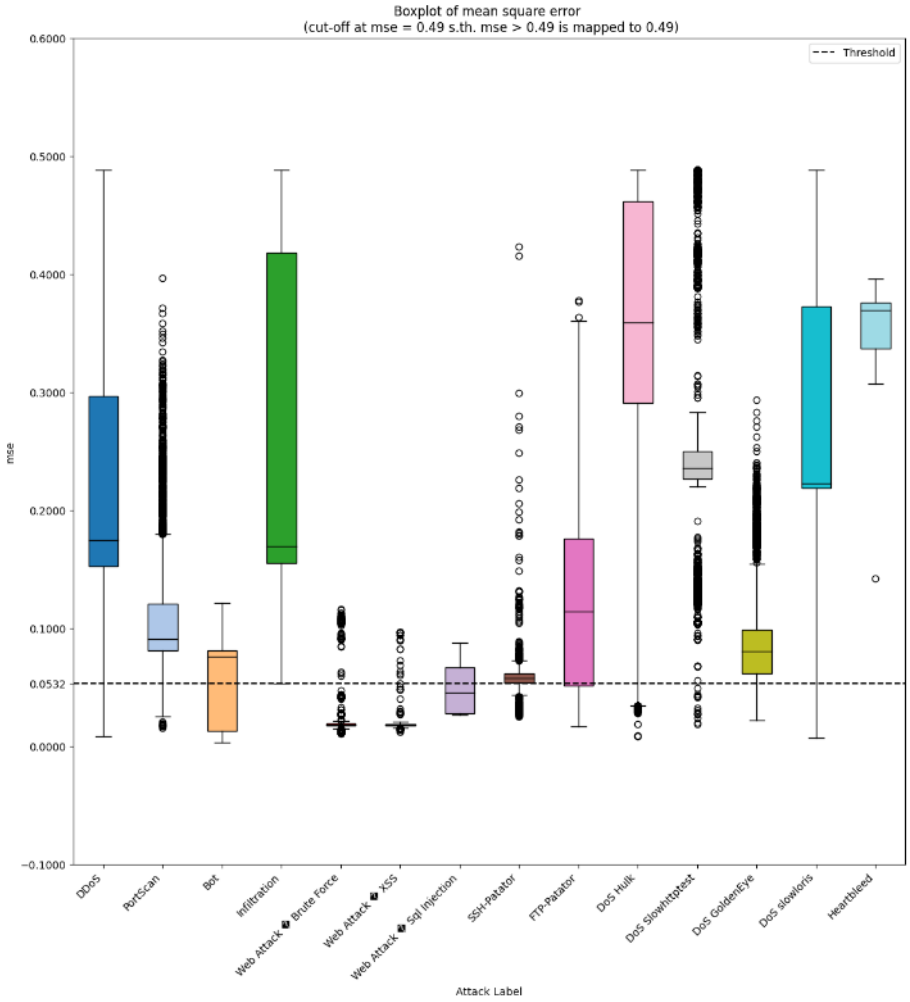


Figure 5.21: *Denoising AutoEncoder Box plot of MSE*

- Activation Function: ReLU
- Loss Function: MSE + KLD

Additional hyperparameters were established, as listed below. The learning curves of the losses can be viewed in Figure 5.23

- Batch size: 1024
- Epochs: Initially 100 (though early stopping is employed during training)
- Optimizer: RMSprop
- Output Layer's Activation function: Linear
- Other Layers' Activation function: ReLU

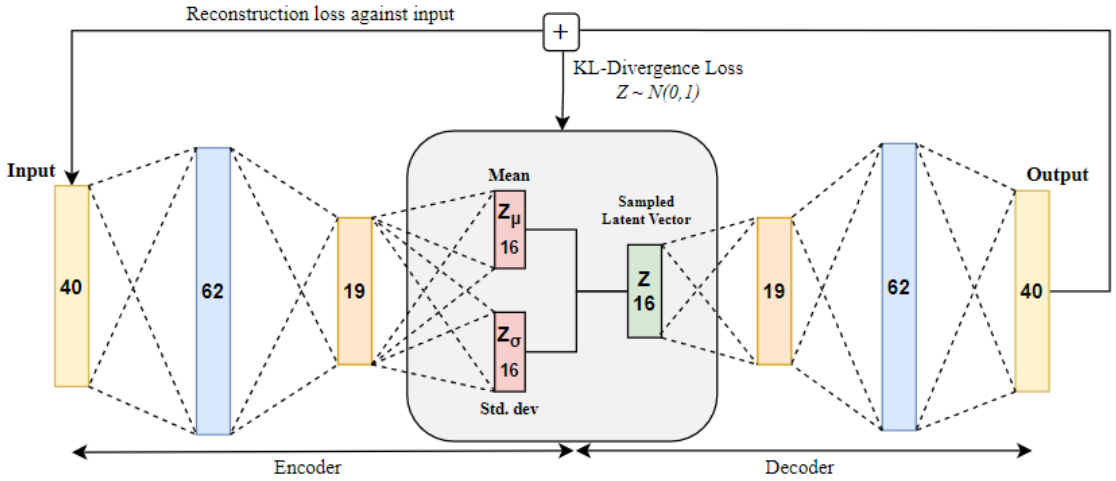


Figure 5.22: *β-VAE Architecture*

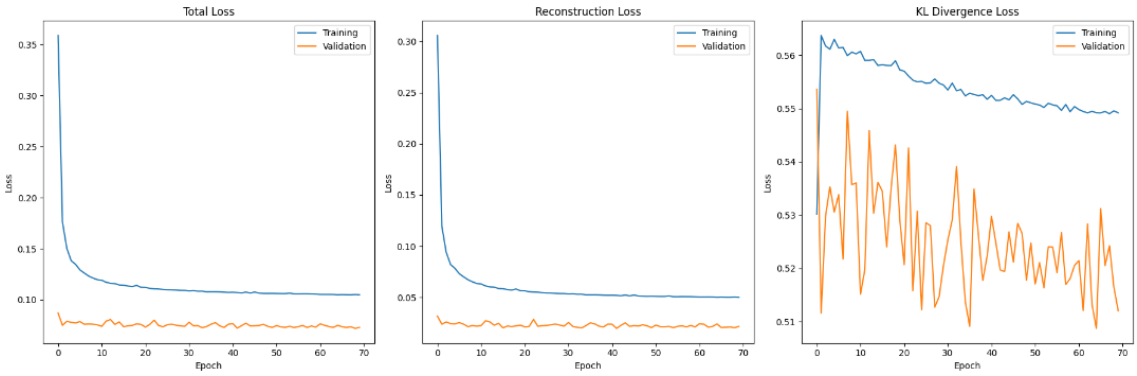


Figure 5.23: *beta-VAE learning curves*

The reconstruction loss of the test set and the latent space plots reveals some interesting aspects about the performance of the Variational Autoencoder (VAE). A notably higher overlap exists between benign and attack data in the reconstruction loss Figure 5.24 when compared to previous Autoencoders, and the distributions appear to be shifted to a higher range, with the peak coinciding with the cut-off value. This shift, however, has implications on the benign detection rate, as evidenced by the metrics. The threshold was set lower than ideal, consequently impacting the benign detection rate.

Moreover, the latent space Figure 5.25 demonstrates a different behavior compared to the rest of the Autoencoders, despite the performance being equivalent. This is manifested in the spread of attack instances across the latent space, deviating from what we've observed with other models. This difference may be attributable to the distinct properties and architecture of VAEs, hinting at their ability to model data in a unique yet effective way.

The Beta-VAE model exhibits high performance on detection rates, similar to

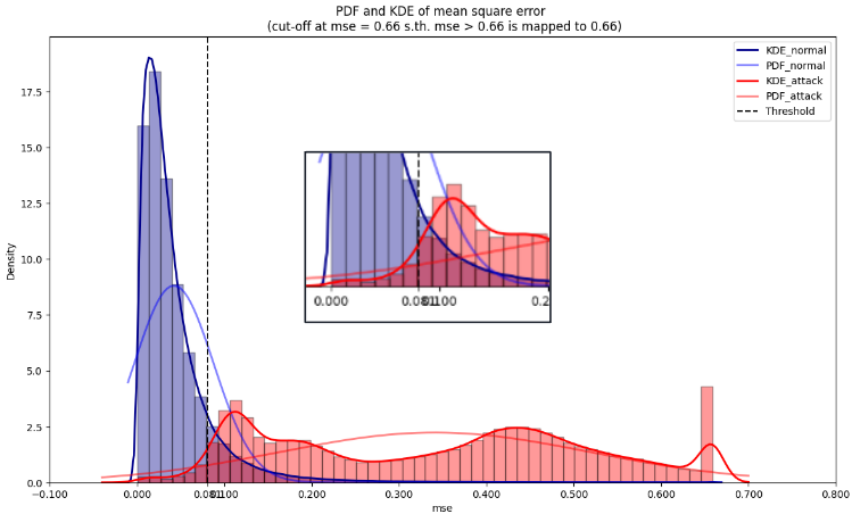


Figure 5.24: *beta-VAE test set reconstruction loss*

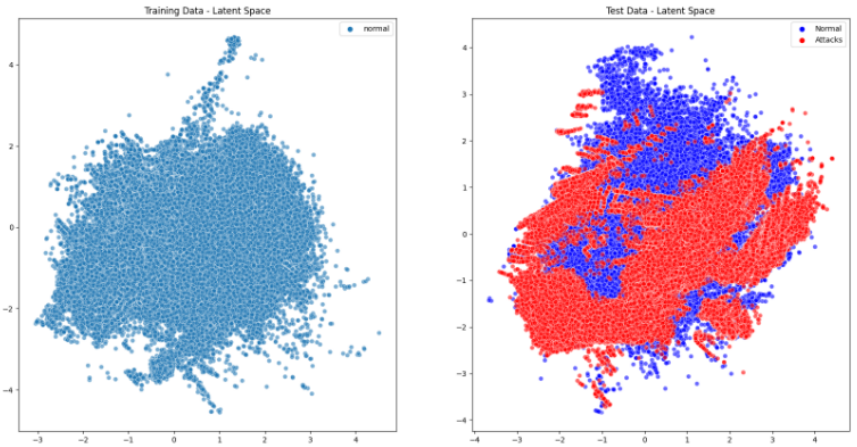


Figure 5.25: *beta-VAE latent space*

AutoEncoders mentioned earlier. This demonstrates the model's ability to strike a balance between the minimization of both Reconstruction loss and KL Divergence, though it's important to note this balance cannot be simultaneously achieved in totality. This trade-off is parameterized via the KL-coefficient (β). With an accuracy of 92.70%, weighted average precision of 93.37%, and recall of 93.08%, Beta-VAE holds its ground firmly. The ROC AUC score of 92.55% further substantiates its efficacy.

The Attacks Detection Rate stands at a commendable 96.95%, while the Normals Detection Rate is at 89.75%. This confirms that the model is efficient in identifying anomalies but falls against other models when it comes to recognize normal data because of the larger range of reconstruction loss presented.

What sets Beta-VAE apart from other AutoEncoders, however, is the differentiation it exhibits in the latent space. Despite this variation, the fact that its

performance is equivalent to the other models is highly promising, indicating that Beta-VAE brings something new to the table while not compromising on efficiency. This interesting finding opens room for further investigation and research, potentially leading to the discovery of new techniques and strategies in the realm of anomaly detection.

Metric	Value
Threshold	0.0810
Accuracy	0.927
W.Avg Precision	0.934
W.Avg Recall	0.931
W.Avg F1-score	0.932
ROC AUC	0.926
Attacks Detection Rate	0.970
Normals Detection Rate	0.898

Table 5.5: *Performance Metrics of the β-VAE using 90_{th} percentile as threshold*

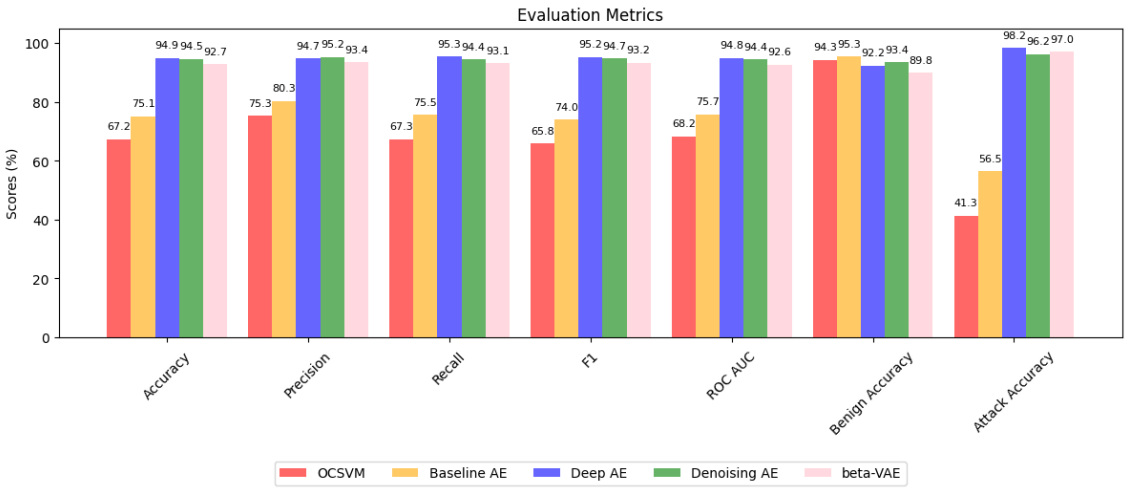


Figure 5.26: *Variational AutoEncoder performance against other models*

As we can see in the Figure 5.27 among the attack Categories we have

- **Best Accuracy:** DDoS, Portscan, Infiltration, SSH-Patator, FTP-Patator,DoS Hulk, DoS slowhttptest, DoS Slowloris, Heartbleed
- **Hard to Detect:** Botnet, Web Attack-Brute Force, Web Attack-XSS, Web Attack-Sql Injection

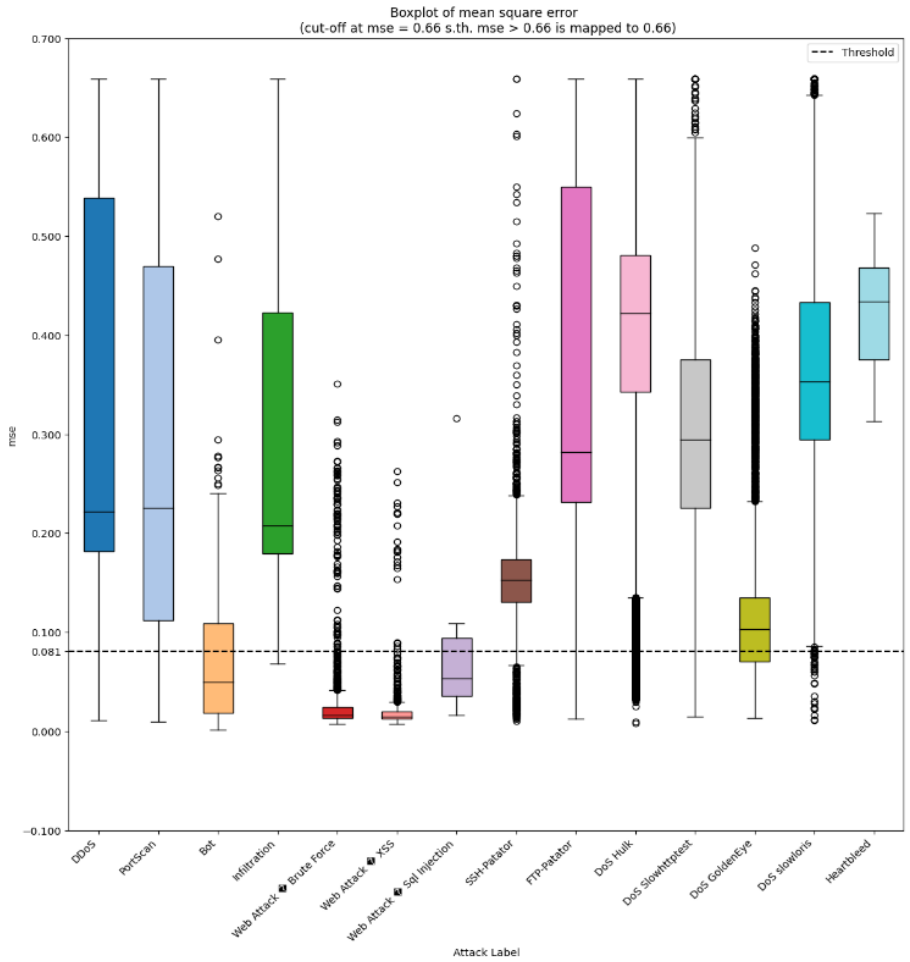


Figure 5.27: *beta-Variational AutoEncoder Box plot of MSE*

5.5 Conclusion

This thesis explored the application of AutoEncoders and their variants (DAE, beta-VAE) in the field of anomaly detection, specifically focusing on Zero-day attacks. The results were remarkable, showcasing that unsupervised learning is an effective approach in this domain, often achieving high standards. Furthermore, AutoEncoders outperformed traditional shallow machine learning algorithms, such as the One-Class SVM.

More specifically, the deep AE emerged as the best model, with the Denoising AE slightly behind by 0.4%. This outcome suggests that introducing noise can potentially enhance the model's performance when combined effectively with this approach. Although the Denoising AE is technically the second-best model, results indicated that a different choice of threshold could compensate for and even surpass the 0.4% performance gap. However, this adjustment does not apply to the best model, as the threshold was optimally set for this model.

A key factor in these experiments was the choice of a fixed threshold, based on the training reconstruction loss. Despite this being a broadly adopted method (and the reason why it was selected for this study), it seems to be sub-optimal. There is certainly potential for exploring more intuitive methods.

Another significant aspect of this thesis is the application of generative models for anomaly detection. While this may seem controversial, the results demonstrated that the advanced modeling capabilities of the VAE could provide valuable indicators for anomaly detection. It was found that the VAE was less adept at detecting anomalies. However, this could be attributed to the fact that as a generative model, its primary focus isn't perfect reconstruction, which was the main criterion used when classifying the samples. It is worth considering that the VAE might benefit from other methods of classification that might leverage their properties more effectively.

Chapter 6

Future work

This thesis has made progress in anomaly detection, but there's always more to explore. Here's a look at some potential next steps for this research, all of which could lead to even better systems for detecting intrusions.

Firstly, diving into more refined techniques like ensemble learning could be a promising route. By weaving together the decision-making capabilities of multiple models, we stand a chance to heighten overall performance, crafting a more accurate and robust Intrusion Detection System (IDS). Moreover, considering multiple evaluation criteria beyond just the reconstruction loss might offer us a more intuition of anomalies. Such an approach, capturing various aspects of the data, could undeniably pave the way for a more robust system.

Secondly, we identified the potential to further explore the capabilities of Variational AutoEncoders (VAEs). There are many aspects of VAEs that could be investigated further, such as the trade-off between the reconstruction loss and KL Divergence. Additionally, finding ways to leverage the unique properties of VAEs, such as their generative nature, could lead to improvements in anomaly detection performance.

Thirdly, Federated Unsupervised Learning could be considered for multi-domain attack detection. This would involve training models across multiple servers or domains, which could potentially improve the detection of more diverse or sophisticated cyber attacks, without exchanging privacy-sensitive data.

Next, there are many potential applications that could benefit from the techniques developed in this thesis. For example, detection of names generated by Domain Generation Algorithms for botnet traffic detection or the analysis of encrypted traffic could be considered. These applications could present unique challenges and opportunities for further development of the IDS.

Finally, the utilization of eXplainable Artificial Intelligence (XAI) techniques could be considered. XAI aims to make the operation of complex machine learning models understandable to humans, which could be particularly valuable for understanding and improving our IDS. For instance, XAI could be used to assess the

importance of different features or to understand the interactions between features, providing valuable insights for model improvement and interpretation.

These future possibilities are promising and could really improve the way we detect anomalies and intrusions..

Κατάλογος Σχημάτων

List of Figures

References

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, «A detailed analysis of the kdd cup 99 data set», in *2009 IEEE symposium on computational intelligence for security and defense applications*, Ieee, 2009, pp. 1–6.

[2] V. Kanimozhi and T. P. Jacob, «Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing», in *2019 international conference on communication and signal processing (ICCSP)*, IEEE, 2019, pp. 0033–0036.

[3] S. J. Pan and Q. Yang, «A survey on transfer learning», *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[4] U. Michelucci, «An introduction to autoencoders», *arXiv preprint arXiv:2201.03898*, 2022.

[5] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: `1312.6114 [stat.ML]`.

[6] L. Weng, «From autoencoder to beta-vae», *lilianweng.github.io*, 2018. [Online]. Available: `https://lilianweng.github.io/posts/2018-08-12-vae/`.

[7] D. Spiekermann and J. Keller, «Unsupervised packet-based anomaly detection in virtual networks», *Computer Networks*, vol. 192, p. 108 017, 2021.

[8] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, «Support vector machines», *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[9] S. Suthaharan, «Support vector machine», in *Machine learning models and algorithms for big data classification*, Springer, 2016, pp. 207–235.

[10] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[11] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, «Extracting and composing robust features with denoising autoencoders», in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[12] I. Higgins, L. Matthey, A. Pal, *et al.*, «Beta-vae: Learning basic visual concepts with a constrained variational framework», in *International conference on learning representations*, 2017.

[13] X. Yuan, C. Li, and X. Li, «Deepdefense: Identifying ddos attack via deep learning», in *2017 IEEE international conference on smart computing (SMARTCOMP)*, IEEE, 2017, pp. 1–8.

[14] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, «Survey on sdn based network intrusion detection system using machine learning approaches», *Peer-to-Peer Networking and Applications*, vol. 12, pp. 493–501, 2019.

[15] A. Aldweesh, A. Derhab, and A. Z. Emam, «Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues», *Knowledge-Based Systems*, vol. 189, p. 105 124, 2020.

[16] M. Sarhan, S. Layeghy, M. Gallagher, and M. Portmann, «From zero-shot machine learning to zero-day attack detection», *International Journal of Information Security*, pp. 1–13, 2023.

[17] S. C. Pallaprolu, R. Sankineni, M. Thevar, G. Karabatis, and J. Wang, «Zero-day attack identification in streaming data using semantics and spark», in *2017 IEEE International Congress on Big Data (BigData Congress)*, IEEE, 2017, pp. 121–128.

[18] R. Tang, Z. Yang, Z. Li, *et al.*, «Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks», in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, IEEE, 2020, pp. 2479–2488.

[19] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, «Towards an effective zero-day attack detection using outlier-based deep learning techniques», *arXiv preprint arXiv: 2006.15344*, 2020.

[20] P. Madani and N. Vlajic, «Robustness of deep autoencoder in intrusion detection under adversarial contamination», in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security*, 2018, pp. 1–8.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, «Toward generating a new intrusion detection dataset and intrusion traffic characterization.», *ICISSp*, vol. 1, pp. 108–116, 2018.

[22] N. Moustafa and J. Slay, «Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)», in *2015 military communications and information systems conference (MilCIS)*, IEEE, 2015, pp. 1–6.

[23] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, «Toward developing a systematic approach to generate benchmark datasets for intrusion detection», *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[24] M. Kuhn, K. Johnson, *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[25] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, «Survey of intrusion detection systems: Techniques, datasets and challenges», *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[26] B. M. Serinelli, A. Collen, and N. A. Nijdam, «On the analysis of open source datasets: Validating ids implementation for well-known and zero day attack detection», *Procedia Computer Science*, vol. 191, pp. 192–199, 2021.

[27] E. M. Rudd and A. Abdallah, «Training transformers for information security tasks: A case study on malicious url prediction», *arXiv preprint arXiv:2011.03040*, 2020.

[28] V. Kumar and D. Sinha, «A robust intelligent zero-day cyber-attack detection technique», *Complex & Intelligent Systems*, vol. 7, no. 5, pp. 2211–2234, 2021.

[29] L. A. Bauer and V. Bindschaedler, «Generative models for security: Attacks, defenses, and opportunities», *arXiv preprint arXiv:2107.10139*, 2021.

[30] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, «Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders», *Information Sciences*, vol. 460, pp. 83–102, 2018.

[31] I.-T. Lee, M. Marwah, and M. Arlitt, «Attention-based self-supervised feature learning for security data», *arXiv preprint arXiv:2003.10639*, 2020.

[32] H. Neuschmied, M. Winter, K. Hofer-Schmitz, B. Stojanovic, and U. Kleb, «Two stage anomaly detection for network intrusion detection.», in *ICISSP*, 2021, pp. 450–457.

[33] M. A. Salahuddin, V. Pourahmadi, H. A. Alameddine, M. F. Bari, and R. Boutaba, «Chronos: Ddos attack detection using time-based autoencoder», *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 627–641, 2021.

[34] S. Gamage and J. Samarabandu, «Deep learning methods in network intrusion detection: A survey and an objective comparison», *Journal of Network and Computer Applications*, vol. 169, p. 102 767, 2020.

[35] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, «Zero-day attack detection: A systematic literature review», *Artificial Intelligence Review*, pp. 1–79, 2023.

[36] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, «An evaluation framework for intrusion detection dataset», in *2016 International Conference on Information Science and Security (ICISS)*, 2016, pp. 1–6. DOI: 10.1109/ICISSEC.2016.7885840.

[37] A. Kamilaris and F. X. Prenafeta-Boldú, «Deep learning in agriculture: A survey», *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2018.02.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169917308803.